

**Zbierka úloh  
Korespondenčného seminára  
z programovania  
1983 – 2001**

zostavil: Michal Winczer



Katedra vyučovania informatiky  
Fakulta matematiky, fyziky a informatiky  
Univerzita Komenského Bratislava



*Korešpondenčný seminár z programovania je súťaž v tvorbe algoritmov a programovaní pre študentov stredných škôl. Zbierka obsahuje zadania a stručné návody na riešenie úloh prvých osemnástich ročníkov tejto súťaže. Materiál môže byť dobrou pomôckou pre samoštúdium a pre učiteľov na stredných aj vysokých školách môže poskytnúť zásobu vhodných príkladov pre vyučovanie. Materiál je používaný na Fakulte matematiky, fyziky a informatiky UK ako pomôcka k Didaktickému semináru zo stredoškolskej informatiky.*

1. vydanie 1996
2. vydanie 1997
3. vydanie 2001

©1996, 1997, 2001 Michal Winczer

## Úvod

Dostávate do rúk brožúrku obsahujúcu príklady prvých osemnástich ročníkov Korešpondenčného seminára z programovania (KSP). Prvé príklady KSP dostali stredoškóľáci v školskom roku 1983/84. Bolo to zásluhou šťastnej náhody, že sa na MFF UK zišiel kolektív nadšencov z radov zamestnancov Ústavu aplikovanej matematiky, niekoľkých práve prijatých a jedného staršieho študenta Katedry informatiky a ich bývalého, neúnavného a nápadmi vždy sršiaceho stredoškolského profesora Oňa Demáčka. Patronát nad práve zrodeným KSP si zobral Jozef Hvorecký z Katedry informatiky. Postupom času prichádzali ďalší študenti, bývalí riešitelia KSP. Ústav aplikovanej matematiky vystriedal Ústav informatiky, neskôr Katedra umelej inteligencie a Katedra vyučovania informatiky. V súčasnosti KSP organizuje už siedma generácia ním odchovaných študentov informatiky a zdá sa, že sa mu darí.

V prvej časti brožúrky, ktorú držíte v rukách, sú zadania úloh. Sú doplnené oproti pôvodným zadaniam o obrázky, ktoré do pôvodných zadaní úloh nebolo možné z technických dôvodov zaradiť. Niektoré úlohy sú mierne upravené, či už preto, lebo pôvodné zadanie obsahovalo chybu, alebo preto, že zadanie časom zostarło a bolo ho treba aktualizovať a výnimočne aj vtedy, keď sa ukázalo, že malá zmena zadania vedie k zaujímavejšej, prípadne poučnejšej úlohe. V roku 1983, kedy KSP vzniklo neboli počítače take bežné. Programovanie sa vyučovalo na zopár školách v celom Československu. Mikropočítače ako ZX 81, ZX Spectrum a PMD 85 vtedy iba začínali. Až neskôr sa začali rozširovať PC, PC XT a PC AT. Zo začiatku nebolo zriedkavosťou, že programy počítač vôbec “nevideli” alebo, že boli spustené na sálových počítačoch EC 1010, EC 1021 alebo minipočítačoch SM-4 alebo ADT. Prvé riešenia prichádzali výlučne v BASICu a zčasti vo FORTRANe a až neskôr sa menili na Pascal, prípadne C. Tieto všetky skutočnosti spôsobili, že obtiažnosť príkladov sa postupom času zvyšovala, príklady sú preto usporiadané podľa toho, kedy sa v seminári objavili. Každá úloha má ako pomôcku uvedené v prvom riadku v rámečku číslo z intervalu 1 až 100, ktoré určuje obtiažnosť úlohy. Obtiažnosť je uvedená vzhľadom na najlepšie riešenie známe zostavovateľovi. Riešenia príkladov označených písmenom M využívajú podstatným spôsobom matematické postupy. Príklady označené ► sú hodné špeciálnej pozornosti. Každá úloha má číslo  $xy$ , ktoré znamená, že úloha z sa objavila v  $y$ -tom kole  $xx$ -tého ročníka súťaže. V desiatom a neskôr od štrnásteho ročníka KSP navyše pribudla kategória Z pre tých, ktorí v minulosti KSP neriešili. Tieto príklady majú uvedené pred číslom úlohy ešte písmeno **z**.

Druhá časť obsahuje stručné návody k riešeniam. Väčšina z nich je iba ukazovateľom, aby ste pri ceste k riešeniu nepoblúdili. Sme si vedomí, že ich rozlúštenie môže chvíľu trvať. Mali by slúžiť len ako kontrola alebo keď si s riešením niektorej úlohy neviete dať rady. Niekoľko, podľa nás, ťažších dôležitých úloh je vyriešených kompletne. K vybraným úlohám je viacero návodov, ktoré vedú k rôzne šikovným riešeniam. V každom prípade by sa mal čitateľ snažiť vyriešiť úlohu bez využitia tejto časti, inak sa ochudobní o radosť z objavovania, nehovoriac o tom, že čo sami vymyslíme, to si aj najlepšie pamätáme.

Brožúrka si kladie za cieľ byť pomôckou pre tých, ktorí začínajú s programovaním, i pre tých, ktorí sa chcú “precvičiť” a preriešiť si niekoľko zaujímavých príkladov. Poskytnúť námety na prácu stredoškolským učiteľom informatiky a ich študentom. Našou predstavou bolo vytvoriť materiál, ktorý by zhrnul všetky príklady KSP spolu s návodmi na ich riešenie. Veríme, že riešenie príkladov vám prinesie príjemné vzrušenie a brožúrka užitočne poslúži každému, kto po nej siahne.

Na záver ešte menný zoznam všetkých, čo sa nezanedbateľnou mierou podieľali za uplynulých osemnásť rokov na organizovaní KSP: Peťo Ágh, Dušan Bezák, Danko Bieracká, Ivona Bezáková, Marek Bezaniuk, Andrej Blaho, Peťo Borovanský, Broňa Brejová, Andy Černík, Oňo Demáček, Števo Dobrev, Martin Domány, Jana Dorotková, Miro Dudík,

Lenka Fibíková, Mišo Foríšek, Janka Gajdošíková, Juro Gottweis, Maľko Hajduch, Bea Hittnerová, Dada Hoďáková-Lúčna, Sisa Horváthová, Jožo Hvorecký, Tino Irman, Dalibor Jakuš, Slavo Kavka, Branislav Katreniak, Miro Kočan, Peter g00ber Košinár, Vlado Koutný, Rasťo Kráľovič, Riško Kráľovič, Peter Lalík, Andy Lúčny, Martin Makúch, Igor Malý, Eva Mariničová, Michal Matoušek, Peťo Minárik, Peťo Miština, Martin Môťovský, Meri Nanásiová, Rišo Nemec, Monika Obrancová-Černíková, Martin Pál, David Pál, Pavol Petrovič, Pavel Petrovič, Zuzka Repaská, Zuzka Rjašková, Maroš Rusnák, Ján Senko, Erika Setteyová-Rudíková, Milan Smolík, Dano Štefankovič, Jozef Šiška, Peťo Tomcsányi, Peter Varsa, Igor Vavro, Tomáš Vinař, Marian Vittek, Mišo Winczer a Ľubo Žiško.

Neoceniteľnú pomoc preukázali Ivona Bezáková, Dušan Bezák, Broňa Brejová, Mišo Foríšek, Naňka Gajdošíková, Maľko Hajduch, Braňo Katreniak, Peter g00ber Košinár, Riško Kráľovič, Meri Nanásiová, Martin Pál, Jano Senko, YoYo Šiška a Tomáš Vinař, ktorí doplnili chýbajúce návody k riešeniam. Bez ich pomoci by tento kompletný materiál vznikol neporovnateľne dlhšie. Za pripomienky k textu ďakujeme Janke Chlebíkovej, Ľudke Moravčíkovej, Zuzke Kubincovej, Danke Pardubskej, Peťovi Borovanskému, Ivanovi Kalašovi, Ivici Chačaturianovej, Martine Hruščákovej, Majovi Dvorskému a študentom odboru Matematika–Informatika, ktorí v šk. rokoch 1994 až 2001 usilovne nachádzali množstvo pravopisných a iných chýb. V tomto treťom úplnom vydaní sme pridali index a hodnotenie obtiažnosti príkladov. V texte sme sa snažili opraviť čo najviac chýb, ale za každú ďalšiu nájdenú chybu, či už matematickú, pravopisnú, historickú alebo štylistickú, budeme vďační. Posielať ich môžete na adresy KVI FMFI UK, Mlynská Dolina, 842 48 Bratislava, alebo [winczer@fmph.uniba.sk](mailto:winczer@fmph.uniba.sk).

Celý materiál je vytvorený pomocou programu  $\text{\TeX}$ , ktorý pomáhali krotiť  $\text{\TeX}$ book a čarodejník Rišo Nemec, patrí im za to vďaka.

V Bratislave, 4. marca 2002

Michal Winczer

## Menný zoznam príkladov

- |                                       |                                    |
|---------------------------------------|------------------------------------|
| 111. O prvočíslach                    | 424. O reálnom čísle               |
| 112. Zhušťovanie poľa                 | 425. O výmenách                    |
| 113. Fibonacciho čísla                | 431. O počte výskytov              |
| 114. Generátor náhodných čísiel       | 432. O pokazenom obracači          |
| 115. Postupnosť                       | 433. O parketách                   |
| 121. Záhada piatich pokladníc         | 434. O symetrických číslach        |
| 122. O zlomkoch                       | 435. O výpise reálneho čísla       |
| 123. Rotácia poľa                     | 511. Zarovnávanie textu            |
| 124. O štvorci                        | 512. O škrtní                      |
| 125. O kódovaní permutácií            | 513. Turnaj rytierov               |
| 211. Príklad sto                      | 514. Plocha kruhu                  |
| 212. O zátvorkách                     | 515. Súvislé obrazce               |
| 213. O súčte                          | 521. O vranách                     |
| 214. O mocninách čífiel               | 522. O smetiach                    |
| 215. O písmenkách                     | 523. O najkratších cestách         |
| 221. O modelovaní                     | 524. O mape                        |
| 222. O zlomkoch                       | 525. O ramene                      |
| 223. O inverziách v permutácii        | 531. O inverzných vranách          |
| 224. O náhrdelníkoch                  | 532. O podieloch                   |
| 225. O šifrovaní                      | 533. O refazcoch                   |
| 231. O ťažobnej spoločnosti           | 534. O postupnosti                 |
| 232. O trinástom                      | 535. O šachovnici                  |
| 233. O postupnostiach                 | 541. O peniazoch                   |
| 234. O troch skupinách                | 542. O úsečkách                    |
| 235. O šifrovaní mriežkou             | 543. O permutáciách                |
| 311. O spoločnom prvku                | 544. O záhadných slovách           |
| 312. O nahrádzaní                     | 545. O kanibaloch a misionároch    |
| 313. O nerozhodných voličoch          | 611. O postupnostiach II           |
| 314. O vážení                         | 612. O vojakoch                    |
| 315. O ekvivalenciách                 | 613. O kódovaní                    |
| 321. O Pythagorejských trojuholníkoch | 614. O minimálnom absolútnom súčte |
| 322. O *-postupnostiach               | 615. O labyrintoch                 |
| 323. O interpolátore                  | 621. O múroch                      |
| 324. O šachovom koňovi                | 622. O N-uholníku                  |
| 325. O Hammingovej postupnosti        | 623. O obdĺžnikoch                 |
| 331. O grafickom okienku              | 624. O spriatelených číslach       |
| 332. O obyvateľoch ostrova            | 625. O lexikálnej analýze          |
| 333. O deliteľoch                     | 631. O delení polynómov            |
| 334. O zlomkoch (Fareyov rad)         | 632. O žabách                      |
| 335. O aritmetických výrazoch         | 633. O kresličí                    |
| 411. Číslo                            | 634. O kódovaní kombinácií         |
| 412. Kódovanie kombinácií             | 635. O zjednotení obdĺžnikov       |
| 413. Konvexný obal                    | 641. O štvorčekoch                 |
| 414. Josifova úloha                   | 642. O čudných permutáciách        |
| 415. Úloha na vyhľadávanie            | 643. Kreslenie jedným ťahom        |
| 421. O mnohouholníku                  | 644. O upratovaní                  |
| 422. O Buffonovej ihle                | 645. O kalkulačke                  |
| 423. O šachovnici                     | 711. O protivných intervaloch      |

- |  |                                      |
|--|--------------------------------------|
| 712. O bezpečnostnom vedení                    | 941. O tanečnom súbore Hatla-Patla   |
| 713. O rímskych číslach                        | 942. O Burundijskom zjednotení       |
| 714. O blábole                                 | 943. O malom lenivom krtkovi         |
| 715. O domine                                  | 944. O valcočlovekovi                |
| 721. O priemere                                | 945. Štvrtý príklad o hre BOXES      |
| 722. O veľkom zlomku                           | 1011. O Santovi a fľašiach           |
| 723. O stabilných úradníkoch                   | 1012. O Kiribatskom chráme           |
| 724. O zjednotení                              | 1013. O šikovnom učiteľovi           |
| 725. O súboroch                                | 1014. O smetiaroch                   |
| 731. O tetraminách                             | 1015. O Batuchánových vyslancoch     |
| 732. Poľský zápis                              | 1021. O dôležitej osobe              |
| 733. O telefónnych maniakoch                   | 1022. O Patagónskom fjorde           |
| 734. O cólštoku                                | 1023. O megalitickej kultúre         |
| 735. O tučnej úsečke a bystrozrakých<br>bodoch | 1024. O dopravnom ihrisku            |
| 811. O výbere                                  | 1025. O Santovi a dynamite           |
| 812. O postupnosti II                          | 1031. U holiča                       |
| 813. O zaostalej krajine                       | 1032. O švajčiarskej pechote         |
| 814. O armáde                                  | 1033. O listine kráľov               |
| 815. O hre BOXES                               | 1034. O lenivých novinároch          |
| 821. O Hilbertovej krivke                      | 1035. O hľadaní bodu                 |
| 822. O kružniciach                             | 1041. Kombinačné číslo               |
| 823. O zásobovaní                              | 1042. Katastrofa v Kiribati          |
| 824. O balíčkoch                               | 1043. Santove vrstvy                 |
| 825. Druhý príklad o hre BOXES                 | 1044. Vekslovanie                    |
| 831. O katastrofe vo východnej krajine         | 1045. Darmožráči                     |
| 832. O dierach v doske                         | z1011. O čiapočkách                  |
| 833. O spore v parlamente                      | z1012. O nešťastnom čísle            |
| 834. O vzbure v Hanibalovom tábore             | z1013. O horskom nosičovi            |
| 835. O svetovej vojne                          | z1014. O pretláčaní                  |
| 841. O orientačnom behu                        | z1021. O čiapočkách II               |
| 842. O Hanojských dvojvežiach                  | z1022. O Perzskej kráľovskej ceste   |
| 843. O mnohouholníku                           | z1023. O horskom nosičovi II         |
| 844. O prekladateľovi                          | z1024. O Santovi a burze             |
| 845. Miškova Super Úloha                       | z1031. O binárnych vektoroch         |
| 911. O Mersenových číslach                     | z1032. O horskom nosičovi III        |
| 912. O výstavbe mosta                          | z1033. $A \dots AB \dots BC \dots C$ |
| 913. O koláči                                  | z1034. "Skladanie" skupín Čiapočiek  |
| 914. O Burundijskej abecede                    | z1041. Sám od seba samého seba       |
| 915. O Froscale                                | z1042. Hoare                         |
| 921. O Kanárskych poliach                      | z1043. AB-slová                      |
| 922. O Kiribatských daniach                    | z1044. Vzorcotvorca                  |
| 923. O opitom zlatokopovi                      | 1111. O arite otáznika               |
| 924. O Burundijskej železnici                  | 1112. O súčte čísel                  |
| 925. Tretí príklad o hre BOXES                 | 1113. O Kiribatských plavidlách      |
| 931. O Santovej rúre                           | 1114. O Santovom pohľade na sídlisko |
| 932. O Kiribatskom pakovači                    | 1115. O ACM                          |
| 933. Telegram Bielym Légiam                    | 1121. O sťahovaní študentov          |
| 934. O občianskej vojne v Burundi              | 1122. Binár                          |
| 935. Zobali vrabce zobali, konvexné<br>obaly   | 1123. O byrokratoch                  |
|  | 1124. O ilegálnej organizácii        |
|  | 1125. Chodíacie písmenká             |

1131. O osemsmerovke
1132. Zátvorky
1133. O stupni čísla
1134. O Kocúrkove
1135. O formátovači
1141. O permutáciách II
1142. O delení siedmimi
1143. O prešmyčkách
1144. O poradí
1145. O preprocesore
1211. O Martowovi
1212. O brčkavých zátvorkách
1213. O Pažravom Riškovi
1214. O ovocnom sade
1215. O kvalitnej tlačiarni
1221. O výskumnom ústave
1222. O indiánskom nárečí
1223. O PNI
1224. O burundijských stavbároch
1225. O počítačoch na mieru
1231. Brutus, Frutus a partície
1232. O Santových kartičkách
1233. O Kiribatských aerolíniách
1234. O modrých prilbách v Burundi
1235. O okienkovom systéme
1241. O blšom tanci
1242. O továrni na ceruzky
1243. O veľkej poľovačke
1244. O N-trise
1245. O veľkých číslach
1311. O kiribatských klebetniciach
1312. O SoDr-e
1313. O malom lenivom krtkovi II
1314. O valcočlovekovi II
1315. Frutus, Brutus a zúrivé výrazy
1321. O kráľovstvách na Kiribati
1322. Brutus a Frutus v knižnici
1323. Banto a dostavníky
1324. O vikingskom cestovateľovi
1325. O politikoch
1331. O snaživom Tomášovi
1332. O Števovej návšteve v Indii
1333. O Kiribatských náleziskách
1334. O telefónnom víruse
1335. O SoDr a meteorológoch
1341. Brutus a Frutus v knižnici II
1342. O univerzitnom knihovníkovi
1343. O čarodejnom písacom stroji
1344. O kachličkovaní
1345. O 276. zákazníkovi
1411. O tybloni
1412. O dlhočizných dážďovkách
1413. O naladenej strune
1414. The Streets of San Benátky
1415. O Števovej sieti
1421. O koláči II
1422. O univerzitnom knihovníkovi II
1423. O vrabcovi Čin-činovi II
1424. O kiribatských náleziskách II
1425. O Števovej sieti II
1431. O magickom kruhu
1432. O koláči III
1433. O vrabcovi Čin-činovi I
1434. O Burundijských mapách
1435. O Števovej sieti III
1441. O prefikanom špiónovi
1442. O Pakovači
1443. O kľúčikoch
1444. O veľkom suchu
1445. O Števovej sieti IV
- z1411. Z univerzitnej knižnice
- z1412. Zorov závet
- z1413. Záletný zemepán
- z1414. Zátvorkove prvočísla
- z1415. Zelená žabka
- z1421. Záviš a jeho rozhlasový sen
- z1422. Ziskuchtiví zbojníci a Kiribati
- z1423. Zlato nad zlato
- z1424. Zátvorkove postupnosti
- z1425. Zrazené vláčiky
- z1431. Závišova záhradka
- z1432. Zahraničné spravodajstvo
- z1433. Zoltán tipuje
- z1434. ZGMYTDJB CHLBNB
- z1435. Záhada sennej nádchy
1511. O recidivistoch
1512. O hliadkach
1513. O leteckej spoločnosti
1514. O tvrdohlavom učiteľovi
1515. O funkcionálnom programovaní I
1521. O prevrate
1522. O komercializácii zlatokopectva
1523. O autíčkach
1524. O vrtoch v Legolande
1525. O funkcionálnom programovaní II
1531. O zlej kráľovnej
1532. O eskimákoch a iglu
1533. O veľkom smäde
1534. O vianočných darčekom
1535. O funkcionálnom programovaní III
1541. O životnom prostredí
1542. O malom Tadeášovi

1543. O prekliatost meste  
 1544. O Rastových otlakoch  
 1545. O funkcionálnom programovaní IV  
 z1511. Zorov znak  
 z1512. Zeofina spomína  
 z1513. Z Písmenkovej ulice  
 z1514. Zlo volejbalových majstrovstiev  
 z1515. Zachráňte nás!  
 z1521. Závišove slimáky  
 z1522. Zeofinina nočná mora  
 z1523. Zanzibarský slon  
 z1524. Zvedavý Jonatán  
 z1525. Záclony čarodejnice Kirky  
 z1531. Zmagorený marsochod  
 z1532. Zeofinine veže  
 z1533. Znak krásy  
 z1534. Zúfalý Santo  
 z1535. Zakrývanie koberca  
 1611. O kubomíre  
 1612. O pažravej Lívii  
 1613. O predvolebných mítingoch  
 1614. O výskumníčke Janke  
 1615. O prof. Indigovi a víle Amálke I  
 1621. O magickom symbole  
 1622. Ako Janko s Marienkou bytovú otázku riešili  
 1623. O bále  
 1624. Populárna prednáška  
 1625. O prof. Indigovi a víle Amálke II  
 1631. O svokre  
 1632. Ako si Janko s Marienkou živnosť založili  
 1633. Santo, Banto a parcela  
 1634. O reforme  
 1635. O prof. Indigovi a víle Amálke III  
 1641. O Kiribatskej mape  
 1642. O tajnej službe  
 1643. O Danovej kostre  
 1644. Lalčo lyžiarom  
 1645. O prof. Indigovi a víle Amálke IV  
 z1611. Zoči-voči – voči-zoči  
 z1612. Zuzkine narodeniny  
 z1613. Z Hanoja  
 z1614. Zanzibarský vláčik  
 z1615. Z kuchyne  
 z1621. Z krajiny byrokratov  
 z1622. Zoltánov tanečný večer  
 z1623. Zoro a zlý drak  
 z1624. Zázvorové pivo  
 z1625. Z centrálného trhoviska  
 z1631. ZY nevergreeny  
 z1632. Zákerý rozvrh  
 z1633. Zamaskuje Ferdo zamestnanie?  
 z1634. Zázvorové pivo II  
 z1635. Znižovanie zadĺženosti  
 1711. O d'Artagnanovi  
 1712. Dopravný Podnik Mesta Bratislava  
 1713. O spartakiáde  
 1714. O vekslákoch  
 1715. Hra s pešíakmi  
 1721. O krízovom štábe  
 1722. O telocviku  
 1723. O lyžiaroch  
 1724. O prederavenej streche  
 1725. O úbohom kráľovičovi  
 1731. O veľkom smäde  
 1732. O automate na lístky  
 1733. O veľkom neporiadku  
 1734. O dlhých tyčiach  
 1735. O Santovi a fľašiach II  
 1741. O vojenských zátarasoch  
 1742. O Kiribatských ponorkách  
 1743. O Veľkej cene Formule 1  
 1744. O meteorológovi Ferdovi  
 1745. O nových zlatokopoch  
 z1711. Z učárne  
 z1712. Zase SoDr  
 z1713. Zo stavby  
 z1714. Zblúdilec v zrúcanine  
 z1715. Zimbabwejský internet  
 z1721. Záhada najkratšej cesty  
 z1722. Záh(r)adne mravenisko  
 z1723. Zostarnutý papagáj  
 z1724. O včielkach  
 z1725. Zibabwejský internet II.  
 z1731. Z rodinnej kroniky  
 z1732. Zaujímavá lanovka  
 z1733. Zelené svahy Kiribati  
 z1734. Zbytočne dlhý ropovod  
 z1735. Zimbabwejský internet III.  
 1811. O vybijanej  
 1812. O prepúšťaní v TSSL  
 1813. O najkratšom tuneli  
 1814. O slepačej farme  
 1815. O Santolande I  
 1821. O bojovej sekere  
 1822. O byrokracii v Bratislave  
 1823. O zabudnutej krajine  
 1824. O nudnej prednáške  
 1825. O Santolande II  
 1831. O zúfalom programátorovi



- |   |                                 |
|---|---------------------------------|
| 1832. O maškrtnom Dávidkovi                     | z1821. Zaneprázdnený knihovník  |
| 1833. O Santovi, Bantovi a parceliach II        | z1822. Zo zapadnutej dedinky    |
| 1834. O futbalovom družstve                     | z1823. Zúfalý tesár             |
| 1835. O Santolande III                          | z1824. Zvláštne zariadenie      |
| 1841. O chemických zlúčeninách                  | z1825. Zeofína v riši fraktálov |
| 1842. O neprijemnostiach                        | z1831. Zdravotné stredisko      |
| 1843. O elixíroch                               | z1832. Zlodejov únik            |
| 1844. O novom probléme v Chile                  | z1833. Zbavme sa drobných!      |
| 1845. O Santolande IV                           | z1834. Zábavný park             |
| z1811. Zo života informačnej kancelárie         | z1835. Zeofína a zlé deti       |
| z1812. Zaoceánske Krížniky, Škunery<br>a Plavby | z1841. Z podzemného archívu     |
| z1813. Zabudnuté čísla                          | z1842. Zbytočný Miško           |
| z1814. Zauzlení studenti                        | z1843. Zvláštne poznámky        |
| z1815. Zeofína sa vracia                        | z1844. Zabudnutá pekáreň        |
|   | z1845. Zblúdila Zeofína         |



## Tematické rozdelenie príkladov

Toto delenie slúži na ľahšiu orientáciu a prípadné hľadanie vhodných príkladov. Úlohy sme rozdelili do niekoľkých skupín podľa témy zadania úlohy, prípadne podľa toho, ktorá metóda alebo dátová štruktúra vedie k dobrému riešeniu. Niektoré príklady sa preto vyskytujú vo viacerých skupinách a niektoré sa naopak nevyskytujú v žiadnej. Niektoré úlohy sme zaradili do skupiny Ťažké úlohy – tieto úlohy nebolo možné zaradiť do žiadnej z kategórií, ale napriek tomu sme vás na ne chceli upozorniť.

**Práca s poľami** 112, 113, 123, 234, 311, 312, 414, 425, 515, 524, 535, 614, 644, 721, 811, 912, 922, 933, 1012, 1042, 1121, 1131, 1342, z1411, z1421, z1431, z1531, z1621, z1712, z1722.

Úlohy, pri ktorých sa precvičuje manipulácia s poľami.

**Grafové algoritmy** 315, 433, 523, 643, 712, 733, 813, 814, 823, 831, 834, 835, 841, 924, 1014, 1021, 1024, 1044, 1123, 1124, 1224, 1233, 1243, 1311, 1313, 1344, 1414, 1434, 1444, z1413, z1433, 1513, 1522, 1532, 1543, z1523, z1531, z1533, 1613, 1621, 1623, 1631, 1643, 1644, z1624, z1634, 1711, 1721, 1723, z1721, z1731, z1812, z1842.

Úlohy, na ktorých riešenie sa používajú algoritmy a poznatky teórie grafov.

**Geometria, výpočtová geometria a počítačová grafika** 323, 331, 413, 421, 514, 621, 623, 635, 735, 822, 843, 923, 935, 943, 1023, 1035, 1043, 1113, 1114, 1134, 1234, 1321, 1411, 1421, 1431, 1441, 1541, z1511, z1514, z1524, 1612, 1622, 1632, 1641, 1724, z1732, 1813, 1823, 1833, 1841, 1843, z1841.

Úlohy zamerané na analytickú geometriu, algoritmy výpočtovej geometrie a jednoduché algoritmy počítačovej grafiky.

**Kombinatorika** 125, 212, 223, 412, 434, 543, 634, 642, 842, 1041, 1141, 1211, 1221, 1231, 1443, z1633, z1713, z1814, z1824, z1834.

Úlohy vyžadujúce prácu s kombinatorickými objektami ako sú permutácie, partície, kombinácie a postupnosti rôznych objektov.

**Matematika** 111, 122, 214, 222, 314, 321, 333, 334, 335, 411, 532, 622, 624, 633, 723, 911, 1133, 1241, z1414, z1424, z1432, 1522, 1532, z1515, z1615, z1623, z1625, 1734.

Úlohy, pri ktorých je nevyhnutné použiť znalosti z matematiky.

**Simulácia** 121, 221, 231, 313, 332, 422, 513, 612, 615, 632, 641, z1415, z1425, z1435, z1724, z1734.

Úlohy na simuláciu jednoduchých procesov — obvykle sa využíva generátor náhodných čísel.

**Semigrafika** 124, 821, z1521.

Alfanumerické obrázky.

**Spracovanie textu** 511, 725, 1034, 1135, 1222, 1521, 1614.

Úlohy zaoberajúce sa formátovaním a spracovaním textu, vyhľadávaním vzorky v texte.

**Prekladače a automaty** 424, 521, 531, 544, 625, 645, 713, 725, 732, 844, 915, 1111, 1145, 1225, 1315, 1422, 1815, 1825, 1835, 1845.

Problémy zahrňujúce oblasti teórie kompilátorov, syntaktickej analýzy a formálnych jazykov.

**Kódovanie a šifrovanie** 215, 225, 235, 613, 714, z1434, z1813, z1823, z1833, z1843.

Úlohy vyžadujúce kódovanie a šifrovanie.

**Dynamické programovanie** 213, 233, 412, 522, 611, 621, 833, 1011, 1015, 1213, 1223, 1343, 1412, 1442, 1443, 1613, 1614, 1633, 1634, 1712, 1732, 1814, 1831, 1834, 1844, z1833.

Úlohy, pri ktorých je na riešenie možné využiť metódu dynamického programovania.

**Backtracking** 211, 224, 423, 433, 715, 731, 832, 841, 913, 1214, 1244, 1313, 1323, 1432, 1523, 1534, 1544, 1642, z1612, z1614.

Úlohy, pri ktorých je jedinou vhodnou metódou backtracking.

**Rekurzia** 1112, 1122, 1132, 1142, z1512, z1513, z1522, z1525, z1532, z1613, z1733, z1834.

Príklady na precvičenie rekurzie.

**Triedenia a vyhľadávanie** 415, 432, 921, 1013, 1031, 1143, 1242, 1322, 1333, 1341, 1424, z1423, 1511, 1531, 1535, 1611, 1624, z1621, z1631, z1844.

Úlohy na triedenie a vyhľadávanie a rôzne modifikácie.

**Distribúované algoritmy** 1415, 1425, 1435, 1445.

Úlohy zaoberajúce sa problematikou distribuovaných algoritmov.

**Funkcionálne programovanie** 1515, 1525, 1535, 1545.

Jednoduché funkcionálne programy.

**Nedeterministické algoritmy** 1615, 1625, 1635, 1645.

Úlohy zaoberajúce sa problematikou nedeterministických algoritmov.

**Rozdel a panuj** 123, 213, 413, 415, 535, 642, 832, 943, 1342, z1421, 1811, 1832, 1842.

**Greedy (pažravé) algoritmy** 613, 824, 1011, z1023, z1024, z1535, 1812.

**Induktívne** 115, 125, 213, 233, 325, 522, 543, 644.

**Ťažké príklady** 512, 534, 734, 812, 845, 931, 1113, 1342, 1514.

**Optimalizačné úlohy** 1634, z1632.

**111. O prvočíslech**

[10]

Zostavte program, ktorý rozloží dané prirodzené číslo  $n$  (z intervalu 1 až maximálne celé číslo zobraziteľné na počítači) na prvočinitele a vypíše

- číslo  $n$
- text **Je prvočíslo**, alebo **je deliteľné prvočísлом  $i$  presne  $j$  krát**, pre všetky prvočísla  $i$  a ich násobky  $j$  v rozklade.

**112. Zhušťovanie poľa**

[5]

V poli  $a[1..n]$  je veľa núl. Zhustíte prvky v poli  $a$ . tým, že všetky nuly z neho odstránite. Počet nenulových prvkov zhusteného poľa bude uložený v premennej  $k$ . Vypíšte

- obsah pôvodného poľa
- obsah zhusteného poľa a počet jeho nenulových prvkov

**113. Fibonacciho čísla**

[10]

Fibonacciho<sup>1</sup> čísla sú čísla definované rekurentnou postupnosťou:

$$F_0 = 0, \quad F_1 = 1, \quad F_{n+2} = F_{n+1} + F_n, \quad \text{pre } n \geq 0$$

Teda každé Fibonacciho číslo je súčtom dvoch predchádzajúcich. Na výpočet hodnôt Fibonacciho čísel však existuje aj iný vzorec:

$$\Phi_k = \frac{1}{\sqrt{5}} \left( \frac{1 + \sqrt{5}}{2} \right)^k - \frac{1}{\sqrt{5}} \left( \frac{1 - \sqrt{5}}{2} \right)^k$$

ÚLOHA: Zostavte program, ktorý bude počítať prvých 15 Fibonacciho čísel oboma spôsobmi a vytlačí

- presnú hodnotu podľa prvého vzorca
- presnú hodnotu podľa druhého vzorca
- absolútnu odchýlku, t.j. absolútnu hodnotu ich rozdielu

**114. Generátor náhodných čísel**

[5]

Generátorom pseudonáhodných čísel rozumieme postupnosť, v ktorej sa členy striedajú natoľko nepravidelne, že ich poradie je pre toho, kto nepozná predpis, ktorý poradie udáva, úplne náhodné. Dobrými generátormi pseudonáhodných čísel sú

- pre 32 bitový počítač, podľa [22],

$$x_{n+1} = (x_n * 5621 + 1) \bmod 65536,$$

ktorý generuje čísla z intervalu 0 až 65535. Rozsah nemôže prekročiť 65535, lebo mod znamená zvyšok po delení.

- Pre počítač s reálnou aritmetikou

$$x_{n+1} = \text{desatinná časť čísla}((x_n + \pi)^5),$$

ktorý generuje reálne čísla z intervalu 0 až 1, pretože výsledkom je zvyšok po odtrhnutí celej časti.

ÚLOHA: Zostavte generátor náhodných čísel ako podprogram, ktorý bude generovať s rovnakou pravdepodobnosťou niektoré z čísel 1, 2, 3, 4, 5 alebo 6 tým, že uvedené intervaly pseudonáhodných čísel rozdelíte na 6 rovnakých častí. Takýto generátor teda môže nahradiť hraciu kocku. Predstavme si teraz, že hádzeme šiestimi kockami. Pri každom vrhu môže padnúť jedno z čísel 6 až 36. Hoďte kockami 100, 300, 500 krát a zistite, koľkokrát padol každý z 31 možných súčtov. Výsledky vytlačte, a vykreslite pomocou rôzne dlhých úsečiek.

<sup>1</sup> Leonardo Pisano nazývaný aj Fibonacci, 1180–1240, najväčší predrenesančný európsky matematik. Postupnosť  $F_n$  pomenoval jeho menom až francúzsky matematik E. Lucas, 1842–1891.

**115. Postupnosť**

► [20]

Zostavte program, ktorý bude vo vzostupnom poradí vytvárať 100 najmenších čísiel v množine  $M$ , ak je  $M$  definovaná nasledujúco:

- i) číslo 1 je z  $M$
- ii) ak číslo  $x$  je z  $M$ , tak aj čísla  $2x + 1$  a  $3x + 1$  sú z  $M$
- iii) žiadne iné čísla, než tie, ktoré vzniknú pomocou pravidiel i) alebo ii) nepatria do  $M$ .

Na ukážku uvedieme prvých sedem prvkov množiny  $M$ : 1, 3, 4, 7, 9, 10, 13, ...

**121. Záhada piatich pokladníc**

[10]

TJ Rozpaky nad Ofsajdom usporiadala na svojom štadióne majáles. Pokladník oddielu umiestnil pri každej zo štyroch brán štadióna jedného svojho pomocníka s pokladnicou. Keď sa zdalo, že už ďalší návštevníci neprídu, obišiel všetkých pomocníkov, preložil peniaze z ich pokladníc do svojej a oddal sa zábave.

Keď už bol mierne „v nálahe“, uvedomil si, že ak by stratil kľúče, nálezca by sa mohol ľahko zmocniť tržby. Preto pokladnice pootváral, do každej hodil po jednom kľúči a opäť ich zatvoril. Tým ich aj zamkol, lebo sa zamykali automaticky.

Až ráno (po vytriezvení) si uvedomil dôsledky svojho činu, veď do pokladníc sa nedostane ani on sám! To ale znamená, že najmenej jednu pokladnicu musí násilne vylomiť, aby sa dostal k niektorému kľúču. Tým kľúčom bude môcť otvoriť ďalšiu pokladnicu (ak to náhodou nebude kľúč od práve vypáčenej pokladnice). V nej bude ďalší kľúč atď.

ÚLOHA:

- a) Pomocou generátora pseudonáhodných čísel nasimulujte náhodné rozloženie kľúčov v piatich pokladniciach. Zistite, koľko pokladníc treba otvoriť násilím na to, aby sme ich otvorili všetky.
- b) Zopakujte pokus 100-krát a zistite, koľkokrát musel pokladník vylomiť práve jednu, práve dve, práve tri, štyri, päť pokladníc, aby otvoril všetky. Aký je priemerný počet vylomených pokladníc?

**122. O zlomkoch**

► [M50]

Pre dané celé čísla  $m$ ,  $n$  vypočítajte podiel ako reálne číslo na taký počet desatinných miest, aby ste skončili delenie vtedy, keď:

- i) ďalšie desatinné miesta obsahujú iba nuly,
- ii) perióda v desatinnej časti sa začne opakovať. V tomto prípade určite aj dĺžku periódy.

Výpočet odlaďte aspoň na 5 podstatne odlišných dvojiciach, napríklad 523 a 19, 1 a 17, 10192 a 52 a pod.

**123. Rotácia poľa**

► [30]

Dané je  $n$  prvkové pole  $x$ . Zostavte program, ktorý pre dané celé číslo  $k$ ,  $-n < k < n$  posunie prvky poľa „do kruhu“. Pre kladné  $k$  to znamená, že prvok  $x[0]$  sa presunie na miesto  $x[k]$ , prvok  $x[1]$  na miesto  $x[k+1]$  atď. Na mieste  $x[0]$  bude prvok, ktorý bol pôvodne na mieste  $x[n-k]$ , na mieste  $x[k]$  ten, ktorý bol na mieste  $x[n-1]$ . Ak teda v 8-prvkovom poli boli prvky 1 2 3 4 5 6 7 8 a  $k = 3$ , tak výsledkom „rotácie poľa“ bude 6 7 8 1 2 3 4 5. Pre  $k = 0$  všetky prvky zostanú na svojich miestach. Pre záporné  $k$  pôjde o rotáciu o  $k$  prvkov opačným smerom.

**124. O štvorci**

[10]

Dané je číslo  $n$ ,  $20 < n < 100$ . Zostavte program, ktorý vytlačí čísla 1, 2, 3, ...,  $n$  „po obvode štvorca“ (so stranou dĺžky približne  $n/4$ ). Pri párnych  $n$ , ktoré nie sú deliteľné štyrmi, skráťte vodorovné strany štvorca, pri nepárnych  $n$  nebude ľavá zvislá strana úplná. Pre  $n = 21$  bude mať „štvorec“ podobu ako na obrázku.

1	2	3	4	5	6
					7
21					8
20					9
19					10
18					11
17	16	15	14	13	12

**125. O kódování permutací**

[20]

Pod permutáciou  $n$  čísel rozumieme takú  $n$ -ticu, že každé z čísel  $1, 2, 3, \dots, n$  sa v nej nachádza práve raz. Permutácie môžeme usporiadať na podobnom princípe ako mená v telefónnom zozname. Toto usporiadanie nazývame *lexikografickým usporiadaním*. Jeho princíp je nasledujúci:

Permutácia  $a_1 \dots a_n$  je v zozname permutácií pred permutáciou  $b_1 \dots b_n$  vtedy a len vtedy, keď:

- i)  $a_1 < b_1$
- ii) alebo sa permutácie zhodujú v niekoľkých prvých členoch, ale keď  $a_i$  a  $b_i$  sú prvé členy, v ktorých sa odlišujú, tak  $a_i < b_i$ .

Napríklad, pre permutácie 5 čísel je permutácia 3 2 4 1 5 pred permutáciou 4 1 5 3 2 podľa princípu vyjadreného v bode i) a pred permutáciou 3 2 5 4 1 podľa princípu, vyjadreného v bode ii). V tomto prípade bola rozhodujúca odlišnosť tretieho člena, lebo prvé dva sa rovnali.

Teraz si predstavte, že všetkých  $p$  permutácií  $n$  čísel je napísaných v stĺpci na papieri podľa predchádzajúceho usporiadania. Očíslujeme permutácie od najmenšej po najväčšiu číslami  $0, 1, \dots, p - 1$ .

ÚLOHA: Zostavte program,

- a) ktorý, keď dostane na vstupe permutáciu  $n$  čísel (pre  $n < 8$ ), určí jej poradové číslo v zozname permutácií príslušného  $n$ . Číslo budeme nazývať kód permutácie.
- b) ktorý, keď prečíta kód permutácie a príslušné  $n$ , vytlačí permutáciu, zodpovedajúcu tomuto kódovému číslu.

Riešenie, ktoré využíva tabuľku všetkých permutácií v pamäti pokladáme za „škarredé“.

**211. Príklad sto**

[18]

Na papieri je napísané dlhé číslo 123456789. Vašou úlohou je medzi niektoré dvojice čífer vsunúť znak  $+$  alebo  $-$  tak, aby ste po vyhodnotení výrazu dostali číslo 100. Ak medzi ciframi nie je vsunutý žiaden znak, tak sa chápu ako súvislá konštanta. Vymieňať poradie čífer nesmiete.

Príklad :  $100 = 123 - 45 - 67 + 89$

ÚLOHA: Napíšte program, ktorý nájde a vypíše všetky možnosti vsunutia znakov  $+$  a  $-$  do reťazca a na konci vypíše počet riešení.

Ak váš program bude dobre napísaný (použijete jasnú dobrú ideu), tak by ste mali byť schopný ľahko do programu dorobiť aj iné aritmetické operácie.

**212. O zátvorkách**

► [30]

Každý školáčik vie, čo sú pekne uzátvorkované aritmetické výrazy. Sú to len také, že ku každej ľavej zátvorke zodpovedá pravá zátvorka.

Napríklad  $((()))()$  je dobre uzátvorkovaný výraz, ale  $()()$ ,  $)(()$ ,  $((()$ , ... nie sú dobre uzátvorkované výrazy. Presnejšie povedané :

1.  $()$  je dobre uzátvorkovaný výraz.
2. Ak výraz  $X$  je dobre uzátvorkovaný, tak  $(X)$  je dobre uzátvorkovaný výraz.
3. Ak  $X, Y$  sú dobre uzátvorkované výrazy, tak  $XY$  je dobre uzátvorkovaný výraz.

ÚLOHA: Máte k dispozícii  $n$  párov zátvoriek  $(, )$ . Zostavte program, ktorý vypíše všetky dobre (pekne) uzátvorkované výrazy (dĺžky  $2n$  zátvoriek).

Napríklad ak  $n = 3$ , tak vypíšte  $((()))$ ,  $(())()$ ,  $()(())$ ,  $((())())$ ,  $()(())()$ .

Dávajte pozor na efektívnosť vášho algoritmu.

**213. O súčte**

► [22]

Daná je postupnosť  $n$  celých (kladných i záporných) čísiel. Nájdite v nej podpostupnosť (ľubovoľnej dĺžky  $\leq n$ ) po sebe nasledujúcich členov postupnosti s najväčším súčtom.

Dobře pošpekulujte nad najlepším a najrýchlejším riešením. Problém je ťažší, než sa na prvý pohľad zdá.

#### 214. O mocninách cifier

20

Zoberte prirodzené číslo, rozdeľte ho na cifry a vypočítajte súčet druhých mocnín týchto cifier. Dostanete nové číslo, s ktorým opakujte tento postup. Po konečnom opakovaní tohto postupu dostávame číslo 1 alebo 4. Ak napríklad začneme s číslom 324, máme

$$\begin{aligned}a_1 &= 324 \\a_2 &= 3 \cdot 3 + 2 \cdot 2 + 4 \cdot 4 = 29 \\a_3 &= 2 \cdot 2 + 9 \cdot 9 = 85 \\a_4 &= 8 \cdot 8 + 5 \cdot 5 = 89 \\a_5 &= 8 \cdot 8 + 9 \cdot 9 = 145 \\a_6 &= 1 \cdot 1 + 4 \cdot 4 + 5 \cdot 5 = 42 \\a_7 &= 4 \cdot 4 + 2 \cdot 2 = 20 \\a_8 &= 2 \cdot 2 + 0 \cdot 0 = 4 \dots \dots \text{koniec postupu}\end{aligned}$$

ÚLOHA: Zistite, ktoré čísla z intervalu 1 až 1000 končia po aplikovaní uvedeného postupu číslom 1.

#### 215. O písmenkách

5

Pracovníci oddelenia počítačovej literatúry potrebujú ako podklad pre svoju novú knihu tabuľku výskytu jednotlivých znakov v slovenskom texte. Preto Vás prosia o spoluprácu. Vašou úlohou bude napísať program, ktorý vytlačí tzv. frekvenčnú tabuľku.

ÚLOHA: Vezmite nejaký súvislý text v slovenčine (z novín, knihy, časopisu a pod.), dlhý cca. 200 slov. Pre účely programu chápťte aj všetky ďalšie znaky (čiarka, bodka, medzera a pod.) ako písmená slovenskej abecedy. Pre každé písmeno abecedy zistite počet jeho výskytov v texte. Výslednú tabuľku usporiadajte podľa poradia výskytu a oznámte percentuálny podiel znaku oproti počtu všetkých znakov v texte.

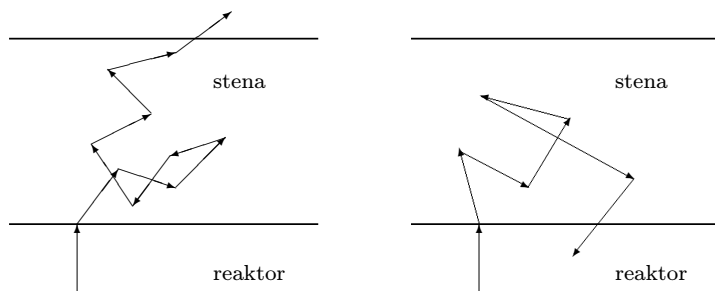
#### 221. O modelovaní

5

Okolo atómového reaktora treba vybudovať ochrannú betónovú stenu, aby rádioaktívne častice nezamorovali okolie. Vieme, že hrúbka steny 5 metrov je postačujúca, lebo energia častíc vyletujúcich z reaktora sa znižuje kolíziami s časticami v betóne, ktoré ich zabrzdia. Na časticu v betóne narazia každých 25 cm a 20 takých nárazov časticu úplne zastaví. (v skutočnosti to platí len „v priemere“, ale pre účel nášho modelovania predpokladáme, že je to vždy tak). Po každom náraze na časticu v betóne sa častica odchyli od smeru, ktorým sa pohybovala do stretnutia, o uhol  $\alpha$ . Jeho veľkosť je náhodná veličina z intervalu  $(0, 2\pi)$ . Teda v skutočnosti nemusí byť stena taká hrubá, lebo dráha častice v stene nevedie priamo von z reaktora. V stene tenšej sú možné tri prípady (i), ii) sú na obrázku):

- i) častica sa vráti späť do reaktora
- ii) častica preletí cez stenu
- iii) častica narazí 20 krát v stene na častice v betóne a zastaví sa





ÚLOHA: Napište program, ktorý bude simulovať prechod častice betónovou stenou. Častice vypúšťajte do steny zo strany reaktora tak, že počiatočná vzdialenosť častice a steny je nulová. Náhodne zvolte aj uhol, pod ktorým častica do steny vletí. Vypíšte percentuálny podiel každého prípadu a, b, c, pre n (niekoľko sto) vypustených častíc a hrúbku steny 2, 3, 4, 5 metrov.

## 222. O zlomkoch

M30

Napište program, ktorý pre dané reálne čísla  $x$  a  $e$  vypočíta nesúdeliteľné celé čísla  $a$  a  $b$  tak, aby  $|x - a/b| \leq e$ , t.j. aby sa reálne číslo dalo nahradiť zlomkom  $a/b$  s presnosťou  $e$ . Vytlačte vstupné hodnoty  $x$ ,  $e$  a výsledky - vypočítané  $a$  a  $b$ .

Napríklad pre  $x = 2.718$  a  $e = 0.01$  je  $a = 19$  a  $b = 7$ . Skúste pre  $x = 3.1415926$ ,  $e = 0.001$  a pre  $e = 0.0001$ .

## 223. O inverziách v permutácii

10

$a_1 a_2 \dots a_n$  je permutácia množiny čísel  $1, 2, 3, \dots, n$ . Tabuľkou inverzií permutácie  $a_1 a_2 \dots a_n$  sa nazýva postupnosť čísel  $b_1, b_2, \dots, b_n$ , kde  $b_j$  je počet prvkov väčších než  $j$  a nachádzajúcich sa vľavo od  $j$  v permutácii  $a_1 a_2 \dots a_n$ . Vymyslite algoritmus vhodný pre realizáciu na počítači, ktorý podľa danej tabuľky inverzií  $b_1, b_2, \dots, b_n$  skonštruuje permutáciu  $a_1 a_2 \dots a_n$ . Algoritmus realizujte programom pre počítač.

Príklad:

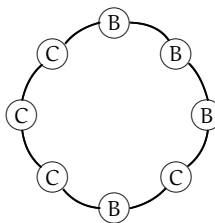
	$a_1$	$a_2$	$a_3$	$a_4$	$a_5$
permutácia	2	5	4	1	3
tabuľka inverzií	3	0	2	1	0
	$b_1$	$b_2$	$b_3$	$b_4$	$b_5$

## 224. O náhrdelníkoch

25

Princezná Arabela sa rada parádila pred dvornými programátormi, a preto nosila do „výpočtáku“ každý deň iný „pekný“ náhrdelník. Páčili sa jej len také dvojfarebné náhrdelníky z perál, že programátori, ktorí sa okolo nej krútili, na ňom mohli nájsť ľubovoľnú farebnú podpostupnosť dĺžky  $n$  (vstupný údaj).

Príklad:  $n = 3$ ; Na obrázku je náhrdelník, ktorý princezná jeden deň nosila. Je naozaj „pekný“, lebo z neho môžeme prečítať všetkých  $2^3 = 8$  variácií s opakovaním dĺžky 3 z dvoch prvkov (čítame v smere chodu hodinových ručičiek). Náhrdelník neotáčame naopak ani nerozdeľujeme. Teda prečítame BBB, BBC, BCB, CBC, BCC, CCC, CCB, CBB. Pre istotu si skontrolujte, či sme vypísali všetky možné trojprvkové variácie dvoch farieb (nemusia byť v tradičnom



poradí). Toto bol jediný úspešný krok Arabely preniknúť do tajov dvojkovej sústavy a tak ovládnuť výpočtovú techniku.

ÚLOHA: Arabela vždy nosí náhrdelník dĺžky  $2^n$  a programátori hľadajú postupnosti dĺžky  $n$  (ukázané pre  $n = 3$ ). Zistíte koľko a akých pekných náhrdelníkov má, keď viete, že sú navzájom rôzne. Za rôzne považujeme aj také, ktoré vzniknú rotáciou, alebo symetrickým preklopením šperku. Pokúste sa nájsť algoritmus, ktorý neprehľadáva všetkých  $2^{2^n}$  možností, to je príliš veľa.

## 225. O šifrovaní

[12]

Pomocou rôznych programov a iných pomôcok musíte rozšifrovať nasledujúci text. Text vznikol zo slovenského textu knihy, ktorej názov tiež môžete hádať. Zachovali sme interpunkčné znamienka, t.j. čiarky, bodky, výkričníky, otázniky, atď. Nahrádzované boli len písmená a až z tak, že všetky výskyty písmena f sa nahradili napríklad písmenom t. Teda, že každé t znamená f, a zároveň žiadne dve písmenká sa nezobrazili na to isté písmenko. Matematici hovoria, že sme urobili len permutáciu písmen abecedy. Mäčkene a dĺžne sa nebrali do úvahy a nerozlišovali sme medzi veľkými a malými písmenami. Pokúste sa rozšifrovať nasledujúci text pomocou frekvenčnej tabuľky, ktorú ste robili v úlohe 215.

Tu je text:

UEUSF PLOTK.

PLOMTCSP HK AYM OKUKTPEYM FPKNLTPR, PNEIM FK EA FMGK YRIKOCM  
TSFSK CKTMUPKHS. DMACK FPKNLTPK HK CKTMUPL FYMNTL, AILJK NHKYL.  
EGM FPKNLTPR FK AKDL CK UIYR UEJTKA TKJPE IEEOCKN. UIMAYKAOKDLVS  
OKGKTS FPKNLTPR FE FYMNTL CKTMUPEL AE CEYSCEYMJE UKUSMIK K  
UETEOS DL EA FMGK CK TKYE. UENEH OKGKTS AE CEYSCEYMJE UKUSMIK  
AILJL FPKNLTPR, PNEIK HK NHKYL CKTMUPL, K UETEOS DL EA FMGK  
CKUIKYE. YMOHM PLOMTCL UKTSVPL K LIEGS CKA EGEHK FPKNLTPKHS  
CSMPETPE PLOMTCRVJ NKJEY. PMA CEYSCR IEOKGTS LPKOM FK, OM  
FPKNLTPK FE FYMNTL CKTMUPEL FK CKVJKOK YUIKYE K FPKNLTPK  
F NHKYL CKTMUPEL YTKYE.EGSAYM FPKNLTPR FS YRHCSTS HSMFNE.  
CK UIMYMACSM PLOTK UENIMGLDMHM AYM UIMUKIEYKCM OKUKTPEYM  
FPKNLTPR K AYK PLFR CEYSCEYMJE UKUSMIK. EGFNKKHM FS AYM  
FPKNLTPR FE FYMNTL K AYM FPKNLTPR F NHKYL CKTMUPEL. O DMACMD  
FYMNTMD K F DMACMD NHKYMD FPKNLTPR CKTMUPR EAHEVSHM K FNSKJCMHM.  
YRFLFSHM SVJ K CMUEFPPEAMCM CKTMUSHM CK AYK EGATOCSPR O  
PIMFTSKVMJE UKUSMIK.NSM HLFSS GRN IEYCKPE YMTM KPE OKUKTPEYK  
FPKNLTPK.EGATOCSPR F CKTMUMCRHS CKTMUPKHS HLFSSM OKNKOSN K  
AEPECKTM YRFLFSN,KGR FK CMPISYSTS. UENEH SVJ LUIKYSHM NKP, KGR  
GETS UIMFCRH EGIKOEY YIVJCMD FNIKCR FPKNLTPR. KP HK NMAK YIVJCK  
FNIKCK FPKNLTPR EPETE CKTMUPR HEAIR UKFSP, HLFSSM JE CKHKTEYKN  
S CK UKUSMIEYR EGATOCSP EPETE DMJE CKTMUPR. UENEH CK FPKNLTPR  
FE FYMNTL CKTMUPEL UETEOSH EGATOCSP F CKTMUPEL NHKYL, K OKFM  
CKEUKP CK FPKNLTPR F CKTMUPEL NHKYL UETEOSH EGATOCSP FE  
FYMNTL CKTMUPEL. EGM FPKNLTPR UETEOSH CK FNET NKP, KGR GETS  
HSHE AEFKJL ASYKPEY. EPIMH FPKNLTSMP UETEOSH CK FNET AYK  
JKIPR CEYSCEYMJE UKUSMIK.  
Y VEH DM YTKFNCM UEAFNKNK PLOTK? PLOMTCSP YMOHM FPKNLTPR, CK  
PNEIMDMOS EGATOCSP F CKTMUMCEL FYMNTL CKTMUPEL, KTM FPKNLTPK  
HK Y FPLNEVCEFNS CKTMUPL NHKYL. OAEIKOCS OM GKTS FYMNTL  
FPKNLTPL K UETEOS DL EA FMGK CK TKYE AILJL FPKNLTPR, CK PNEIMDMOS  
EGATOCSP F NHKYL CKTMUPEL KTM FPKNLTPK HK Y FPLNEVCEFNS  
FYMNTL. CKTMUPL OKGKTS AE AILJMJE CEYSCEYMJE UKUSMIK K UETEOS

DL EA FMGK CKUIKYE. CMOKGLACM AEAKN, OM NHKYK FPKNLTK TMOS  
 YUIKYE. YMOHM PLOMTCL UKTSVPL K LIEGS CKA EGEHK OKGKTMCRHS  
 FPKNLTPKHS CSMPTPE HKBSVPRVJ NKJEY. EOCKHS, OM UE NRVJNE  
 NKJEVJ FS FPKNLTPR YRHMCSSTS HSMFNE. CK AEPKO FYEDJE NYIAMCSK  
 IEOGKTS CKDUIY TKYL FPKNLTP. UIS IEOGKTEYKCS CMVJK Y CEYSCEYEH  
 UKUSMIS S EGATOCSP FE FYMNTTEL CKTMUPEL. UE IEOGKTMCFS FK NMAK  
 EGDKYS YTKYE FPKNLTPK F NHKYEL CKTMUPEL. UIS IEOGKTEYKCS NHKYMD  
 FPKNLTPR CMVJK Y CEYSCEYEH UKUSMIS EGATOCSP F NHKYEL CKTMUPEL.  
 NKPOM NMIO DM YUIKYE FPKNLTPK FE FYMNTTEL CKTMUPEL. EGSAYM  
 FPKNLTPR FS NMAK YRHMCSSTS HSMFNE.  
 AETMOSNM LUEOEICMCSM !  
 FE FPKNLTPKHS, CK PNEIRVJ YETCM TMOSK EGATOCSPR F SCRHS  
 CKTMUPKHS NIMGK HKCSULTEYKN EUKNICM, KGR FK EGATOCSPR CMUEFLCLTS  
 NRH GR FK UIMOIKASTK UEA FNKNK NISPL.

### 231. O ťažobnej spoločnosti.

[5]

Ťažobná spoločnosť sa rozhodla investovať 10 miliónov korún do nového náleziska ropy. Pripravná technická skupina podala tri alternatívne návrhy postupu:

- Na existujúcom vrtnom zariadení vykonať 10 vrtov, každý v cene 1 mil. Kčs. Pravdepodobnosť úspešného vrtu týmto zariadením je 25% a v prípade úspešného vrtu dá takýto zdroj 10 000 ton ropy ročne.
- Kúpiť nové prieskumné zariadenie za 3 mil. Kčs. Po predbežnom prieskume vykonať 7 vrtov (po 1 mil. Kčs). Pravdepodobnosť úspešného vrtu stúpne na 50%, kapacita zdroja bude 10 000 ton ročne.
- Kúpiť nové prieskumné zariadenie za 3 mil. Kčs a novú vrtnú súpravu za 4 mil. Kčs. Na vykonanie vrtov sice ostane iba 3 mil. Kčs (t.j. 3 vrty), ale pravdepodobnosť úspechu bude 50% a kapacita zdroja stúpne na 18000 ton ročne.

ÚLOHA: Napíšte program, ktorý simuluje 100 pokusov pre každú z troch stratégií a pre každú stratégiu vypíše priemerný ročný výťažok zo všetkých úspešných vrtov a priemerný čas (v mesiacoch), za ktorý sa vyťažilo 100 000 ton ropy. Za najlepšiu stratégiu vyhlási tú, pri ktorej je tento čas najmenší.

Dobre si rozmyslite, ako sa dá pomocou generátora náhodných čísel z intervalu  $(0, 1)$  generovať (áno, nie) tak, aby pravdepodobnosť „áno“ bola  $p\%$  (25 alebo 50).

### 232. O trinástom

[18]

Iste všetci poznáte poveru o nešťastnej trinástke. Niektorí ľudia sa boja vyjsť trinásteho von, aby sa im niečo nestalo. Tobôž keď padne trinásteho na piatok.

ÚLOHA: Napíšte program, ktorý vyrobí takúto štatistiku o trinástom dni v mesiaci: pre každý deň v týždni (pondelok, ..., nedeľa) zistí koľkokrát v dvadsiatom storočí (t.j. od 1. 1. 1901 do 31. 12. 2000) pripadol (alebo pripadne) na tento deň dátum trinásteho v mesiaci. Program má výsledky vypísať v tabuľke.

Pomôcka: 1.1.1901 bol utorok

### 233. O postupnostiach

[25]

Napíšte program, ktorý vytlačí pre dané  $n$ ,  $1 \leq n \leq 12$ , počet takých postupností, obsahujúcich len nuly a jednotky, dĺžky  $n$ , v ktorých sa nevyskytujú za sebou dve nuly.

Skúste program napísať tak, aby neskúšal všetkých  $2^n$  postupností dĺžky  $n$ .

### 234. O troch skupinách

[15]

V poli  $P$  je  $n$  celých čísel - kladné, záporné a práve jeden nulový prvok. Napíšte program, ktorý pre dané  $n$  a  $P$  „utriedi“ pole tak, že na jeho začiatku budú všetky záporné prvky, potom nulový prvok a na konci všetky kladné. Na poradí čísel v jednotlivých skupinách nezáleží.

Pokúste sa vymyslieť algoritmus, ktorý nepoužije všeobecné triedenie.

### 235. O šifrovaní mriežkou

[15]

V príklade 225 sme sa zoznámili s jednou z najjednoduchších metód šifrovania – jednoduchým substitučným kódom. Teraz si popíšeme inú šifrovaciu metódu: šifrovacou mriežkou  $n \times n$  pre párne číslo  $n$  nazveme štvorec (napríklad z papiera) rozdelený na  $n \times n$  štvorčekov. Z týchto štvorčekov je  $\frac{1}{4}n^2$  štvorčekov vystrihnutých. Správna šifrovacia mriežka je taká, ktorú keď položíme na iný štvorec  $s \times n$  políčkami a postupne ju otáčame o štvrtkruh v smere hodinových ručičiek (čím dostaneme 4 polohy), tak nám vystrihnuté okienka odkrývajú vždy iné štvorčeky.

Správnou šifrovacou mriežkou sa dá zašifrovať text o  $n \times n$  písmenách takto: nakreslíme štvorec so stranou  $n$  a urobíme v ňom sieť  $n \times n$  políčok. Na tento štvorec položíme šifrovaciu mriežku. Cez jej vystrihnuté okienka začíname písať postupne písmená šifrovaného textu. Keď vyplníme prvých  $\frac{1}{4}n^2$  okienok, otočíme mriežku o štvrtkruh v smere hodinových ručičiek. Pretože je to správna šifrovacia mriežka, odkryje nám iných  $\frac{1}{4}n^2$  políčok v našom štvorci kam môžeme napísať ďalšie písmená šifrovaného textu. Po dopísaní druhej časti mriežku znova pootočíme a po dopísaní tretej časti ju pootočíme ešte raz. Takto dostaneme zašifrovaný text v tvare štvorca  $n \times n$  zloženého z písmen. Postup dešifrovania je zrejmý (za predpokladu, že máme rovnakú mriežku a vieme jej prvú polohu).

ÚLOHA: Napíšte program, ktorý pre dané párne  $n$ ,  $4 \leq n \leq 10$ , zašifruje daný text dĺžky  $n^2$  pomocou danej šifrovacej mriežky, ak je mriežka správna, alebo vypíše oznam, že mriežka je nesprávna.

Mriežka môže vstupovať napríklad v tvare matice  $n \times n$  núl a jednotiek (1 je vystrihnutý štvorček, 0 je nevystrihnutý štvorček).

V prípade nejasností o spôsobe šifrovania odporúčame knižku Julesa Verna, Matej Sandor – Nový gróf Monte Christo.

### 311. O spoločnom prvku

[18]

Napište program, ktorý pre dané dvojrozmerné pole  $P$  celých čísel o  $m$  riadkoch a  $n$  stĺpcoch, v ktorom je každý riadok usporiadaný vzostupne, nájde a vypíše všetky čísla, ktoré sa nachádzajú v každom riadku. Ak také číslo neexistuje, program o tom podá správu.

Napríklad pre  $m = 4$ ,  $n = 5$  a dané pole  $P$ :

2	3	5	7	9
4	6	9	11	13
1	3	7	9	10
3	3	4	8	9

je výsledok 9, lebo toto číslo sa nachádza v každom riadku.

### 312. O nahrádzaní

[15]

V jednorozmernom poli  $P$  sa ukladajú skupiny celých čísel takýmto spôsobom:

- každá skupina čísel začína záporným číslom (ďalej toto číslo nazývame hlavička skupiny), ostatné čísla v skupine sú kladné.
- celé pole obsahuje niekoľko skupín (rôzne dlhých), ktoré sú uložené tesne za sebou tak, že sú utriedené podľa hodnôt hlavičky zostupne.
- zvyšok poľa za koncom poslednej skupiny je doplnený nulami.

Príklad takéhoto poľa:

$-10, 2, 3, 5, -20, 4, 6, 7, 8, -25, 8, 6, 5, 4, 0, 0, 0, 0, 0, 0$

ÚLOHA: Napíšte program, ktorý začne s prázdnyim poľom  $P$  (t.j. obsahujúcim samé nuly), postupne číta skupiny čísel (v ľubovoľnom poradí ich hlavičiek) a začleňuje ich do poľa podľa nasledujúcich pravidiel:

- ak má vstupná skupina len hlavičku (t.j. pozostáva len zo záporného čísla), tak:
  - ak má hlavička hodnotu  $-9999$ , tak program končí činnosť
  - inak sa z poľa vyhodí skupina s rovnakou hlavičkou a zvyšok poľa sa príslušne posunie. Ak v poli neexistuje skupina s rovnakou hlavičkou, tak sa neurobí nič.
- ak vstupná skupina pozostáva z viac prvkov než len hlavičky, tak:
  - ak sa už v poli nachádza skupina s rovnakou hlavičkou, tak ju nová skupina nahradí, zvyšok poľa za touto skupinou sa príslušne posunie,
  - ak sa v poli nenachádza skupina s rovnakou hlavičkou, tak sa nová skupina vsunie na miesto, ktoré jej prislúcha podľa hodnoty hlavičky, zvyšok poľa sa zase príslušne posunie,
  - ak by sa pri pridávaní do poľa mal prekročiť jeho koniec, tak sa vypíše chyba a posledne prečítaná skupina sa ignoruje (t.j. pole musí byť v takom stave, v akom bolo pred pokusom o začlenenie do ďalšej skupiny) a pokračuje sa v práci programu.

Program vypisuje na výstup každú vstupnú skupinu a stav poľa  $P$  po jej spracovaní.

Tvar, v akom vstupujú skupiny do programu, nepredpisujeme. Každý si ho môže zvoliť tak, aby vyhovoval použitému programovaciemu jazyku.

Ten, kto pracuje s BASICom, iste spoznal v zložito popísaných pravidlách spôsob, akým pracuje interpreter BASICu pri zápise a opravovaní programu.

Príklad výstupu pre pole  $P$  dĺžky 20.

Vstup: -10,2,3,4,5

pole : -10,2,3,4,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

Vstup: -20,4,5,6

pole : -10,2,3,4,5,-20,4,5,6,0,0,0,0,0,0,0,0,0,0,0

Vstup: -15,6,7

pole : -10,2,3,4,5,-15,6,7,-20,4,5,6,0,0,0,0,0,0,0,0,0

Vstup: -15,8,8,8

pole : -10,2,3,4,5,-15,8,8,8,-20,4,5,6,0,0,0,0,0,0,0,0

Vstup: -10

pole : -15,8,8,8,-20,4,5,6,0,0,0,0,0,0,0,0,0,0,0,0

Vstup: -5,1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,1,2,3,4

Skupina sa nezmestila

pole : -15,8,8,8,-20,4,5,6,0,0,0,0,0,0,0,0,0,0,0,0

Vstup: -9999

Koniec práce programu

pole : -15,8,8,8,-20,4,5,6,0,0,0,0,0,0,0,0,0,0,0,0

Ak si zvolíte dlhšie pole  $P$ , nemusíte vypisovať v medzivýsledkoch nuly na jeho konci.

### 313. O nerozhodných voličoch

10

Predstavte si takéto hlasovanie: Hlasujúci sedia v sále, kde sú stoličky usporiadané vo štvorci  $n \times n$ . Majú hlasovať o niečom, ale ani jeden z nich nie je pevne rozhodnutý, či má hlasovať za alebo proti. Preto sa spýta niektorého zo svojich ôsmich susedov, akého je názoru a stotožní sa s ním. Keďže sú všetci hlasujúci nerozhodní, každý z nich sa takto „pýta susedov“. Každý z hlasujúcich má jeden z názorov 0 až  $m$ .

ÚLOHA: Je daný štvorec  $n \times n$  políčok, kde v každom políčku je napísané číslo 0 až  $m$ .

- Susedmi políčka so súradnicami  $(i, j)$ ,  $1 \leq i, j \leq n$ , nazývame políčka so súradnicami  $(i-1, j-1)$ ,  $(i-1, j)$ ,  $(i-1, j+1)$ ,  $(i, j-1)$ ,  $(i, j+1)$ ,  $(i+1, j-1)$ ,  $(i+1, j)$ ,  $(i+1, j+1)$ . Pritom operácie „+1“ a „-1“ v popise susedov treba chápať tak, že ak je ich výsledok mimo intervalu  $\langle 1, n \rangle$ , tak sa do tohto tvaru upraví vhodným pripočítaním alebo odpočítaním čísla  $n$ . Teda krajné políčka majú susedov na „druhom kraji“, napríklad pre  $n = 10$  má políčko  $(1, 2)$  susedov  $(10, 1)$ ,  $(10, 2)$ ,  $(10, 3)$ ,  $(1, 1)$ ,  $(1, 3)$ ,  $(2, 1)$ ,  $(2, 2)$ ,  $(2, 3)$ , lebo  $1 - 1 = 0$  upravíme do intervalu  $\langle 1, 10 \rangle$  pripočítaním čísla 10.

- v čase  $t = 0$  nadobudnú všetky políčka náhodné hodnoty (0 až  $m$ ).
- v čase  $t + 1$  nadobudne náhodne vybrané políčko takú hodnotu, akú mal jeho niektorý náhodne vybraný sused v čase  $t$ .

Napište program, ktorý pre danú veľkosť  $n$ ,  $m$  a počet krokov simulácie  $k$  simuluje vyššie popísanú situáciu a vypíše na obrazovku stav poľa v čase  $0, 1, \dots, k$ .

Simulácia je zaujímavá pre veľké  $k$  a  $n$  väčšie než 10. Pokúste sa sformulovať hypotézu o chovaní sa voličov.

### 314. O vážení

[20]

Máme daných  $n$  závaží s hmotnosťami  $1, 3, 9, \dots, 3^{n-1}$  jednotiek (napríklad kilogramov), ďalej máme  $k$  dispozícií dvojramenné váhy (mancier) a predmet, ktorý máme odvážiť, pritom máme zaručené, že jeho hmotnosť je celé číslo od 1 do  $\frac{3^n - 1}{2}$ .

ÚLOHA: Napište program, ktorý pre ľubovoľnú hmotnosť  $h$  predmetu (z daného intervalu) zistí, ako treba rozložiť závažia na obe ramená tak, aby vznikla rovnováha.

Napríklad, majme štyri závažia hmotností  $1, 3, 9$  a  $27$  jednotiek. Vašou úlohou je zistiť ako tieto závažia rozložiť na misky váh, aby sme odvážili predmet s hmotnosťou  $5$  jednotiek. Zrejme jediným riešením je na miskú s predmetom dať závažia  $1$  a  $3$  a na druhú miskú dať závažie  $9$ .

Zvoľte výstup z vášho programu podobne ako v uvedenom príklade. Vstupmi do programu budú dve čísla  $n$  - počet závaží,  $n \leq 10$  a  $h$  - hmotnosť predmetu,  $0 \leq h \leq \frac{3^n - 1}{2}$ .

Úloha má pre dané  $n$  a  $h$  (z uvedeného intervalu) vždy jediné riešenie.

### 315. O ekvivalenciách

[20]

Zamyslime sa nad takouto úlohou: vieme, že Jana má toľko súrodencov ako Jožo, Fero má toľko súrodencov ako Katka, Jožo má toľko súrodencov ako Zdeno a Fero má toľko súrodencov ako Jožo. Nás zaujíma, či na základe týchto údajov vieme povedať, či aj Jana a Katka majú rovnaký počet súrodencov.

Keď sa trochu zamyslíte nad uvedenou úlohou (a nakreslíte si situáciu), ľahko zistíte, že všetci majú rovnaký počet súrodencov.

Nech  $R$  je relácia ekvivalencie (t.j. reflexívna, tranzitívna a symetrická binárna relácia). Máme daných  $n$  dvojíc  $(x_1, y_1), \dots, (x_n, y_n)$ , ktoré sú prvkami  $R$ . Okrem toho máme danú dvojicu  $(x, y)$ . Pýtame sa, či na základe predošlých údajov môžeme odvodiť, že aj  $(x, y)$  je prvkom relácie  $R$ .

ÚLOHA: Napište program, ktorý pre dané  $n$ ,  $(x_1, y_1), \dots, (x_n, y_n)$  zistí, či daná dvojica  $(x, y)$  patrí do relácie ekvivalencie, ktorá je daná dvojicami  $(x_1, y_1), \dots, (x_n, y_n)$ .

Prvky  $x, y$  (množina na ktorej je definovaná uvažovaná relácia  $R$ ) môžu byť reťazce (všetky, ktoré sa dajú zobraziť v počítači) alebo prirodzené čísla v rozsahu počítača podľa toho, či je Váš počítač schopný spracovať reťazce.

Napríklad pre dané  $n = 7$ , dvojice  $(1, 2), (5, 4), (6, 4), (6, 6), (7, 8), (2, 3), (8, 4)$  sa pýtame

- je dvojica  $(8, 5)$  prvkom relácie? Odpoveď je áno lebo:
  - ak je  $(5, 4)$  prvkom  $R$ , tak aj  $(4, 5)$  je prvkom  $R$  (vlastnosť symetrie),
  - ak  $(8, 4)$  je prvkom  $R$  a aj  $(4, 5)$  je prvkom  $R$ , tak aj  $(8, 5)$  je prvkom  $R$  (vlastnosť tranzitívnosti).
- je dvojica  $(8, 8)$  prvkom relácie? Odpoveď je áno, lebo  $R$  je reflexívna (t.j. obsahuje všetky dvojice  $(x, x)$ ).

### 321. O Pytagorejských trojuholníkoch

[M18]

Usporiadaná trojica prirodzených čísiel  $(a, b, c)$  sa nazýva Pytagorejská<sup>2</sup>, ak  $a^2 + b^2 = c^2$ ,  $a \leq b$ . Strana dĺžky  $c$  je prepona,  $a$  je kratšia,  $b$  dlhšia odvesna.

ÚLOHA: Napište program, ktorý vypíše prvých  $n$  Pytagorejských trojíc, usporiadaných zostupne podľa prepony a pre rovnaké prepony, podľa kratšej odvesny.

<sup>2</sup> Pytagoras asi 580 – asi 500 p.n.l., grécky matematik a filozof

**322. O \*-postupnostiach**

[15]

Postupnosť  $a_1 a_2 \dots a_n$  zloženú z núl a jedničiek (t.j. každé  $a_i$  je nula alebo jednička) nazveme 01-postupnosťou. Postupnosť  $b_1 b_2 \dots b_m$  zloženú z núl, jedničiek a hviezdíčiek nazveme \*-postupnosťou. 01-postupnosť  $a_1 \dots a_n$  je odvodená z \*-postupnosti  $b_1 \dots b_m$  ak:

- a)  $m = n$
- b) a zároveň pre všetky  $i = 1, \dots, n$  platí: ak  $a_i \neq b_i$  tak  $b_i = *$ .

Napríklad 01-postupnosť 110101 je odvodená z \*-postupnosti  $*101*1$ , ale nie z postupnosti  $11*10$ , lebo má inú dĺžku, ani z postupnosti  $***100$ , lebo na poslednom mieste sa líšia, pričom v druhej postupnosti nie je posledná hviezdíčka.

ÚLOHA: Napíšte program, ktorý pre danú \*-postupnosť dĺžky  $n$ ,  $1 \leq n \leq 10$  vypíše všetky 01-postupnosti z nej odvodené.

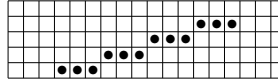
**323. O interpolátore**

[M30]

Každý osobný mikropočítač, ktorý má bodovú grafiku (napríklad PMD-85, PP 01, ZX-Spectrum, IBM PC a pod.) musí vedieť vykresľovať na svojej obrazovke úsečky. Úlohu vykresliť úsečku v bodovom rastru má väčšinou podprogram, ktorý sa nazýva interpolátor. Vašou úlohou bude navrhnúť algoritmus interpolátora. Znakové pole  $D$  rozmeru  $m \times n$  predstavuje grafickú bodovú obrazovku. Každý jej prvok obsahuje informáciu o jase jedného bodu obrazovky.  $X$  znamená rozsvietený bod a  $.$  (bod) znamená zhasnutý bod.

ÚLOHA: Napíšte program, ktorý začne pracovať s prázdnu obrazovkou (t.j. v poli  $D$  sú samé bodky) a pre dané dva body  $[x_1, y_1]$  a  $[x_2, y_2]$ , kde  $1 \leq x_1 \leq m$ ,  $1 \leq y_1 \leq n$ ,  $1 \leq x_2 \leq m$ ,  $1 \leq y_2 \leq n$ , vykreslí v poli  $D$  úsečku z bodu so súradnicami  $[x_1, y_1]$  do bodu so súradnicami  $[x_2, y_2]$  a vypíše obsah poľa  $D$ .

Pod vykreslením rozumieme to, že niektorým prvkom matice priradíme hodnotu  $X$  tak, aby po vypísaní poľa na papier vznikol dojem úsečky, ktorá spája dva dané body.



$m$  a  $n$  zvolte podľa možností Vášho počítača. Odporúčame skúšať program pre maticu aspoň  $50 \times 50$ . Vypisovanie znakov  $X$  a  $.$  sa dá nahradiť priamo zmenou jasu príslušných bodov (pixelov) na obrazovke, ktorá bude predstavovať pole bodov  $m \times n$ .

V reálnych mikropočítačoch veľmi záleží na rýchlosti interpolátora, aby bol použiteľný aj na vytváranie pohybového dojmu (animácie), preto sa dobre zamyslite nad počtom operácií, ktoré je nevyhnutné vykonať na vykreslenie čiary.

**324. O šachovom koňovi.**

[20]

Máme danú šachovnicu rozmeru  $m \times n$ ,  $m, n \leq 10$ . Na nej je daná počiatočná pozícia šachového koňa  $x, y$ ,  $1 \leq x \leq m$ ,  $1 \leq y \leq n$ . Každému políčku šachovnice so súradnicami  $i, j$  priradíme číslo  $T(i, j, x, y)$ , ktoré určuje minimálny počet ťahov, na ktoré sa môže dostať šachový kôň z políčka  $x, y$  na políčko  $i, j$ . Ak prechod medzi týmito dvoma políčkami nie je možný, tak bude príslušná hodnota  $-10$ .

ÚLOHA: Napíšte program, ktorý pre dané  $m, n, x$  a  $y$  vypíše  $T(i, j, x, y)$  pre všetky  $i, j = 1, \dots, n$  usporiadané do obdĺžnika  $m \times n$ .

**325. O Hammingovej postupnosti**

[20]

Hammingova<sup>3</sup> postupnosť je vzostupne usporiadaná postupnosť čísiel, ktoré nie sú deliteľné inými prvočíslami, ako 2, 3 a 5. Postupnosť začína takto 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, ...

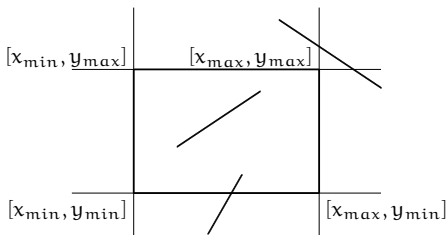
ÚLOHA: Napíšte program, ktorý pre dané  $n$  vypíše prvých  $n$  členov hammingovej postupnosti.

<sup>3</sup> Richard W. Hamming (1915–1998), americký matematik

**331. O grafickom okienku**

20

V rovine nech je dané obdĺžnikové okno (obrázok), ktoré zobrazuje dvojrozmerný interval  $\langle x_{\min}, x_{\max} \rangle \times \langle y_{\min}, y_{\max} \rangle$ , teda obdĺžnik s vrcholmi  $[x_{\min}, y_{\min}]$ ,  $[x_{\min}, y_{\max}]$ ,  $[x_{\max}, y_{\min}]$ ,  $[x_{\max}, y_{\max}]$ . Toto okno nám predstavuje zobrazovaciu plochu (tzv. vykresľovací rozsah) nejakého grafického zapisovača. My chceme vykresliť úsečku s koncovými bodmi  $[x_1, y_1]$ ,  $[x_2, y_2]$ , ktorá nemusí celá ležať v zobrazovacom obdĺžniku.



Môžu nastať tieto prípady:

- celá úsečka leží v obdĺžniku, potom ju jednoducho vykreslíme;
- z úsečky len „nejaký kúsok“ leží v obdĺžniku, potom vykreslíme len tento kúsok;
- žiadna časť úsečky neleží v obdĺžniku, potom nekreslíme nič. V druhom prípade „nejaký kúsok“ môže byť úsečka alebo bod.

ÚLOHA: Napíšte program, ktorý ako vstupné údaje dostane súradnice zobrazovacej plochy  $x_{\min}, x_{\max}, y_{\min}$  a  $y_{\max}$ . Potom program prečíta súradnice koncových bodov úsečky  $x_1, y_1$  a  $x_2, y_2$ . Pre každú danú úsečku program vypíše súradnice začiatku a konca skutočne kreslenej časti úsečky. Jeden bod chápeme ako úsečku s rovnakými koncovými bodmi. Medzi výsledkami pošlite príklady na všetky možné varianty polohy úsečky k obdĺžnikovému oknu.

**332. O obyvateľoch ostrova**

10

Ostrov AXA obýva  $N$  domorodcov. Od neho na 5 dní plavby na kanoe sa nachádza opustený ostrov YBY. Oba ostrovy sú bohaté na rôzne druhy ovocia, takže sú lákadlom pre domorodcov. Hneď po objavení ostrova YBY sa niekoľkí domorodci rozhodli presťahovať. Miestny matematik vypočítal, že s pravdepodobnosťou  $p$ ,  $0 < p < 1$  každý domorodec opúšťa buď ostrov AXA, alebo YBY (t.j. domorodec opúšťa aj ostrov YBY s pravdepodobnosťou  $p$ ). Domorodcom je známe, že po  $d$  dňoch začína obdobie dažďov a teda po týchto  $d$  dňoch už žiaden nie je na ceste (plavbe).

ÚLOHA: Napíšte program, ktorý modeluje sťahovanie domorodcov pre vstupné hodnoty  $n, p$  a  $d$  a vypíše počty obyvateľov na ostrovoch po  $d$  dňoch (po  $d-5$  dní už žiaden nezačne plavbu). Uvažujte, že domorodci sa rozhodnú, či majú vyplávať, vždy ráno. Skúste pre  $n = 100, p = 0.1$  a  $d = 200$ .

**333. O deliteľoch**

23

Napíšte algoritmus, ktorý pre dané prirodzené číslo  $n$  z intervalu  $\langle 1, 1000 \rangle$  nájde a vypíše najmenšie také prirodzené číslo, ktoré je deliteľné všetkými prirodzenými číslami menšími alebo rovnými ako  $n$ . Skúste pre  $n = 10, 100$  a  $200$ .

Už pre  $n = 20$  dostávame také veľké číslo, ktoré aritmetika počítača zaokrúhli. Preto vymyslite taký spôsob výpočtu hľadaného čísla, ktorý umožní určiť všetky jeho cifry aj v prípade, že je väčšie než dovoľuje aritmetika počítača.

**334. O zlomkoch (Fareyov<sup>4</sup> rad)**

M50

Napíšte program, ktorý pre dané prirodzené číslo  $n$  vypíše rastúcu postupnosť všetkých pravých zlomkov z intervalu  $\langle 0, 1 \rangle$ , ktorých menovateľ nie je väčší ako  $n$ . Pravý zlomok je taký, ktorého čitateľ aj menovateľ sú nesúdeliteľné.

<sup>4</sup> John Farrey (1766–1826), anglický geológ. V roku 1816 publikoval niektoré vlastnosti uvedeného radu.



Pre  $n = 7$  je postupnosť zlomkov:

$$\frac{1}{7}, \frac{1}{6}, \frac{1}{5}, \frac{1}{4}, \frac{2}{7}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{3}{7}, \frac{2}{3}, \frac{4}{5}, \frac{3}{7}, \frac{5}{6}, \frac{4}{7}, \frac{6}{7}, \frac{1}{7}.$$

### 335. O aritmetických výrazoch

[23]

Bezzátvorkový výraz budeme nazývať taký aritmetický výraz, ktorý obsahuje len jednociferné čísla a znamienka '+' alebo '-'. Ozátvorkovaním bezzátvorkového výrazu nazveme niektoré rozmiestnenie párov zátvoriek '(' a ')' tak, aby vznikol dobre uzátvorkovaný výraz. Vyhodnotenie výrazu je číslo, ktoré vznikne aplikovaním operátorov '+' a '-' na operandy podľa bežných zvyklostí s prihliadnutím na zátvorky.

Bezzátvorkový výraz  $1+2-3+4-5$  má napríklad takéto ozátvorkovania:  $1+2-(3+4)-5$ ,  $1+(2-(3+4-5))$ . Vyhodnotením prvého výrazu je  $-9$ , vyhodnotením druhého výrazu je  $1$ .

ÚLOHA: Napíšte program, ktorý po prečítaní bezzátvorkového výrazu v tvare znakového reťazca vytlačí všetky možné vyhodnotenia, ktoré môžeme dosiahnuť rôznym ozátvorkovaním daného výrazu. (Jedno vyhodnotenie sa môže vo výsledkoch vyskytnúť aj viackrát, žiadne však nesmie chýbať.) Ku každému vyhodnoteniu program vypíše aj ozátvorkovaný výraz, ktorého je to hodnota. Bezzátvorkový výraz  $1-2+1$  má ozátvorkovania:  $1-2+1$ ,  $(1-2)+1$ ,  $(1-2+1)$ ,  $1-(2+1)$ ,  $(1-(2+1))$ ,  $((1-2)+1)$ ; ich vyhodnotenia sú  $0$ ,  $0$ ,  $0$ ,  $-2$ ,  $-2$ ,  $0$ . Program teda môže vypísať len čísla  $0$ ,  $-2$ , ale môže vypísať aj napríklad  $0$ ,  $0$ ,  $0$ ,  $0$ ,  $-2$ ,  $-2$ ,  $0$ ,  $0$ ,  $0$  (a k nim zodpovedajúce výrazy).

### 411. Číslo

[20]

Napíšte algoritmus a program, ktorý efektívne nájde všetky prirodzené čísla  $n$  také, že najvyššia cifra čísla  $n$  je počet cifier  $0$  v čísle  $n$ , druhá najvyššia cifra čísla  $n$  je počet cifier  $1$  v čísle  $n$ , tretia najvyššia cifra čísla  $n$  je počet cifier  $2$  v čísle  $n$ , atď. až desiatu najvyššia cifra čísla  $n$  je počet cifier  $9$  v čísle  $n$ . Najvyššia cifra je cifra pri najvyššej mocnine  $10$ .

Príklad:  $n = 1210$ .

### 412. Kódovanie kombinácií

[20]

Nech  $n$ ,  $k$  sú prirodzené čísla. Uvažujeme lexikograficky zoradené všetky  $k$ -prvkové podmnožiny množiny  $\{1, 2, \dots, n\}$ . Ich počet je  $\binom{n}{k}$ . Napíšte algoritmus a program, ktorý každej  $k$ -prvkovej podmnožine množiny  $\{1, 2, \dots, n\}$  z uvedeného lexikografického poradia priradí „poradové číslo“, t.j. číslo z množiny  $\{1, 2, \dots, \binom{n}{k}\}$ . Dôraz kladte na efektívnosť algoritmu!

### 413. Konvexný obal

[40]

V rovine je daných  $n$  bodov popísaných súradnicami  $[x_i, y_i]$ ,  $1 \leq i \leq n$ . Napíšte program, ktorý vypíše všetky body, ktoré tvoria vrcholy konvexného obalu bodov  $[x_i, y_i]$ ,  $1 \leq i \leq n$ . Konvexný obal množiny  $A$  je najmenšia konvexná množina obsahujúca  $A$ .

### 414. Josifova úloha

[40]

Počas židovskej vojny, keď Rimania obsadili Jotopatu, podarilo sa Josifovi spolu so štyridsiatimi vojakmi ujsť a schovať sa v jaskyni. Josif zistil, že všetci vojaci – okrem neho a ešte jedného človeka – radšej spáchajú samovraždu, akoby padli do zajatia. Báľ sa vystúpiť proti takému riešeniu, preto navrhol vykonať samovraždy organizovane: Všetci sa postavia do kruhu a počnúc od niektorého sa každý tretí zavraždí až po posledného. Potom Josif akoby náhodou postavil seba a toho druhého na 31. a 16. miesto od začiatku. Toľko hovorí história. A teraz formálnejšie.

ÚLOHA: V kruhu je  $n$  ľudí. Ľudia sú na začiatku očíslovaní od  $1$  po  $n$  postupne. Počnúc od prvého ideme jedným smerom a z kruhu vylučujeme každého  $m$ -tého (ďalej sa nepočíta) až pokiaľ nevyhlúčime posledného.

Napište program, ktorý pre dané  $n$  a  $m$  zistí o každom človeku kedy-kol'ký vypadol z kruhu. Kruh tvorí aj jeden, alebo dvaja ľudia.

#### 415. Úloha na vyhľadávanie

[20]

Dané je pole  $A$ , ktoré obsahuje  $n$  čísel usporiadaných vzostupne a číslo  $x$ , pre ktoré platí  $A[1] < x < A[n]$ .

ÚLOHA: Napište program, ktorý nájde index  $i$  v poli  $A$  taký, že  $A[i-1] < x \leq A[i]$ .

Program nesmie používať žiadne pomocné pole a mal by minimalizovať počet prezretých prvkov poľa  $A$ . (Napríklad ak  $A[n-1] < x$ , tak by nemal prezerať všetkých  $n-1$  prvkov.)

#### 421. O mnohouholníku

[25]

Napište program a algoritmus, ktorý určí, či daná uzavretá lomená čiara tvorí konvexný  $n$ -uholník.

Vstupom do programu je číslo  $n$  a  $n$  dvojíc reálnych čísel  $[x_i, y_i]$ , kde  $1 \leq i \leq n$ , ktoré predstavujú po sebe idúce vrcholy lomenej čiary.

#### 422. O Buffonovej<sup>5</sup> ihle

[10]

Na „nekonečnej“ rovnej ploche (rovine) je nakreslený nekonečný systém rovnobežných priamok, pričom vzdialenosť susedných priamok je  $h$ . Na túto rovinu hádzeme ihlu dĺžky  $h$ . Zisťujeme pravdepodobnosť (presnejšie relatívnu početnosť) javu, že dopadnutá ihla pretína nejakú priamku (alebo na nej leží) z uvedeného systému. Napište program, ktorý bude simulovať popísaný pokus. Porozmýšľajte akým spôsobom súvisí pravdepodobnosť daného javu s číslom  $\pi = 3.1415926 \dots$

Relatívna početnosť nejakého javu pri simulácii je pomer počtu úspešných výsledkov (t.j. kedy jav nastal) k počtu všetkých výsledkov (t.j. k počtu pokusov).

#### 423. O šachovnici

[20]

Je dané číslo  $n$ . Napište algoritmus a program, ktorý zafarbí šachovnicu  $n \times n$  pomocou  $n$  navzájom rôznych farieb, tak aby v každom riadku a stĺpci bolo všetkých  $n$  farieb. Program vypíše všetky možné zafarbenia šachovnice.

Riešte napríklad pre  $n = 3, 4$ .

#### 424. O reálnom čísle

[18]

Napište program, ktorý zo vstupu načíta znakovú reprezentáciu reálneho čísla a vypíše hodnotu tohoto čísla. Predpokladajte, že číslo bude z rozsahu reálnych čísel vo Vami použitom programovacom jazyku. Znaková reprezentácia reálneho čísla má vo všeobecnosti tento tvar (je podobná ako v Basicu alebo Pascale):

- môže začínať unárnym  $+$  alebo  $-$ ;
- časť pred desatinnou bodkou musí obsahovať aspoň jednu cifru;
- ak číslo obsahuje desatinnú bodku, desatinná časť musí obsahovať aspoň jednu cifru;
- reálne číslo môže byť napísané v semilogaritmickom tvare, t.j. za číslom nasleduje znak  $E$  a maximálne dvojciferné celé číslo (môže mať znamienko  $+$  alebo  $-$ ), ktoré označuje exponent.

Príklady reálnych čísel:  $0.23$ ,  $+57.9E3$ ,  $-5E-5$

#### 425. O výmenách

[20]

Program dostáva na vstupe prirodzené číslo  $n$ ,  $r$  také, že  $r|n$  a pole  $A$  s indexami  $0, \dots, n-1$ , ktorého prvkami sú čísla  $0$  až  $n-r$  s týmito vlastnosťami:

- (i) čísel  $0$  je v poli  $A$  presne  $r$ ;

<sup>5</sup> G. L. Buffon, 1707–1788, bol francúzsky prírodovedec

- (ii) každé z čísel 1 až  $n - r$  sa nachádza v poli práve raz a to tak, že pre každé  $0 \leq i < j \leq n - 1$ : ak  $A[i] \neq 0$  a  $A[j] \neq 0$ , tak  $A[i] < A[j]$ , t.j tak, že ak by sme z poľa odstránili nulové prvky, pole  $A$  by bolo vzostupne usporiadané).

ÚLOHA: Napište algoritmus, ktorý preusporiada pole  $A$  tak, že nulové prvky majú indexy násobkov  $n/r$  (čiže  $0, n/r, (2n)/r, \dots$ ) a pritom platí podmienka (ii) pre pole  $A$ . Pri preusporiadavaní poľa  $A$  môžete použiť len vzájomnú výmenu dvoch prvkov. Váš program bude vypisovať dvojice prvkov, nad ktorými sa vykonáva operácia výmeny.

#### 431. O počte výskytov

20

Je dané prirodzené číslo  $n$  a  $n$ -prvkové pole celých čísel  $A$ .

Napište program, ktorý nájde v poli  $A$  taký prvok, ktorý sa v ňom vyskytuje najviacrát (ak je takých prvkov viac, tak niektorý z nich).

Napríklad pre  $n = 5$  sa v poli  $A = (1, 2, 3, 2, 3)$  najviackrát vyskytuje číslo 2 (alebo číslo 3).

#### 432. O pokazenom obracači

18

Definujme si špeciálne triediace zariadenie, ktoré nazveme obracač. Obracač sa skladá z pozícií. Na každej pozícii sa nachádza práve jedna kartička, na ktorej je napísané prirodzené číslo  $\leq n$ . Pritom každé z čísel 1 až  $n$  sa nachádza práve na jednej kartičke. Úlohou obracača je kartičky utriediť vzostupne podľa čísel na nich napísaných čísel. Obracač môže triediť len pomocou operácií  $obrat(i, j)$ , kde  $i \leq j$ . Táto operácia realizuje zrkadlové otočenie poradia kartičiek od  $i$ -tej po  $j$ -tu. Teda ak si  $i$ -tu kartičku označíme  $k_i$  a znakom  $:=$  označíme výmenu dvoch kartičiek, tak môžeme operáciu  $obrat(i, j)$  rozpísať takto:

$$\begin{aligned} k_i &:= k_j \\ k_{i+1} &:= k_{j-1} \\ k_{i+2} &:= k_{j-2} \\ &\vdots \\ k_{i+((j-i)\text{div}2)} &:= k_{j-((j-i)\text{div}2)} \end{aligned}$$

Predpokladajme, že máme k dispozícii obracač s malou chybičkou – má „zaseknutú“  $k$ -tu pozíciu. To znamená, že dokáže vykonať len také operácie  $obrat$ , ktoré nepohnú, či nemusia pohnúť kartičkou na  $k$ -tej pozícii. Zdá sa však, že aj takýto obracač dokáže utriediť kartičky, pravda len za predpokladu, že  $k$ -ta kartička je už od začiatku na svojom mieste.

ÚLOHA: Napište program, ktorý najprv prečíta prirodzené čísla  $n$  a  $k \leq n$  a  $n$ -prvkové pole  $A$ , ktoré obsahuje každé z prirodzených čísel od 1 do  $n$  práve raz a zároveň platí  $A[k] = k$ . Čísla  $n$  a  $k$  definujú  $n$ -prvkový obracač so zaseknutou  $k$ -tou pozíciou a pole  $A$  určuje počiatočné rozmiestnenie kartičiek v obracači.

Výsledkom práce Vášho programu nech je postupnosť takých operácií  $obrat$ , aby po ich realizácii obracačom (so zaseknutou  $k$ -tou pozíciou) boli kartičky vzostupne utriedené. (Vášou úlohou je teda napísať program, ktorý riadi obracač). Ak taká postupnosť neexistuje, program o tom podá správu.

Predpokladáme, že obracač pracuje oveľa pomalšie ako počítač, ktorý realizuje Váš program, preto je treba minimalizovať v prvom rade počet operácií  $obrat$ , ktoré Váš program predpíše obracaču.

Pre  $n = 9$ ,  $k = 4$  a pole  $A = (2, 3, 8, 4, 6, 5, 1, 7, 9)$  je správny (nie však najlepší) výsledok činnosti programu, napríklad:

```
obrat(6,7); obrat(5,6); obrat(3,5); obrat(1,3); obrat(2,3);
obrat(5,6); obrat(6,7); obrat(7,8); obrat(5,6).
```

**433. O parketách**

[18]

Na štvorčekovom papieri máme nakreslený pôdorys miestnosti. Predpokladajme, že miestnosť sa skladá z nejakého počtu celých štvorčekov, z nich každý sa dotýka ďalších štvorčekov izby aspoň jednou stranou. Túto miestnosť máme „vyparketovať“ obdĺžnikmi veľkosti  $1 \times 2$  štvorčeky.

Napište program, ktorý prečíta zo vstupu popis miestnosti a vytlačí jej pokrytie obdĺžnikmi  $1 \times 2$ , alebo správu o tom, že pokrytie nie je možné.

Tvar miestnosti odporúčame kódovať tak, že ju ohraničíme nejakým obdĺžnikom, zadáme veľkosť obdĺžnika (počet riadkov a stĺpcov) a dvojrozmerné pole zložené z núl a jednotiek, kde jednotka predstavuje štvorček patriaci do miestnosti. Výstup požadujeme v tvare matice čísiel, kde nula označuje štvorček, ktorý neprislúcha miestnosti, a číslo  $i$ ,  $2 \leq i < k$  ( $k$  je počet použitých „parkiet“) označuje jednoznačne  $(i-1)$ -vú použitú parketu.

Príklad:

Vstup: rozмеры:4,5	miestnosť	01100	Výstup:	02200
		11110		33440
		11111		55677
		11100		88600

**434. O symetrických číslach**

[18]

Napište program, ktorý pre dané  $k$  vypíše všetky nepárne čísla  $< 2^k$ , ktorých zápis v dvojkovej sústave je symetrický podľa stredu. Dvojkový zápis čísla 93  $(1011101)_2$  je symetrický podľa stredu, zápis čísla 67  $(100011)_2$  nie je symetrický. Pre  $k = 4$  program vypíše 1, 3, 5, 7, 9, 15. Program sa snažte urobiť čo najrýchlejší.

**435. O výpise reálneho čísla**

[18]

Pre dané reálne číslo  $r$  a prirodzené čísla  $n$  a  $m$ , kde  $m \leq n - 2$  definujeme zápis reálneho čísla  $r$  v pevnom formáte dĺžky  $n$  o  $m$  desatinných miestach (skrátene  $F_n.m$  zápis čísla  $r$ ) nasledovne.

Nech  $r$  je číslo.  $F_n.m$ -zápis čísla  $r$  je

- má práve  $m$  cifier za desatinnou bodkou,
- posledná cifra je správne zaokrúhlená,
- ak je číslo záporné, tak je pred ním znamienko  $-$ ,
- Počet všetkých znakov zápisu je  $n$ , pričom prípadné medzery sú doplnené z ľava.

Napríklad:

- $F_{10.2}$ -zápis čísla 1236.758 je `0001236.76`
- $F_{7.2}$ -zápis čísla  $-12.1$  je `-12.10`
- $F_{5.2}$ -zápis čísla 125.1 neexistuje.

ÚLOHA: Napište program, ktorý pre dané  $r$ ,  $n$  a  $m$  (pritom  $r$  program prečíta do reálnej číselnej premennej) vytvorí  $F_n.m$ -zápis čísla  $r$  do reťazcovej premennej  $z$  a výstupnú premennú  $Z$  vypíše. Ak pre dané  $n$  a  $m$  neexistuje  $F_n.m$ -zápis čísla  $r$ , tak program uloží do  $Z$  (a vytlačí)  $n$  hviezdíčiek.

V programe nie je povolené používať žiadnu štandardnú funkciu, ktorá prevádza číslo do jeho znakového zobrazenia (vo väčšine BASIC-ov je to funkcia `STR$`). Pravdaže, je dovolené používať konverzné funkcie medzi znakmi a ich kódmi (`CHR$`, `ASC`, `CODE`, `CHR`, `ORD` a pod.).

**511. Zarovnávanie textu.**

► [21]

Súčasťou systémov na spracovanie textu je vždy program, nazývaný formátovač, ktorého úlohou je zarovnať odseky textu tak, aby mali rovnaký pravý okraj. Program číta vstupné riadky rôznej dĺžky a skráti ich, alebo predĺži čo najbližšie k určenej šírke riadku, a potom vsunie medzi niektoré slová medzery tak, aby mal riadok presnú dĺžku.

ÚLOHA: Napište program, ktorý najprv prečíta čísla  $l, p$  a  $z$ . Potom prečíta text pozostávajúci z ľubovoľného počtu riadkov ľubovoľnej (zhora ohraničenej) dĺžky. Koniec každého odseku je v texte označený znakom @, každý odsek začína na novom riadku.

Program vypíše ten istý text upravený do riadkov, ktoré začínajú na  $l$ -tej a končia na  $p$ -tej pozícii v riadku. Prvý riadok každého odseku je odsunutý na  $l+z$ -tú pozíciu. Všetky riadky sú doplnené medzerami tak, aby mali rovnaký pravý okraj.

Príklad: Tento text je zarovnaný pomocou formátovača  
pre  $p=60$ ,  $l=1$  a  $z=5$ , pre lepšiu predstavu uvádzame tento  
odsek v takom tvare, v akom ho prečítal náš program:

Príklad: Tento text je zarovnaný  
pomocou formátovača  
pre  $p=60$ ,  $l=1$  a  $z=5$ , pre lepšiu predstavu  
uvádzame tento odsek v takom tvare, v akom  
ho prečítal náš program:@

## 512. O škrtaní

M45

Nech  $i$  je kladné číslo a  $P$  je nekonečná postupnosť prirodzených čísel:  $1, 2, 3, \dots$

Definujeme operáciu  $\text{Škrt}ni(i, P)$ , ktorá upraví postupnosť  $P$  nasledovne: prvých  $i$  členov ponechá, ďalších  $i$  členov vyškrtnie, potom zase  $i$  členov ponechá,  $i$  vyškrtnie atď. Toto pokračuje nekonečne dlho. Výsledkom operácie je takto vyškrtnaná nekonečná postupnosť.

Postupnosť prirodzených čísiel  $Q$  je definovaná nasledovne:

- $Q_0$  je vzostupne usporiadaná postupnosť zložená zo všetkých prirodzených čísiel.
- $Q_1$  je výsledkom operácie  $\text{Škrt}ni(1, Q_0)$ ;
- $Q_2$  je výsledkom operácie  $\text{Škrt}ni(2, Q_1)$ , atď. ...
- $Q_i$  je výsledkom operácie  $\text{Škrt}ni(i, Q_{i-1})$ , ...

Tento proces sa opakuje donekonečna. Výsledná postupnosť nech je  $Q$ .

Hoci je takto popísaný proces nekonečný (teda nekončí v  $i$ -tom kroku), vieme napriek tomu v konečnom čase určiť prvých  $k$  členov postupnosti  $Q$  pre ľubovoľné  $k$ .

ÚLOHA: Napište program, ktorý pre dané  $k$  vypíše prvých  $k$  členov postupnosti  $Q$ .

Výsledok pre  $k = 5$  je 1, 3, 9, 25, 57. Program považujeme za veľmi dobrý, ak dokáže vypísať v rozumnom čase (menej ako 8 hodín) prvých 15 členov postupnosti.

## 513. Turnaj rytierov

10

V krajine Stochastika sa konajú rytierske turnaje takýmto spôsobom:  $n$  rytierov sa zoradí. Začína najslabší rytier a pokúsi sa zhodiť zo sedla druhého rytiera. Ak sa mu to nepodari, tak pokračuje druhý rytier a pokúsi sa zhodiť zo sedla tretieho. Ak sa prvému podarilo druhého zhodiť, tak druhý vypadáva z turnaja a pokračuje tretí rytier.

Takto turnaj pokračuje, pričom každý rytier, ktorý ešte nevypadol z turnaja, sa pokúša zhodiť zo sedla nasledujúceho rytiera. Nasledujúci rytier  $k$   $n$ -tému rytierovi je prvý rytier. Turnaj končí, keď zostane len jeden rytier, ktorého princezná (dosiaľ slobodná) vyhlási za víťaza.

Pravdepodobnosť, že prvý rytier zhodí zo sedla ľubovoľného protivníka je  $p_1$ , pravdepodobnosť, že druhý rytier zhodí protivníka je  $p_2$  atď. Pritom platí  $p_1 \leq p_2 \leq \dots \leq p_n$ .

ÚLOHA: Napište program, ktorý pre dané  $n$  a  $p_1, \dots, p_n$  nasimuluje turnaj a určí víťaza. Počas simulácie sa vypisuje protokol podľa nasledujúceho vzoru (pre  $n = 4$  môže mať turnaj napríklad takýto priebeh):

```
1.rytier nezhodil
2.rytier nezhodil
3.rytier zhodil 4.rytier
1.rytier nezhodil
2.rytier zhodil 3.rytier
```

```

1.rytier nezhodil
2.rytier zhodil 1.rytier
Zvíťazil 2.rytier.

```

Pozmeňte program tak, aby simuloval 1000 krát turnaj pre  $n = 10$  a  $p_1 = 0.25, p_2 = 0.3, p_3 = 0.35, \dots, p_{10} = 0.7$ . Počas simulácie sa, pravdaže, nevypisuje žiadny protokol a na konci sa vypíše tabuľka počtu víťazstiev každého z rytierov.

#### 514. Plocha kruhu 18

Na štvorčekovom papieri je nakreslená kružnica s celočíselným polomerom  $r, 1 < r \leq 100$ , a so stredom v niektorom mrežovom bode. Dĺžka strany štvorčeka je 1.

ÚLOHA: Napíšte program, ktorý pre daný celočíselný polomer  $r$  vypíše počet štvorčekov, ktoré ležia celé vo vnútri kružnice.

#### 515. Súvislé obrazce 22

Z listu štvorčekového papiera rozmerov  $m \times n$  štvorčekov sme vystrihli niektoré štvorčeky. Zaujímá nás na koľko častí sa papier rozpadne. List papiera je reprezentovaný poľom  $A$ , kde sú vystrihnuté štvorčeky označené jedničkami, ostatné štvorčeky sú označené nulami.

ÚLOHA: Napíšte program, ktorý pre dané prirodzené  $m$  a  $n$  dané dvojrozmerné pole  $A$  vypíše počet častí, na ktoré sa papier rozpadne.

#### 521. O vranách 20

Napíšte program, ktorý pre dané prirodzené číslo  $k, 0 < k < 32768$ , vytlačí frázu NAD KOMINOM LETELO  $k$  VRAN. slovami, zodpovedajúcu gramatike slovenského jazyka t.j. za  $k$  dosadí slovnú formuláciu a vetu upraví.

Príklad:

$k = 23$

NAD KOMINOM LETELO DVADSAT TRI VRAN.

$k = 1$

NAD KOMINOM LETELA JEDNA VRANA.

#### 522. O smetiach 25

Napíšte program, ktorý pre danú postupnosť  $n$  celých navzájom rôznych čísel vypíše novú postupnosť, ktorá vznikne z pôvodnej vyčiarknutím minimálneho počtu čísel tak, aby nová postupnosť bola rastúca.

Napríklad pre  $n = 7$  a postupnosť 1, 7, 5, 8, 2, 3, 4 je nová postupnosť 1, 2, 3, 4

#### 523. O najkratších cestách 27

Pole  $A$  veľkosti  $n \times n$  reprezentuje jednosmerné (!) cesty medzi  $n$  mestami nasledujúcim spôsobom: Ak je v poli  $A$  prvok

- $A[i, j] = 0$ , tak z mesta  $i$  nevedie priama jednosmerná cesta do mesta  $j$ . (Naopak to nemusí platiť).
- $A[i, j] = b$ , kde  $b > 0$ , tak existuje jednosmerná cesta vedúca priamo z mesta  $i$  do mesta  $j$  a má dĺžku  $b$ . (Naopak to nemusí platiť).

ÚLOHA: Napíšte program, ktorý pre dané  $n$ , cesty (pole  $A$ ) a jedno mesto  $u$  určí pre každé mesto  $v$  číslo  $c_v$ , čo je dĺžka najkratšej cesty z  $u$  do  $v$ . Ak neexistuje žiadne spojenie medzi mestami  $u$  a  $v$ , tak  $c_v = 0$ .

Cesta môže viesť len po jednosmerných cestách spájajúcich mestá. Riešte pre  $A[1, 2] = 2$ ;  $A[1, 4] = 3$ ;  $A[2, 3] = 7$ ;  $A[2, 4] = 1$ ;  $A[3, 6] = 1$ ;  $A[4, 5] = 2$ ;  $A[5, 1] = 4$ ;  $A[5, 3] = 1$ ;  $A[6, 1] = 1$ ;  $A[6, 2] = 2$ ; ostatné prvky sú nulové. Za  $u$  zvolte postupne všetky mestá 1 až 6.

## 524. O mape

19

Mapa súostrovia je zakódovaná v celočíselnom poli  $M$  veľkosti  $n \times n$ . Nulové prvky predstavujú more. Súvislé oblasti nenulových prvkov predstavujú ostrovy, pričom pre tieto ostrovy platí:

- i) dva nenulové prvky patria k jednému ostrovu ak sa líšia práve v jednom indexe o jedničku.
- ii) ak pre  $j < k$ ,  $M[i, j]$  a  $M[i, k]$  patria k jednému ostrovu, tak musia byť všetky prvky  $M[i, x]$  nenulové pre  $j < x < k$ , t.j. tiež patria k tomuto ostrovu.
- iii) nenulové číslo vyjadruje nadmorskú výšku príslušného miesta ostrova.
- iv) okraj mapy je more, t.j. prvky poľa  $M$  s indexom 1 alebo  $n$  majú hodnotu 0.

ÚLOHA: Napíšte program, ktorý zistí plochu (t.j. počet nenulových prvkov) ostrova s najvyšším kopcom v rámci danej mapy. Ak ich je viac, tak ľubovoľný z nich.

Názorný príklad: (nuly sú nahradené bodkami)

```

. . . . .
. . . 1 2 1 . . . . . 1 1 4 2 . . .
. . 3 1 1 2 3 4 . . 5 . . 1 2 4 4 4 . .
. 1 1 1 1 1 1 1 . . 4 . . 2 3 . . . .
. . 4 5 6 1 2 . . . . .
. . . . 2 2 . . . . . 2 4 4 5 2 . .
. . . . . . . . . 2 6 3 6 7 3 1 .
. . . . . . . . . 4 3 1 1 3 5 2 .
. . . . . . . 2 4 1 . 3 5 8 9 4 5 .
. . . . . . . . 3 3 . . 3 3 7 . . .
. . . . .

```

Na tejto mape je päť ostrovov a najvyšší (vpravo dole) má plochu 28.

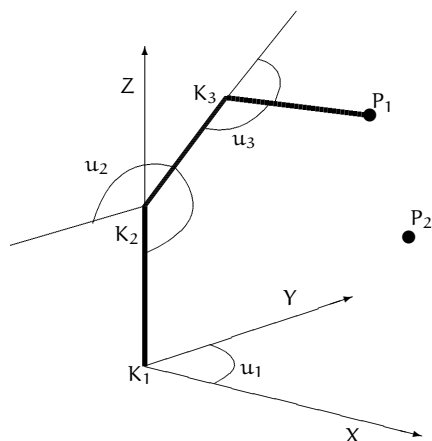
## 525. O ramene

19

Koncový bod ramena robota sa má presunúť v trojrozmernom priestore z bodu  $P_1$  do bodu  $P_2$ . Rameno robota má tri časti a tri kĺby. Každá časť je dlhá 180cm (obrázok). Kĺby umožňujú častiam ramena meniť polohu. Kĺb  $k_1$  má súradnice  $[0, 0, 0]$  a môže sa otáčať horizontálne o uhol  $u_1$  z intervalu  $\langle 0, 90^\circ \rangle$ . Kĺb  $k_2$  má súradnice  $[0, 0, 180]$  a dá sa otáčať vertikálne o uhol  $u_2$  z intervalu  $\langle 0, 270^\circ \rangle$ . Kĺb  $k_3$  sa dá otáčať vertikálne o uhol  $u_3$  z intervalu  $\langle 0, 180^\circ \rangle$ . Uhly sa merajú proti smeru chodu hodinových ručičiek. Uvedomte si, že časti ramena robota sú v jednej rovine.

Riadenie ramena robota sa vykonáva príkazmi  $OTOC(i, j, k)$ , kde  $i, j, k$  sú z množiny  $\{-1, 0, 1\}$ . Vykonanie príkazu  $OTOC$  si ukážeme na príklade:  $OTOC(0, -1, 1)$  spôsobí, že kĺb  $k_1$  nezmení svoju polohu, kĺb  $k_2$  sa otočí tak, že zmenší svoj uhol o jeden stupeň a kĺb  $k_3$  zasa zväčší svoj uhol o jeden stupeň. Uhly pritom musia zostať v prípustných hraniciach, inak sa príslušné rameno nepohne.

ÚLOHA: Napíšte program, ktorý na základe súradníc  $[x_1, y_1, z_1]$ ,  $[x_2, y_2, z_2]$  bodov  $P_1$  a  $P_2$  (vstupné hodnoty), vypíše postupnosť príkazov  $OTOC$ , ktoré by premiestnili koncový bod ramena z bodu  $P_1$  čo najbližšie k bodu  $P_2$ . Skúste pre  $P_1 = [100, 0, 0]$ ,  $P_2 = [100, 100, 0]$ . Snažte sa, aby program generoval čo najkratšiu postupnosť príkazov  $OTOC$ .

**531. O inverzných vranách.**

[20]

Napište program, ktorý prečíta vstupný reťazec znakov predstavujúci prirodzené číslo menšie ako 32000 zapísané slovami (po slovensky) a vypíše jeho číselnú hodnotu. V prípade, že vstupný reťazec nepredstavuje slovami zapísanú číslovku, vypíše správu o chybe. Napríklad:

vstup:	výstup:
dvesto tridsaťpäť	235
tisíc dvadsaťštyri	1024
päť a tridsať	chyba
vrany odleteli	chyba

Použite pravidlá slovenského pravopisu. Vstupný text nemusí obsahovať interpunkčné znamienka.

**532. O podieloch**

► [M50]

Dané sú dve kladné prirodzené čísla  $m, n$ .

- Zistíte celú časť podielu  $m/n$ , základnú periódu, jej dĺžku, predperiódu a jej dĺžku.
- Riešate problém z časti a) s nasledovným obmedzením - v programe môžete použiť najviac 10 jednoduchých premenných, ktorých hodnoty sú v každom okamihu výpočtu celé čísla (t.j. nemôžete použiť pole ani iné dátové štruktúry).

Napríklad pre vstup  $m = 7$  a  $n = 12$  sa vypočíta, že celá časť je 0, perióda (3) má dĺžku 1 a predperióda (58) má dĺžku 2. Pre vstup  $m = 125$ ,  $n = 198$  sa vypočíta, že celá časť je 0, perióda (31) má dĺžku 2 a predperióda (6) má dĺžku 1.

**533. O reťazcoch**

[21]

Dané sú dva vstupné reťazce pozostávajúce z cifier  $0, 1, \dots, 9$  a písmen  $A, B, \dots, Z$ . Dosadte za písmená vyskytujúce sa v reťazcoch cifry  $0, 1, \dots, 9$  tak, aby reťazce s dosadenými hodnotami boli zhodné. V prípade viacnásobného výskytu písmena v reťazcoch dosadzujeme rovnakú hodnotu za všetky jeho výskyty. Zistite, koľko rôznych dosadení existuje a vypíšte reťazce vo všetkých správnych dosadeniach.

Napríklad:

vstup: 0XY1, 0YZZ; výstup: jediné riešenie 0111,  
vstup: 0XXZY23, 0YZY1Z3; výstup: nemá žiadne riešenie,



vstup: 0YZ, XZZ; výstup: riešenia sú 000, 011, 022, 033, 044, 055, 066, 077, 088, 099.

Pokúste sa odhadnúť teoretickú zložitosť Vášho programu – koľko operácií približne vykoná v závislosti na dĺžke reťazca a počte premenných v ňom.

### 534. O postupnosti

M45

Daná je postupnosť prirodzených čísel väčších ako 2 – t.j. 3, 4, 5, 6, ... V každom kroku s postupnosťou vykonáme nasledujúce operácie: určíme vedúci člen, čo je prvý člen postupnosti, preskočíme s prvým členom postupnosti toľko členov, koľko je jeho hodnota a zaradíme ho do postupnosti.

ÚLOHA: Napíšte program, ktorý pre dané vstupné číslo zistí, v ktorom kroku sa prvýkrát stane vedúcim. Napríklad:

1.krok 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, ...

2.krok 4, 5, 6, 3, 7, 8, 9, 10, 11, 12, 13, ...

3.krok 5, 6, 3, 7, 4, 8, 9, 10, 11, 12, 13, ...

4.krok 6, 3, 7, 4, 8, 5, 9, 10, 11, 12, 13, ...

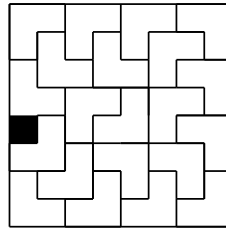
Teda číslo 6 je prvýkrát vedúcim v 4.kroku.

### 535. O šachovnici

► 21

Daná je šachovnica rozmeru  $n \times n$ , kde  $n$  je mocnina 2, t.j.  $n = 2^m$ ,  $m$  je celé. Políčko šachovnice, ktoré má súradnice  $[x, y]$ , je vystrihnuté,  $1 \leq x, y \leq n$ . Napíšte program, ktorý zvyšok šachovnice rozstrihá na triminá tvaru „L“, čo je útvar zložený z troch štvorcov do tvaru „L“. Zamyslite sa pre aké  $m$  a  $[x, y]$  má úloha riešenie, a dokažte.

Napríklad pre  $n = 8$  ( $m = 3$ ), dieťu  $[x, y] = [1, 4]$  úloha má riešenie (jedno z riešení je na obrázku).



### 541. O peniazoch

M25

Koľko existuje rôznych spôsobov vyplatenia - rozmenenia peňažnej sumy z rozmedzia 5 hal. až 10 Kčs pomocou mincí 5, 10, 20, 50 hal. a 1, 2, 5 Kčs?

ÚLOHA: Napíšte program, ktorý pre zadanú sumu zistí počet jej rozmenení. Napríklad suma 25 hal. (t.j. 0.25 Kčs) sa dá rozmeniť 4 spôsobmi.

### 542. O úsečkách

► 30

V rovine je daných  $n$  úsečiek s koncovými bodmi  $A_i = [x_i, y_i]$ ,  $B_i = [u_i, v_i]$ . Napíšte program, ktorý pre dané  $n$  prečíta  $n$  štvoric  $x_i, y_i, u_i, v_i$ ,  $1 \leq i \leq n$  vypíše všetky intervaly  $\langle p_j; q_j \rangle$  s číslami  $f_j$ , pre ktoré platí, že z intervalu  $\langle p_j; q_j \rangle$  na osi  $x$  „vidno“ úsečku s číslom  $f_j$  pri jej kolmom priemete na os  $x$ . Pri zisťovaní viditeľnosti predpokladajte, že úsečky sú „nepriesvitné“, t.j. „nižšia“ úsečka zakrýva „vyššiu“.

Príklad:  $n = 3$ ,

$A_1[1, 1]$ ,  $B_1[10, 7]$ ,

$A_2[3, 1]$ ,  $B_2[5, 1]$

$A_3[5, 7]$ ,  $B_3[9, 3]$

Výsledky:

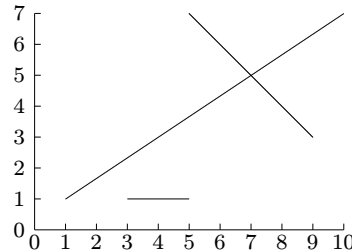
$\langle 1; 3 \rangle$  1

$\langle 3; 5 \rangle$  2

$\langle 5; 7 \rangle$  1

$\langle 7; 9 \rangle$  3

$\langle 9; 10 \rangle$  1



**543. O permutáciách**

► [25]

Napište program, ktorý pre dané  $n$  vypíše všetky  $n$  prvkové permutácie množiny  $\{1, 2, \dots, n\}$ . Snažte sa vytvoriť program, ktorý potrebuje čo najmenej pamäti a pritom pracuje dosť rýchlo. Napríklad: pre  $n = 3$  je výsledok  $(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)$ .

**544. O záhadných slovách**

► [30]

Záhadné refazce sú refazce písmen anglickej abecedy, ktoré vzniknú spojením (zrefazčením) refazcov  $u, v, s, t$ . Refazce  $u, v$  pozostávajú z ľubovoľného (prípadne i nulového) počtu písmen anglickej abecedy a pre refazce  $s, t$  platí, že  $s$  je zrkadlovým obrazom refazca  $u$  a  $t$  je zrkadlovým obrazom refazca  $v$  (zrkadlovým obrazom refazca **abac** je refazec **caba**).

ÚLOHA: Napište program, ktorý načíta slovo a zistí, či je záhadné (v hore uvedenom zmysle). Príklad záhadných slov: **abcdbadc, abab, abccba**.

**545. O kanibaloch a misionároch**

[18]

K ľavému brehu potôčika Ľudožrútia slinka prišlo  $m$  misionárov a  $k$  kanibalov, ktorí sa majú pomocou malej dvojmiestnej loďky priviazanej k brehu, preplaviť na druhý breh potôčika Ľudožrútia slinka. Lenže do loďky sa zmestia najviac dvaja (t.j. dvaja kanibalovia, alebo dvaja misionári, alebo misionár s kanibalom). V loďke môže ísť cez rieku aj samotný kanibal alebo misionár. Prázdna loďka sa cez rieku previezť nedá.

ÚLOHA: Napište program, ktorý určí, akým spôsobom sa majú misionári s kanibalmi preplaviť cez rieku tak, aby v žiadnom momente transportu na žiadnom brehu nebola presila kanibalov nad misionármi, lebo by ich zožrali. Samotní kanibali na brehu sa navzájom nepožerú. Ak úloha nemá riešenie, Váš program vypíše o tom správu.

Príklad pre  $m = k = 3$ :

ľavý breh	loďčka	pravý breh
(m m m k k k)		()
(m m k k)	(m k)->	(m k)
(m m m k k)	<-(m)	(k)
(m m m)	(k k)->	(k k k)
(m m m k)	<-(k)	(k k)
(m k)	(m m)->	(m m k k)
(m m k k)	<-(m k)	(m k)
(k k)	(m m)->	(m m m k)
(k k k)	<-(k)	(m m m)
(k)	(k k)->	(m m m k k)
(k k)	<-(k)	(m m m k)
()	(k k)->	(m m m k k k)

**611. O postupnostiach II**

► [21]

Napište program, ktorý určí pre dané  $n$  a  $k$ , koľko existuje rôznych postupností dĺžky  $n$ , pozostávajúcich len z núl a jednotiek, v ktorých sa nevyskytuje  $k$  núl tesne za sebou. Riešte s podmienkami  $1 < k \leq n$ . Napríklad pre  $n = 5$  a  $k = 3$  existuje 24 rôznych postupností.

**612. O vojakoch**

[25]

$n$  vojakov stojí v rade (t.j. vedľa seba). Na povel „vľavo bok“ sa všetci náhodne otočia o  $90^\circ$ , niektorí vľavo, iní vpravo. O sekundu sa všetci, ktorí stoja tvárou v tvár svojmu susedovi, otočia o  $180^\circ$ . O ďalšiu sekundu tak isto atď.

ÚLOHA: Napište program, ktorý pre dané  $n$  modeluje tento dej, pokiaľ sa niektorý z vojakov otáča. Program vypisuje stav vojakov po každej sekunde. Keď sa už žiadny z vojakov neotáča, program vypíše čas trvania a skončí. Viete zistiť (a zdôvodniť), ako najdlhšie môže trvať tento dej pre dané  $n$ ?

## 613. O kódování

► 25

Pracovníci Střediska pro kosmický výzkum se snaží co nejefektivněji využívat možnosti rádiového spojení na přenos zpráv. Poslední projekt na zkrácení průměrné délky textu byl:

- Zo vzorových textů zpráv se pro každý použitý znak zjistí frekvence jeho výskytu (t.j. počet výskytů znaku v poměru k počtu všech znaků).
- Používané znaky se zakódují do postupností nul a jednotek tak, aby frekventovanější znaky měli kratší kód, méně frekventované delší.
- Zakódované zprávy se musí dáť rozkódovat, preto sa autori rozhodli, že kód žiadneho znaku nesmie byť predponou (začiatkom) iného kódu.
- S použitím takto vzniknutej kódovacej tabuľky sa kódujú a dekodujú správy.

Pre vzorový text s ôsmimi znakmi: BACADAEAFABBAAAGAH je frekvencia nasledovná:

$$\begin{array}{llll} A - \frac{9}{18} & B - \frac{3}{18} & C - \frac{1}{18} & D - \frac{1}{18} \\ E - \frac{1}{18} & F - \frac{1}{18} & G - \frac{1}{18} & H - \frac{1}{18} \end{array}$$

Keby sa tento text kodoval kódom s pevnou dĺžkou (v tomto prípade aspoň 3 bity), tak výsledný zakódovaný text by mal 54 bitov. Jeden z kódov vyhovujúci nášmu projektu je:

$$\begin{array}{llll} A \rightarrow 0 & B \rightarrow 100 & C \rightarrow 1010 & D \rightarrow 1011 \\ E \rightarrow 1100 & F \rightarrow 1101 & G \rightarrow 1110 & H \rightarrow 1111 \end{array}$$

V tomto prípade zakódovaním vzorového textu dostaneme 42 bitov (nul a jednotiek):

100010100101101100011010100100000111001111

Za predpokladu, že vysielané a prijímané zprávy budú mať podobné frekvencie výskytů znaků, sa ušetrí asi 20% bitů (a aj vysielacieho času). Všimnite si, že kód:

$$\begin{array}{llll} A \rightarrow 0 & B \rightarrow 100 & C \rightarrow 1001 & D \rightarrow 1011 \\ E \rightarrow 1100 & F \rightarrow 1101 & G \rightarrow 1110 & H \rightarrow 1111 \end{array}$$

projektu nevyhovuje, lebo kódy znakov B, C nespĺňajú podmienku z bodu c).

ÚLOHA: Vytvorte program, ktorý podľa danej vzorovej zprávy vytvorí a vypíše frekvenčnú tabuľku použitých znakov, týmto podľa bodů b), c), d) priradí a vypíše kód a zakóduje, prípadne rozkóduje ďalší text.

## 614. O minimálnom absolútnom súčte

18

Je dané dvojrozmerné pole  $P$  rozmeru  $n \times n$ , v ktorom sú kladné čísla. Napíšte program, ktorý nájde postupnosť susedných prvků začínajúcich prvkom  $P[1,1]$  a končiacich prvkom  $P[n,n]$  tak, aby súčet absolútnych hodnôt rozdielů susedných prvků vybranej postupnosti bol minimálny. Susedné prvky majú spoločnú „stranu“ – susedné sú  $P[2,3]$  a  $P[2,4]$ , ale nie  $P[2,2]$  a  $P[3,3]$ .

## 615. O labyrintoch

10

Napíšte program, ktorý pre dané  $m$  a  $n$  naplní pole  $L$  s rozmermi  $m \times n$  nulami a jednotkami tak, aby to bol „najprefikanejší labyrint“, aký ste kedy videli. Labyrint znázorní graficky (napríklad na tlačiarňi pomocou hviezdíček) a uvedie, kde je vchod do labyrintu a kde sa nachádza poklad (alebo uväznená princezná). Dodržte dohodu, že jednotka v poli  $L$  znamená múr a nula voľno.

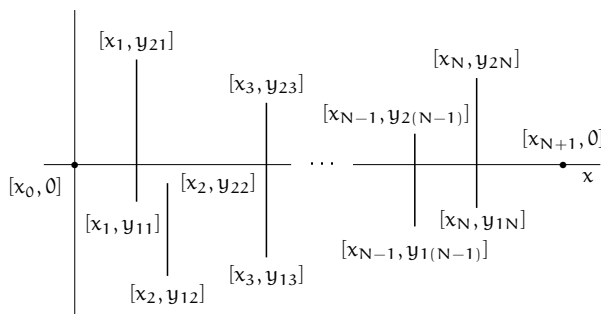
Výsledok nie je presne definovaný, opiera sa o vašu predstavivosť a fantáziu. Poinťou tohoto príkladu je navrhnuť labyrint, ktorý má čo najbližšie k intuitívnej predstave labyrintu.

## 621. O múroch

20

Janko Vtipko nám napísal, že u nich na ihrisku usporadúvajú bežecké preteky v behu medzi múrikmi. Medzi cieľom a štartom sú rovnobežné múriky, ktoré sú kolmé na os cieľ-štart. Žiaden múrik sa nesmie pri behu preskočiť. Jankovi je jasné, že ten, kto si vie nájsť medzi múrikmi najkratšiu dráhu, zvíťazí, preto nás požiadal o pomoc pri jej hľadaní.

Os cieľ-štart stotožníme s  $x$ -ovou osou.  $x_0$  bude  $x$ -ová súradnica štartu, v našom prípade bude vždy rovná 0. Ak bude na trati  $n$  múrikov, potom  $x_{n+1}$  bude  $x$ -ová súradnica cieľa.  $x_i$  bude  $x$ -ová súradnica  $i$ -teho múrika a  $y_{1i}$  a  $y_{2i}$  budú  $y$ -ové súradnice koncových bodov  $i$ -teho múrika.



ÚLOHA: Napíšte program, ktorý pre dané  $n$ ,  $x_i$ ,  $y_{1i}$ ,  $y_{2i}$ ,  $x_{n+1}$  ( $1 \leq i < n+1$ ,  $0 < x_i < x_{i+1}$ ,  $y_{1i} < y_{2i}$ ) nájde najkratšiu lomenú čiaru so začiatkom v bode  $[0, 0]$  a koncom v bode  $[x_{n+1}, 0]$ , ktorá nepretína žiaden múrik. Súradnice bodov, v ktorých sa lomená čiara „láme“ budú  $[a_1, b_1], \dots, [a_m, b_m]$ .

Lomená čiara sa môže múrika (teda jeho koncového bodu) dotýkať, múrik si predstavte ako úsečku. Zo zadania nevyplýva, že  $y_{1i}$  a  $y_{2i}$  majú rôzne znamienka.

622. O  $n$ -uholníku

M40

Vo vrcholoch  $n$ -uholníka sú celé čísla, ktorých súčet je kladný. Máme danú nasledovnú operáciu nad číslami  $x$ ,  $y$ ,  $z$  v troch za sebou idúcich vrcholoch ( $y$  je stredný z nich): ak  $y < 0$ , tak postupne vykonáme tieto priradenia

$$y := -y; \quad x := x - y; \quad z := z - y$$

Príklad: Nech  $x = 3$ ,  $y = -2$ ,  $z = -2$ , potom po vykonaní operácie s danými číslami  $x = 1$ ,  $y = 2$ ,  $z = -4$ .

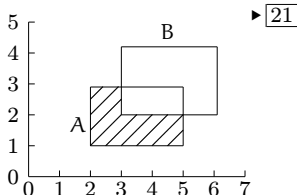
ÚLOHA: Napíšte program, ktorý dostane na vstupe číslo  $n$  a  $n$ -ticu celých čísel, ktoré sú vo vrcholoch  $n$ -uholníka. Program len použitím uvedenej operácie zmení dané čísla tak, aby boli všetky nezáporné. Pritom bude vypisovať, na ktoré čísla (vrcholy) použil operáciu, alebo oznámi, že to pre dané čísla nie je možné.

Zdôvodnite (dokážte) správnosť svojho programu.

## 623. O obdĺžnikoch

Obdĺžnik v rovine je určený súradnicami jeho ľavého horného vrchola a pravého dolného vrchola, jeho strany sú rovnobežné so súradničnými osami.

ÚLOHA: Napíšte program, ktorý pre dané dva obdĺžniky  $A$  a  $B$  vypočíta a vypíše veľkosť plochy rozdielu obdĺžnikov  $A$  mínus  $B$  ( $A - B$ ). Teda plochu  $A$  bez plochy, ktorú z neho „vysekne“  $B$ .





polynómy  $r(x)$  a  $q(x)$  v „čo najčitateľnejšom tvare“, tzn. najlepšie v takom, na aký sme zvyknutí z matematiky (s ohľadom na zápis exponentov). Vstup programu sú polynómy  $a(x)$  a  $b(x)$  zadávané po koeficientoch  $a_n, \dots, a_1, a_0$ .

Príklady :

Vstup: polynóm  $a(x) \dots 2, -4, 3, 2$ ; polynóm  $b(x) \dots -1, 0, 1$ .

Výstup:  $a(x)=2x^3-4x^2+3x+2$ ,  $b(x)=-x^2+1$

$q(x)=a(x) \div b(x) = -2x+4$ ,  $r(x) = a(x) \bmod b(x) = 5x-2$

Odporúčame Vám urobiť si pomocné programy na sčítanie a násobenie polynómov, aby ste mohli skontrolovať výsledky - „ručne“ sa ľahko pomýlite a je to zdĺhavé.

### 632. O žabách

[25]

Na hladine rybníka na  $n+1$  kameňoch sedí  $n$  žiab so štartovými číslami od 1 po  $n$ . Na začiatku žaby sedia na kameňoch v rade vedľa seba, zoradené podľa ich čísiel od najmenšej po najväčšiu. Vedľa žaby s číslom 1 je jeden kameň voľný. Situácia je teda  $\_ 1 2 \dots n$ . Každá žaba môže skákať len na voľný kameň, a to

I - ak je susedný, alebo

II - je susedný susednej žabe

Žabám treba poradiť, v akom poradí majú skákať, aby sedeli na kameňoch v opačnom poradí (od najväčšej po najmenšiu) a voľný kameň bol vedľa žaby s číslom  $n$  (teda  $\_ n \dots 3 2 1$ ). Vašou úlohou je im radiť tak, aby museli najmenej skákať.

ÚLOHA: Napíšte program, ktorý pre dané  $n$  vypíše postupnosť skákaní žiab po kameňoch a počet potrebných skokov typu I a II.

### 633. O kresliči

►[M23]

Predstavte si, že máte riadiť súradnicový zapisovač, ktorý kreslí obrázky pomocou príkazov KRESLI\_DO\_BODU  $x, y$  (kreslí úsečku z momentálneho bodu do bodu  $[x, y]$ ) a POSUN\_DO\_BODU  $x, y$  (presúva sa do bodu  $[x, y]$ ). Aby toto zariadenie kreslilo čo najrýchlejšie, je potrebné minimalizovať dĺžku dráhy pera (dráhu pera tvoria úseky, keď je pero spustené na papieri a kreslí a keď sa presúva ponad papier).

ÚLOHA: Napíšte program, ktorý pre dané  $n$  pomocou príkazov KRESLI\_DO a POSUN\_DO riadi súradnicový zapisovač, ktorý podľa nich vykreslí pravidelný  $n$ -uholník (vpísaný do kružnice s polomerom 1) a všetky jeho uhlopriečky neprechádzajúce stredom  $n$ -uholníka. Snažte sa pri tom minimalizovať dĺžku dráhy pera súradnicového zapisovača. Vypíšte protokol riadenia súradnicového zapisovača pomocou príkazov: KRESLI\_DO číslo, POSUN\_DO číslo, kde číslo je poradové číslo vrcholu, v číslovaní proti smeru chodu hodinových ručičiek. Na začiatku je pero nastavené vo vrchole 1. Nakoniec vypíšte dĺžku dráhy pera.

Napríklad pre  $n=6$ :

```
KRESLI\_DO 2, POSUN\_DO 1, KRESLI\_DO 3, POSUN\_DO 1, KRESLI\_DO 5,
POSUN\_DO 1, KRESLI\_DO 6, POSUN\_DO 1, POSUN\_DO 2, KRESLI\_DO 3,
POSUN\_DO 2, KRESLI\_DO 4, POSUN\_DO 2, KRESLI\_DO 6, POSUN\_DO 2,
POSUN\_DO 3, KRESLI\_DO 4, POSUN\_DO 3, KRESLI\_DO 5, POSUN\_DO 3,
POSUN\_DO 4, KRESLI\_DO 5, POSUN\_DO 4, KRESLI\_DO 6, POSUN\_DO 4,
POSUN\_DO 5, KRESLI\_DO 6
```

DŠŽKA DRÁHY PERA JE 35.7846.

Dĺžka dráhy pera v príklade zrejme nie je minimálna.

### 634. O kódovaní kombinácií

[21]

Nech  $n, k$  sú prirodzené čísla. Uvažujme všetky  $k$ -prvkové podmnožiny množiny  $\{1, \dots, n\}$ , lexikograficky usporiadané (od najmenšej po najväčšiu). Ich počet je  $p = \binom{n}{k}$  a sú očíslované v poradí číslami  $0, \dots, p-1$ .

ÚLOHA: Napište program, ktorý k danému  $n, k$  a číslu  $x$  z množiny  $\{0, \dots, p-1\}$  priradí  $k$ -prvkovú podmnožinu množiny  $\{1, \dots, n\}$  s poradovým číslom  $x$ .

Napríklad keď  $n = 4$  a  $k = 2$ , podmnožina s poradovým číslom 3 je  $\{2, 3\}$ , keď  $n = 4$  a  $k = 4$ , podmnožina s poradovým číslom 0 je  $\{1, 2, 3, 4\}$  a keď  $n = 4$  a  $k = 3$ , podmnožina s poradovým číslom 1 je  $\{1, 2, 4\}$ .

Program by nemal vytvárať všetky podmnožiny od 0-tej po  $x$ -tú.

### 635. O zjednotení obdĺžnikov

21

Obdĺžnik v rovine, ktorého strany sú rovnobežné s osami, je určený súradnicami jeho ľavého dolného a pravého horného vrcholu (nemusia byť celočíselné).

ÚLOHA: Napište program, ktorý pre daných  $n$  obdĺžnikov  $A_1, A_2, \dots, A_n$  vypočíta plochu ich zjednotenia.

Napríklad pre  $n = 4$ ,

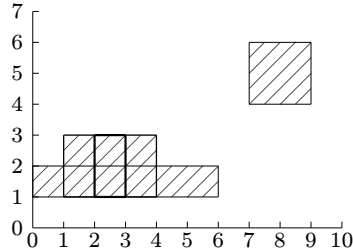
1. obdĺžnik je určený bodmi  $[1; 1], [4; 3]$

2. obdĺžnik je určený bodmi  $[0; 1], [6; 2]$

3. obdĺžnik je určený bodmi  $[2; 1], [3; 3]$

4. obdĺžnik je určený bodmi  $[7; 4], [9; 6]$

Plocha zjednotenia je 13.



### 641. O štvorčekoch

20

Je daná nekonečná štvorčeková rovina (dostatočne veľký štvorčekový papier), na ktorej je zafarbených na čierne  $n$  štvorčekov (všetky ostatné sú biele). V každej sekunde sa vykoná prefarbenie a vznikne nový obrazec štvorčekov.

Pri prefarbovaní bude mať každý štvorček takú farbu, ktorá prevláda na ňom, jeho hornom susedovi a jeho pravom susedovi (susedné štvorčeky sú tie, čo majú spoločnú stranu). Prefarbovanie sa zastaví, ak už neexistuje žiaden čierny štvorček.

ÚLOHA: Napište program, ktorý pre daný počiatočný obrazec počíta a postupne vypisuje vznikajúce obrazce. Existuje obrazec, ktorého prefarbovanie sa nikdy neskončí? Zamyslite sa nad tým, ako dlho môže prefarbovanie trvať pre  $n$  štvorčekov. Svoju hypotézu dokažte.

### 642. O čudných permutáciách

27

Nech  $M$  je daná neprázdna podmnožina prirodzených čísel obsahujúca  $m$  prvkov navzájom rôznych.

ÚLOHA: Napište program, ktorý pre danú množinu  $M$  vytvorí takú permutáciu jej prvkov, že medzi žiadnymi dvoma prvkami v tejto permutácii nie je ich aritmetický priemer. Príklad:  $M = \{1, 2, 3, 4, 5, 6\}$ ; výsledkom je napríklad  $\{1, 5, 6, 3, 2, 4\}$ .

Prvok „medzi“ dvoma prvkami nemusí byť susedný ani s jedným z nich.

### 643. Kreslenie jedným ťahom

25

Možno viete, že ľubovoľný spojitý obrázok zložený z čiar sa dá nakresliť jedným uzavretým ťahom (teda takým, že kreslenie skončíme v tom bode, z ktorého sme začali) práve vtedy, keď sa v každom priesečníku čiar stretáva párny počet čiar. Priesečníky nazveme vrcholmi, čiary nazveme hranami.

ÚLOHA: Napište program, ktorý prečíta počet vrcholov  $n$  a niekoľko dvojíc vrcholov  $(i, j)$ , ktoré sú spojené hranou. Potom vypíše takú postupnosť čísel vrcholov, ktorá určuje, ako sa dá nakresliť daný obrázok jedným uzavretým ťahom. Ak sa daný obrázok nedá nakresliť jedným uzavretým ťahom, program to musí zistiť a oznámiť (pozor, môže to byť aj vtedy, ak obrázok nie je spojitý).

**644. O upratovaní**

► 21

V sklade majú jeden druh krabíc v troch farbách, a to modré, červené a biele. Všetky krabice majú na jednej polici. Skladník potrebuje usporiadať krabice tak, aby boli uložené najprv všetky biele krabice, potom všetky modré krabice a na konci červené krabice. Všetky ostatné police však v sklade boli obsadené (prefarbiť krabice nemohol). Skladníkovi neostalo iné len pohýbať rozumom, aby vyriešil úlohu, ktorú si zaumienil.

ÚLOHA: Je dané  $m$  prvkové pole (polica) a tri farby. Krabica farby  $f$  je v poli reprezentovaná poradovým číslom farby  $f$ . Napíšte program, ktorý pre dané pole  $M$  usporiada prvky poľa (krabice) tak, ako to potrebuje skladník. Nemôžete pri tom však použiť pomocné pole (pomocnú policu), ani si spočítať, koľko prvkov má byť ktorej farby, a potom ich do poľa len vypísať (prefarbenie krabíc). Môžete len navzájom vymieňať niektoré dva prvky poľa. Program vypíše postupnosť dvojíc krabíc v takom poradí, ako ich má skladník vymieňať, aby policu upratol do požadovaného tvaru. Snažte sa, aby skladník nemusel urobiť viac než  $m$  výmen krabíc.

**645. O kalkulačke**

24

Dôležitou súčasťou prekladačov programovacích jazykov je preklad aritmetických výrazov do strojového kódu. Vašou úlohou bude naprogramovať takýto jednoduchý prekladač.

Strojový kód počítača si nahradíme jednoduchým kalkulačkovým kódom. Predstavme si, že máme kalkulačku, ktorá má 26 pamäťových miest označených A až Z, dostatočne veľa pomocných pamäťových miest označených  $p_1, p_2, \dots$  a jeden špeciálny register – sumátor (to je akoby displej kalkulačky). Sumátor označme  $\$$ . V ďalších riadkoch  $p$  označuje meno ľubovoľného (aj pomocného) pamäťového miesta a  $p_k$  označuje meno ľubovoľného (aj pomocného) pamäťového miesta alebo celočíselnú konštantu. Kalkulačka pozná takéto príkazy:

$\$ := p_k$	do sumátora ulož hodnotu z $p_k$
$p := \$$	hodnotu zo sumátora ulož do $p$
$\$ + p_k$	k sumátoru pripočítaj hodnotu z $p_k$
$\$ - p_k$	od sumátora odčítaj hodnotu z $p_k$
$\$ * p_k$	vynásob sumátor hodnotou z $p_k$
$\$ / p_k$	vydeľ sumátor hodnotou z $p_k$
$\$ ^ p_k$	umocni sumátor na hodnotu $p_k$
@ $\$$	zmeň znamienko sumátora (vynásob ho $-1$ )

Príklad: Predpokladajme, že v pamätiach A a B už máme uložené nejaké hodnoty a chceme vypočítať hodnotu výrazu  $(-A * B + B * 2)^2$  a výsledok uložiť do pamäti C (chceme vlastne vykonať priradovací príkaz  $C := (-A * B + B * 2)^2$ ), musíme dať kalkulačke (napríklad) takúto postupnosť príkazov:

$\$ := A, \$ * B, @\$, P_1 := \$, \$ := B, \$ * 2, \$ + P_1, \$^2, C := \$$

ÚLOHA: Napíšte program, ktorý prečíta priradovací príkaz a vypíše postupnosť príkazov pre kalkulačku, ktorá vykoná tento priradovací príkaz. Výraz na pravej strane je zostavený z mien pamäťových miest, celočíselných konštánt, binárnych operátorov  $+$ ,  $-$ ,  $*$ ,  $/$ ,  $^$ , unárnych operátorov  $+$ ,  $-$  a zátvoriek. Pre tvar výrazu na pravej strane si môžete vybrať jeden z dvoch variantov (variant b je podstatne ťažší, stačí vyriešiť len ten).

- Výraz je úplne uzátvorkovaný. To znamená, že každý operátor je spolu so svojimi dvoma operandmi (alebo jedným, ak je to unárne mínus alebo plus) uzavretý v zátvorkách. Postupnosť výpočtu je teda jasne daná zátvorkami. (Úplne uzátvorkovaný variant výrazu z našej ukážky je  $((-(A * B)) + (B * 2))^2$ ). Program môže predpokladať, že doň vstupujú len správne a úplne uzátvorkované výrazy.
- Pre poradie operácií platia bežné pravidlá priority - najskôr sa umocňuje, potom sa násobí a delí a až potom sa sčítuje a odčítuje, zátvorky menia poradie vykonávania operácií bežným spôsobom. Unárne plus a mínus majú prioritu ako sčítanie. Vo výraze



nemôžu nasledovať po sebe dva operátory (nemôžeme napísať  $A + -B$ , ale  $A + (-B)$ ).

Program musí zistiť, či je výraz správny - ak je nesprávny, vypíše správu.

Pri riešení b) odporúčame použiť oddelený lexikálny analyzátor (pozri úlohu 625)

### 711. O protivných intervaloch

► 23

Na číselnej osi máme daných  $n$  uzavretých intervalov  $\langle a_i, b_i \rangle$ . Zaujímá nás, ktoré čísla ležia práve v párnom počte z týchto intervalov. Množinu týchto čísel môžeme opísať rôznymi spôsobmi ako zjednotenie niekoľkých intervalov. Nás však zaujíma to vyjadrenie, ktoré obsahuje najmenej intervalov.

Napríklad pre dané  $n = 4$  a intervaly  $\langle 2, 5 \rangle$ ,  $\langle 1, 5 \rangle$ ,  $\langle 0, 8 \rangle$  a  $\langle 4, 7 \rangle$  je možné množinu bodov, ktoré sa nachádzajú práve v párnom počte intervalov, opísať napríklad ako zjednotenie piatich intervalov  $(-\infty, 0)$ ,  $\langle 1, 2 \rangle$ ,  $\langle 4, 5 \rangle$ ,  $\langle 5, 7 \rangle$  a  $(8, +\infty)$ . Dá sa však opísať aj ako zjednotenie štyroch intervalov  $(-\infty, 0)$ ,  $\langle 1, 2 \rangle$  a  $\langle 4, 7 \rangle$  a  $(8, +\infty)$ . Všimnite si, že výsledné intervaly nemusia byť uzavreté.

ÚLOHA: Napíšte program, ktorý prečíta prirodzené číslo  $n$  a  $n$  dvojíc koncových bodov uzavretých intervalov  $\langle a_i, b_i \rangle$  a nájde množinu bodov, ktoré sa nachádzajú v práve párnom počte intervalov. Množinu vypíše v tvare zjednotenia čo najmenšieho počtu intervalov.

### 712. O bezpečnostnom vedení

18

Bezpečnostným vedením nazveme štvorcovú sieť z  $n \times n$  vodivých drôtov, ktoré sú v uzlových bodoch vodivo spojené. Drôty sú očíslované (pre jednoznačnosť zhora nadol od 1 po  $n$  a zľava doprava od 1 po  $n$ ). Priešeknik drôtov  $i$  a  $j$  označíme  $[i, j]$ . Vývody spojenia sú v protilahlých vrcholoch štvorca (vo vrcholoch  $[1, 1]$  a  $[n, n]$ ).

Vojaci majú vedenú telefónnu linku z miesta A do miesta B bezpečnostným vedením  $n \times n$  (kde A a B sú vývody spojenia). Nepriateľ im objavil a zničil niektoré vodivé uzly. Zničené uzly nevedú prúd ani v jednom smere (celý uzol aj s kusmi drôtov okolo je zo siete vyseknutý). Zaujímá nás, či sa napriek tomu dá dosiahnuť spojenie medzi mestami A a B.

ÚLOHA: Napíšte program, ktorý pre danú veľkosť bezpečnostného vedenia a dané súradnice zničených uzlov vypíše, či je možné nadviazať spojenie medzi vrcholmi  $[1, 1]$  a  $[n, n]$ .

### 713. O rímskych číslach

10

Na vstupe sú dva reťazce, obsahujúce iba znaky I, V, X, L, C, D a M, reprezentujúce dve čísla v rímskom zápise. Napíšte program, ktorý vypíše ich súčet a rozdiel tiež v rímskom zápise, pričom bude pracovať ako rímski úradníci iba s ich rímskym zápisom. Môžete predpokladať, že prvé číslo je väčšie ako druhé.

Napríklad pre vstupné hodnoty XXIII a IV sú výsledné hodnoty XXVII a XIX.

### 714. O blábole

20

Kozmické Stredisko pre Prieskum (KSP) vyslalo sondu na neznámu planétu. Sonda funguje tak, že preskúma svoje okolie a odvysíla do Strediska to najzaujímavejšie, čo sa tam nachádza. Planéta je veľmi ďaleko od Zeme a rôzne šumy správu sondy rušia a komolia. Preto sonda správu vysíla stále dookola.

V Stredisku majú už len jednu úlohu. Z prijatého rušeného textu získať maximálne pravdepodobnú správu sondy. Maximálne pravdepodobná správa je taká, že keď sa cyklicky podpíše pod prijatý text, je počet znakov, v ktorých sa líšia, minimálny.

ÚLOHA:

- Pomôžte Kozmickému Stredisku pre Prieskum a napíšte program, ktorý pre daný text a číslo  $k > 0$  vypíše všetky maximálne pravdepodobné správy dĺžky  $k$ . (Dĺžka textu nemusí byť celočíselným násobkom čísla  $k$ .)
- Zistíte, čo zaujímavé objavila sonda na planéte, keď prijatý text vyzeral takto:

BLABOLQIALVBPTPGVYWXRBTAWXNCTZIAWQFLGBDNPZYKIAWEFHIONIIIEYHNSAXH

EKSEHVWQDEQDYDYMABFDYIBCVKCNTRNBDKZPZCSBHICIZIUNFIUGCKMCFKFQIGUNAEP  
 IRUKYVSEIIFUBCXFMUIHCBTSBVNGBMDNYDVCORXMDNIGHZPRREGCEQRILDMPPBOBOIXFU  
 TEHQNDTDNMNLKEYLSHYZKAFADIAytaunvvzegyqmqxjxvvutodvknqcxcpacajyvuiNK  
 ASNGKTDOVJUDTTISPMYAZXIBBHLLEJPGNGXDDXEGOXQOENKHKCAAUJRQYXONFPAPRFJ  
 QNVLCRIMOVGGZANYUMDAFXEHLNXXVNBGBVENICHZQQGNBNKQRNNDNERRCGBSHXZLTGM  
 CIHAXFEJXUGFADNGUXVWPLYROEVTFKCWPWGCAVVCPSOUOKMSMDFHDOHTMYKSPIKDOUNI  
 YZTLECVANUHLNRIERUVNQIRGIOHFRLLUORICNICGHYJAMNMMUXCJCENUVARWOMRNDYNW  
 PDNAZECWINYXCMFMKGRSTSTBTFHISXANBRYOSYNMQHGYBNJZVEKDEUOVDBWZSGOILRPKWD  
 PNQCNIHQUSNIRSKQIOKGTWLAUKHKCMFYIYVIALKALBAYEZAJZUJBWFESQQRACEUEGPWAB  
 BEMXPPHNFCKMZHRHKCWFQTQNAONUMNTINLRRSLUIVURCQZFEPVHNKNRTUBRHCNWZWNHMAI  
 GVRNJZWUNRBIWCZENOGWDOQFIIINPSGDLAMPETWTSEHICQASRPLMUEPWUDKEBBIAMZPZ  
 ZNDHQIMFYPIJFZSBYEMFSMCBEMHNYCSQCMTLRPWVHKHCQEBQCMOHWWYJBAAIRZTRSEBQ  
 NHHKGFCHRCVWBKJBjFVEECETfIEWGIBYNUYROXRDRNESIIBVTSDJKLWRCITCLWMISVQ  
 KCCWJFQZPUTREVKIFSDRDENCNSJAHQCRXWBIINRIFUTNASWBMNVAMGBUCMNDPNCYWBRPE  
 DLCCTABRIPAAKYXDKIQNTJKXZRZDMKBZCYMXRMVXFIXRSTTVTQQNZBONFKVCLTANHGGC  
 KQOLTSHWSJIWLSNUWISYRJNMGWVICZIXBIRNLARBIMDYJWUYCYCMGJHMMZQKHHDZCGOO  
 QRDPOEAIRAWGFKCVHONZHNTITOXBWNYFDYJXHZISDPEDCHIBGSKLFAUIELPHNJKYDINP  
 XYDAIVARCPAAJBEHHWIZOROJTAICMICNDHYXLJDXATEQFCENFRWONHLKUDRTVP

### 715. O domine

[21]

Na stole máme rozsypaných  $n$  kociek matematického domina. Každá kocka takéhoto domina má na každej polovici jednu cifru v desiatkovej sústave (t.j. celé číslo z intervalu  $(0, 9)$ ). Tieto kocky treba poukladať širšími stranami k sebe a číslami nahor, pričom sa môžu aj otáčať. Takýmto uložením vzniknú dve čísla, pričom jedno sa číta v dolnom rade od prvej kocky po poslednú a druhé v hornom rade od poslednej po prvú.

ÚLOHA: Napište program, ktorý

- uloží kocky tak, aby súčet dvoch vzniknutých čísel bol maximálny,
- uloží kocky ako v prípade a), ale s predpokladom, že kocky domina sa môžu ukladať k sebe, iba ak majú aspoň jednu cifru rovnakú (a potom aj musia ležať touto cifrou pri sebe) a nemusíme použiť všetky kocky.

### 721. O priemere

[18]

Daná je postupnosť  $n$  čísel,  $n > 0$ . Treba nájsť čo najdlhšiu súvislú podpostupnosť s priemerom väčším alebo rovným ako dané  $p$ .

ÚLOHA: Napište program, ktorý načíta kladné celé číslo  $n$ , reálne číslo  $p$  a postupnosť reálnych čísel dĺžky  $n$  a vypíše prvý a posledný index hľadanej podpostupnosti. Ak neexistuje, tak vypíše o tom správu.

Priemer postupnosti  $a_1, a_2, \dots, a_n$  je číslo  $\bar{x} = \frac{a_1 + a_2 + \dots + a_n}{n}$ .

### 722. O veľkom zlomku

[18]

Máme daný zlomok, ktorého čitateľ aj menovateľ sú v tvare súčinu viacerých celých čísel. Zlomok má teda tvar

$$\frac{a_1 \cdot a_2 \cdot \dots \cdot a_n}{b_1 \cdot b_2 \cdot \dots \cdot b_m}.$$

Pritom každé  $a_i$  a  $b_i$  je dostatočne malé (zmestí sa do celočíselnej premennej), ale  $n$  a  $m$  môže byť veľké (napríklad 100). Súčiny v čitateli a menovateli sa teda už takmer iste nezmestia do jednej číselnej premennej. Zaujímá nás základný tvar daného zlomku (teda tvar, v ktorom bude čitateľ nesúdeliteľný s menovateľom). Príklad:

$$\frac{25 \cdot 2 \cdot 320 \cdot (-125) \cdot 131 \cdot 100}{128 \cdot 25 \cdot 25 \cdot 25 \cdot 15} = \frac{-2620}{3}$$

ÚLOHA: Napište program, ktorý prečíta celé čísla  $m$  a  $n$ ,  $m, n \leq 1000$  a dve postupnosti celých čísel  $a_1, a_2, \dots, a_n$  a  $b_1, b_2, \dots, b_m$  (všetky  $|a_i|, |b_i| \leq 500$ ) a vypíše základný tvar zlomku  $\frac{a_1 \cdot a_2 \cdot \dots \cdot a_n}{b_1 \cdot b_2 \cdot \dots \cdot b_m}$ . Výsledok programu môže byť v rovnakom tvare ako daný zlomok (teda v tvare podielu dvoch súčínov).

### 723. O stabilných úradníkoch

[18]

V úrade je  $n$  úradníkov, ktorí každú hodinu preberajú lajstrá. V celom úrade je sto miliónov lajstier na úradníckych stóloch. Každú celú hodinu zaznie gong a každý  $i$ -ty úradník odobere  $i$ -tému kolegovi  $U_{i,j} > 0$  percent jeho papierov zo stola a dá si ich na stôl (z ktorého odobrali zase iní kolegovia). Ak pre maticu  $U$  platí, že súčet prvkov v každom stĺpci je 100, tak sa po niekoľkých rokoch činnosti úradu kópka každého úradníka ustáli na nejakej hodnote.

ÚLOHA: Napište program, ktorý pre dané  $n$  a maticu  $U$  takú, že súčet prvkov v každom stĺpci je 100, nájde „stabilné“ rozloženie lajstier u úradníkov.

### 724. O zjednotení

[21]

Máme postupnosť disjunktných množín  $z_1, z_2, \dots, z_n$ . Poznáme ich počty prvkov  $v_1, v_2, \dots, v_n$ . Chceme získať zjednotenie  $z_1 \cup z_2 \cup \dots \cup z_n$ . Na určenie zjednotenia môžeme použiť iba operáciu  $\cup$ , ktorá zjednotí dve množiny  $z_i$  a  $z_j$ . Vykonanie tejto operácie trvá  $v_i + v_j$  časových jednotiek (časovú jednotku označme *čj*). Zaujímá nás, ako máme postupne zjednocovať množiny, aby bol výsledný čas čo najmenší.

Napríklad ak máme tri množiny 2, 10 a 7 prvkové, tak ak zjednotíme najprv prvé dve (to trvá  $2 + 10 = 12$  čj, vznikne 12 prvková množina), potom k výsledku pridáme tretiu (na  $12 + 7 = 19$  čj), trvá to spolu 31 čj. Ak však zjednotíme prvú a tretiu ( $2 + 7 = 9$  čj) a potom k nim druhú ( $9 + 10 = 19$  čj), výsledný čas bude len 28 čj.

ÚLOHA: Napište program, ktorý prečíta prirodzené číslo  $n$  a postupnosť prirodzených čísel  $v_1, v_2, \dots, v_n$ , ktoré predstavujú veľkosti množín. Výsledkom programu bude predpis na získanie zjednotenia za minimálny čas a tento čas. Predpis sa skladá z „príkazov“  $i \cup j = k$ , kde  $i$  a  $j$  sú čísla množín, ktoré sa majú zjednotiť a  $k$  je číslo, ktorým označíme číslo výslednej množiny.

Prvý predpis z nášho príkladu by sme zapísali napríklad  $1 \cup 2 = 4$ ,  $4 \cup 3 = 5$ . (Čísla označujúce výsledné množiny môžete voliť ľubovoľne, najlepšie je to však postupne od  $n + 1$ ).

### 725. O súboroch

► [25]

Softwarové družstvo SODR si kúpilo lacno počítač ABM kompatibilný s ničím a potrebuje doňho dorobiť základné programové vybavenie. Tento počítač má jeden disk, na ktorom sú uložené súbory. Úplné meno súboru na počítači ABM sa skladá z názvu súboru, bodky a prípony. Názov súboru aj prípona sa skladajú iba z veľkých písmen a majú najviac 20 písmen. Typické meno súboru na počítači ABM je napríklad KONDICIOGRAMSUPRAVOU.PASCALOVSKYPROGRAM.

Pri práci so súborami (kopírovanie, rušenie a pod.) je niekedy potrebné označiť naraz viac súborov so spoločnou niektorou časťou mena. Na to slúži hviezdičková konvencia: Pri zápise mena súboru sa môžu použiť znaky  $?$  (otáznik) a  $*$  (hviezdička), pričom otáznik zastupuje jedno ľubovoľné písmeno a hviezdička zastupuje ľubovoľne dlhý reťazec (aj prázdny) ľubovoľných písmen. Napríklad zápisu  $?ONDI*S*UPR*. *PAS*$  vyhovuje aj horeuvedené meno súboru.

ÚLOHA: Napište čo najrýchlejší program, ktorý najprv prečíta meno súboru v hviezdičkovej konvencii a potom pre prichádzajúce úplné mená súborov (teda bez hviezdičiek a otáznikov) vypisuje VYHOVUJE alebo NEVYHOVUJE, podľa toho, či úplné meno vyhovuje danému hviezdičkovému zápisu alebo nie. Činnosť sa končí súborom s prázdny názvom

aj príponou (t.j. iba bodka). Počítajte s tým, že mien, ktoré má program vyhodnotiť pre jeden hviezdíčkový zápis, môže byť veľmi veľa.

### 731. O tetraminách

[23]

Na štvorčekovom papieri máme nakreslený pôdorys miestnosti. Predpokladáme, že miestnosť sa skladá z nejakého počtu celých štvorčekov, ktoré navzájom susedia aspoň jednou stranou. Túto miestnosť máme „vyparketovať“ parketami tvaru tetramín. Tetramíná sú útvary zo štyroch štvorčekov (vysvetlenie pre hráčov tetrisu: tetramíná sú útvary, ktoré padajú v tetrise). Existujú takéto tetramíná:

```
XXXX XXX XXX XX XX
  X  X  XX XX
```

Pri parketovaní môžeme jednotlivé tetramíná ľubovoľne otáčať aj prevracáť a každé z nich môžeme použiť pri parketovaní ľubovoľný počet krát.

ÚLOHA: Napíšte program, ktorý prečíta zo vstupu tvar miestnosti a vytlačí jej pokrytie tetramínami, alebo ak pokrytie nie je možné, napíše o tom oznam.

Vstup zadávajú tak, že miestnosť ohraničíte obdĺžnikom  $m \times n$  a zadáte  $m, n$  a pole  $m \times n$  núl a jednotiek, kde jednotky určujú štvorčeky patriace miestnosti. Program môže predpokladať, že na vstupe je zadaná jedna miestnosť (nemusí teda kontrolovať správnosť vstupu). Na obrázku vpravo je vstup pre  $m = 4, n = 5$ .

Výstup požadujeme v tvare poľa  $m \times n$  čísiel, kde nula označuje štvorček, ktorý neprislúcha miestnosti a číslami  $i$ , kde  $2 \leq i \leq k+1$  ( $k$  je počet použitých tetramín), je vyznačená poloha  $(i-1)$ -ho položeného tetramína.

```
0 1 1 0 0
1 1 1 1 0
1 1 1 1 1
1 1 1 1 1

0 2 2 0 0
3 2 2 4 0
3 4 4 4 5
3 3 5 5 5
```

### 732. Poľský zápis

► [21]

Majme aritmetický výraz  $A*B+C*D$ . V poľskom zápise tento výraz vyzerá  $AB*CD*+$ . Vidíte, že na rozdiel od normálneho zápisu sa v poľskom zápise zapisujú najprv operandy a za nimi príslušná operácia:

```
operandy : A, B      operácia : *
operandy : C, D      operácia : *
operandy : (A B *), (C D *) operácia : +
```

Výhodou tohoto zápisu je napríklad presné určenie poradia vykonávania operácií.

ÚLOHA: Napíšte program, ktorý načíta normálny aritmetický výraz obsahujúci operácie  $+$ ,  $-$ ,  $*$ ,  $/$ ,  $^$  (so zvyčajnou prioritou, mínus môže byť aj unárne), zátvorky  $(,)$ , jednoduché premenné  $A, B, \dots, Z$  a celočíselné konštanty a prepíše ho do poľského zápisu<sup>6</sup> (pre jednoznačnosť v poľskom zápise bude unárne mínus značiť znakom  $@$ ).

Príklady:

vstup	výstup
$(-D * (A + B)^D) * D + T$	$D @ A B + D ^ * D * T +$
$17 * (D * D) * D * D$	$17 D D * * D * D *$
$A + B + C + D + E$	$A B + C + D + E +$
$A/B/C/D$	$A B / C / D /$
$A - A * 22 - B - C * D$	$A A 22 * - B - C D * -$

<sup>6</sup> Zápis sa nazýva poľský, lebo ho vymyslel poľský matematik J. Lukasiewicz, 1878–1956. Na čo je to dobré? Keď máme aritmetický výraz zapísaný v poľskom zápise, vieme ho jednoducho vypočítať na „kalkulačke so zásobníkom“. Postupujeme tak, že čítame zápis zľava doprava. Ak nájdeme operand, dáme jeho hodnotu na vrch zásobníka. Ak nájdeme operátor, tak spravíme príslušnú operáciu nad dvomi vrchnými prvkami zásobníka. Tie zo zásobníka vyberieme a namiesto nich vložíme výsledok operácie. Pri unárnom mínus postupujeme podobne, len vyberáme jeden operand. Keď takto spracujeme celý výraz, zostane nám v zásobníku výsledok. Niektoré počítače (a aj kalkulačky Hewlett-Packard) majú možnosť počítat výrazy uvedeným spôsobom a aj preto sa poľský zápis používa v prekladačoch programovacích jazykov ako medzikód (záujemcom odporúčame 5. kapitolu z knižky [42])

Predpokladajte správnosť vstupného výrazu. Všetky operácie sú zľava asociatívne, teda  $A/B/C/D = (((A/B)/C)/D)$ ,  $A - B - C - D = (((A - B) - C) - D)$ , atď.

### 733. O telefónnych maniakoch

[18]

V jednej krajine je  $n$  telefónnych maniakov. Pre jednoduchosť predpokladajme, že majú telefónne čísla  $1, 2, \dots, n$ . Každý maniak má telefónny zoznam, obsahujúci telefónne čísla jeho známych (a jeho vlastné), pričom neustále telefonuje všetkým maniakom, ktorých telefónne čísla na tomto zozname má (sebe samozrejme netelefonuje). Aby mohli maniaci spoznávať aj ďalších kolegov, oznamujú si navzájom pri každom telefonáte všetky telefónne čísla, ktoré majú na svojich zoznamoch, čím sa ich zoznamy rozrastajú. Po istom čase nastane situácia, že už žiaden maniak nemôže spoznať žiadneho nového maniaka.

ÚLOHA: Napíšte program, ktorý pre daný počet maniakov  $n > 0$  a telefónny zoznam každého maniaka na začiatku, vypíše všetky skupiny maniakov, ktorí na konci budú navzájom poznať svoje telefónne čísla (každú skupinu vypíše len raz). Napríklad pre  $n = 5$  a telefónne zoznamy

číslo telefónneho maniaka	1	2	3	4	5
telefónny zoznam maniaka	1, 2, 5	2, 5	3, 5	4	1, 5

je 1. skupina  $\{1, 2, 3, 5\}$  a 2. skupina  $\{4\}$ .

### 734. O cólštoku

[28]

Cólštok je programátorské náradie veľmi podobné skladaciemu stolárskemu metru, ale je celý rovnomerne popísaný znakmi a spĺňa nasledovné podmienky:

1. Má  $n \geq 3$  zlomov, pričom zlomy môžu byť buď všetky na jednom znaku, alebo všetky medzi dvoma znakmi (viď príklady).
2. Dĺžka každého úseku medzi dvoma zlomami je rovnaká (prítom aj začiatok a koniec cólštoku považujeme za zlom).
3. Pri tzv. „harmonikovom zložení“ sú všetky na sebe ležiace znaky rovnaké.

Príklady cólštokov:

$abcbaabc$ ; zložený  $\begin{matrix} a & & aa \\ b & b & b \\ & cc & c \end{matrix}$  iný cólštok  $abcbabc$  zložený  $\begin{matrix} a & & a \\ b & b & b \\ & c & c \end{matrix}$ .

ÚLOHA: Napíšte program, ktorý z danej skupiny znakov vyberie a vypíše najdlhší reťazec znakov, ktorým sa dá popísať cólštok. Na cólštoku môžu byť teda použité len tie znaky, ktoré sa nachádzajú v danej skupine a najviac toľkokrát, koľkokrát sa nachádzajú v danej skupine. Napríklad pre skupinu znakov  $a, a, b, b, c, c, d$  je to cólštok  $abcabcba$ .

(Možno ho zložiť  $\begin{matrix} a & & a \\ b & b \\ c & c \\ & b \end{matrix}$ ).

### 735. O tučnej úsečke a bystrozrakých bodoch

[23]

V rovine máme daných  $n$  bystrozrakých bodov a jednu tučnú úsečku, cez ktorú nevidno.

ÚLOHA: Napíšte program, ktorý načíta číslo  $n > 0$ ,  $n$  bodov  $[x_i, y_i]$  a koncové body  $[a_x, a_y]$ ,  $[b_x, b_y]$  tučnej úsečky a vypíše počet dvojíc tých bodov, ktoré na seba „nevidia“ cez tučnú úsečku (t.j. ich spojnica prechádza tučnou úsečkou).

Zákaz: Snažte sa nepoužiť žiadnu goniometrickú funkciu ( $\sin$ ,  $\cos$ ,  $tg$ ,  $arctg$ , atď.).

### 811. O výbere

[20]

Napíšte program, ktorý načíta pole reálnych čísel  $A$  dĺžky  $n$ , reálne číslo  $t$  a prirodzené číslo  $k$ ,  $0 < k \leq n$  a zistí, či je možné vybrať  $k$  prvkov poľa  $A$  (každý prvok možno vybrať

iba raz, t.j. indexy vybraných prvkov musia byť rôzne) tak, aby ich súčet bol menší než  $t$  (výstupom programu je odpoveď **ANO** alebo **NIE**).

Snažte sa, aby ste nemenili pole  $A$  a nepoužívali iné polia, smerníky a pod.

### 812. O postupnosti II

M50

Sú dané prirodzené čísla  $m$ ,  $n$ ,  $m \nmid n$  a  $n \nmid m$ . Napište program, ktorý vygeneruje a vypíše najdlhšiu postupnosť celých čísel, v ktorej je súčet každých  $m$  po sebe idúcich členov záporný a každých  $n$  po sebe idúcich členov kladný (postupnosť môže byť aj prázdna).

Keďže takýchto postupností môže byť nekonečne veľa, vypíšte vždy iba jednu. Napríklad pre  $m = 7$  a  $n = 11$  má hľadaná postupnosť dĺžku 16 a jedna z nich je

5, 5, -13, 5, 5, 5, -13, 5, 5, -13, 5, 5, 5, -13, 5, 5.

### 813. O zaostalej krajine

► 25

Na mape zaostalej krajiny je  $n$  miest očíslovaných  $1, 2, \dots, n$  so súradnicami  $(x_1, y_1)$ ,  $(x_2, y_2)$ ,  $\dots$ ,  $(x_n, y_n)$ . Vládcu sa práve rozhodol svoju krajinu elektrifikovať. V meste č. 1 nechal postaviť elektrárňu a teraz ešte potrebuje vybudovať elektrickú sieť. Pozval si preto inžinierov z celej krajiny a pod trestom smrti im nariadil postaviť túto sieť za minimálne prostriedky. Inžinieri sa ocitli v nezávideniahodnej situácii a potrebujú vašu pomoc, lebo je im jasné iba to, že čím je vedenie kratšie, tým je aj lacnejšie.

ÚLOHA: Napište program, ktorý načíta počet miest  $n$ , ich súradnice na mape  $(x_i, y_i)$ , kde  $1 \leq i \leq n$  a vypíše všetky také dvojice miest, ktoré treba spojiť elektrickým vedením, aby malo každé mesto elektrinu a dĺžka vedenia bola čo najkratšia.

### 814. O armáde

► 20

Jedna armáda sa chystá na veľký husí pochod (t.j. v zástupe). V armádnych predpisoch však stojí: „...pri ľubovoľnej bojovej činnosti musí byť nadriadený schopný jediným pohľadom prekontrolovať všetkých svojich podriadených. Toto pravidlo musia obzvlášť prísne dodržiavať všetci členovia pochodového útvaru (ďalej ČPU)...“. Preto musia pri pochode všetci vojaci v zástupe zachovávať také poradie, aby mal každý nadriadený v ČPU všetkých svojich podriadených pred sebou.

ÚLOHA: Na pochod sa chystá  $n$  vojakov očíslovaných  $1, 2, \dots, n$ . Kto je komu nadriadený a podriadený určujú armádne predpisy vymenovaním  $k$  dvojíc (nadriadený <sub>$i$</sub> , podriadený <sub>$i$</sub> ), kde  $1 \leq i \leq k$ . Napište program, ktorý vypíše poradie, v akom sa majú vojaci zaradiť do zástupu, aby dodržali vyššie uvedené predpisy. Ak to nie je možné, program to musí zistiť a ohlásiť.

Napríklad pre  $n = 6$  a  $k = 5$ , nadriadený 1 4 2 1 2 ;  
podriadený 5 5 3 3 4 ;

je výsledok 1 2 4 6 3 5 alebo 6 2 1 4 5 3. Stačí jedno riešenie.

### 815. O hre BOXES

20

o o o o o o Hra BOXES sa hrá na modrom papieri, na ktorom je  $5 \times 6$  bodov usporiadaných do tvaru obdĺžnika (viď obrázok). Hru hrajú dvaja hráči. V každom ťahu musí hráč spojiť dva dosiaľ nespojené susedné body horizontálnou alebo vertikálnou čiarou svojej farby. (Jeden hráč používa na tento účel čiernu a druhý bielu farbu.) Keď týmto ťahom utvorí BOX, t.j. útvar o---o, v ktorom sú všetky spojenia bodov jeho farby, ťahá ešte raz.

| |

o---o

Hra končí, keď sa už nedá vytvoriť žiaden BOX. Vyhráva hráč, ktorý urobil viacej BOXov. Situácia hry BOXES sa dá reprezentovať nasledujúcim spôsobom:

Situácia:	Reprezentácia:	Vstup pre počítač:
+ čierny      - biely	C čierny      B biely	
o---o+++o    o+++o    o	B   C   0   C   0	BC0C0
+	B   C   B   0   B   0	BCB0B0
o   o   o+++o---o    o	0   0   C   B   0	00CB0
+	B   C   B   B   B   0	BCBBB0
o   o+++o    o---o    o	0   C   0   B   0	0C0B0
	0   0   0   0   0   0	000000
o   o   o   o   o+++o	0   0   0   0   0   C	0000C
	0   0   0   0   C   C	0000CC
o   o   o   o   o+++o	0   0   0   0   C	0000C

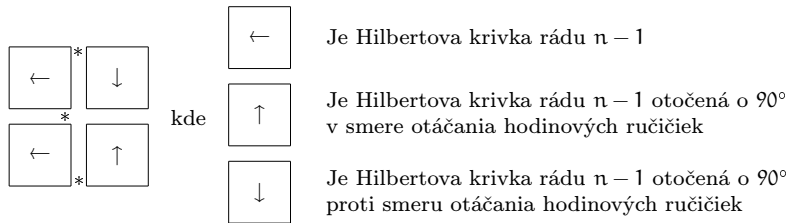
ÚLOHA: Napíšte program, ktorý načíta reprezentáciu určitej hernej situácie a rozhodne, či hra ešte pokračuje, alebo či už skončila a ktorý hráč vyhral (prípadne oznámi remízu).

### 821. O Hilbertovej krivke

20

Hilbertova<sup>7</sup> krivka rádu  $n > 1$  je spojitá lomená čiara, ktorá je definovaná spojením štyroch otočení Hilbertových kriviek rádu  $n - 1$  pomocou troch čiar, pričom Hilbertova krivka prvého rádu je na obrázku vpravo.

Všeobecne má Hilbertova krivka rádu  $n$  tvar:



a znak '\*' predstavuje čiaru spojenia.

Skúste si nakresliť Hilbertovu krivku druhého rádu, mali by ste dostať krivku ako na obrázku vľavo. Všimnite si, že Hilbertova krivka rádu  $n$  sa vždy zmestí do štvorčkovej siete s rozmermi  $(2^{n+1} - 1) \times (2^{n+1} - 1)$  políčok (na políčku môže byť hviezdička alebo medzera). Túto sieť nazvime H-sieť a definujme v nej súradnice políčok tak, že políčko v riadku  $x$  a v stĺpci  $y$  má súradnice  $[x, y]$  (ľavý horný roh H-siete má súradnice  $[1, 1]$ ).

ÚLOHA: Je daný rád Hilbertovej krivky  $n$  a súradnice dvoch políčok v H-sieti  $[p_x, p_y]$ ,  $[q_x, q_y]$ . Tieto políčka v H-sieti určujú obdĺžnik a to tak, že  $[p_x, p_y]$  je jeho ľavým horným a  $[q_x, q_y]$  jeho pravým dolným políčkom. Napíšte program, ktorý zobrazí časť Hilbertovej krivky, ktorá sa nachádza v určenom obdĺžniku. Napríklad vstup :  $n = 2$ ,  $[p_x, p_y] = [2, 2]$ ,  $[q_x, q_y] = [5, 6]$  výstup je na obrázku vpravo.

```

+- - - -+
!      *  !
!* *    * !
! *      !
!* *    * !
+- - - -+

```

### 822. O kružniciach

19

V rovine je daných  $n$  bodov so súradnicami  $[x_i, y_i]$ , ktoré ležia na nejakej kružnici so stredom v bode  $[0, 0]$ .

<sup>7</sup> David Hilbert, 1862–1948, nemecký matematik

ÚLOHA: Napište program, ktorý načíta  $n$ ,  $x_i$ ,  $y_i$ , pre  $1 \leq i \leq n$  a zistí, či je možné zostrojiť v rovine priamku prechádzajúcu bodom  $[0,0]$  takú, že všetky dané body  $[x_i, y_i]$  ležia v jednej, ňou určenej polrovine (t.j. priamka nerozdeľuje body na dve skupiny; všetky ležia „na jednej strane“).

### 823. O zásobovaní

[23]

V krajine je  $n$  miest očíslovaných  $1, 2, \dots, n$ , medzi ktorými je vybudovaná cestná sieť tak, že z každého mesta sa dá dostať do ľubovoľného iného (nemusí sa to však dať priamym spojením, t.j. cestou, ktorá neprechádza cez žiadne mesto). V krajine sa chystajú vybudovať centrálny sklad potravín, z ktorého by každé ráno vyštartovalo  $n - 1$  zásobovacích áut naložených potravinami do ostatných miest (každé do iného) a do ktorého by sa každý večer zásobovacie autá vrátili. Každá doprava však niečo stojí. O tejto predpokladáme, že je tým drahšia, čím dlhšiu cestu zásobovacie autá denne vykonajú.

ÚLOHA: Napište program, ktorý načíta počet miest  $n$ , maticu  $A[1..n, 1..n]$ , kde  $A[i, j] = 0$ , ak medzi mestami neexistuje priame spojenie, alebo  $A[i, j] = [\text{dĺžka priameho cestného spojenia medzi mestami } i \text{ a } j]$ , ak také spojenie existuje. Výstupom je číslo mesta, v ktorom treba vybudovať centrálny sklad, aby bolo zásobovanie krajiny čo najlacnejšie (t.j. aby bol minimálny súčet dĺžok ciest vykonaných zásobovacími autami).

Pozor! Tento príklad nie je ani náhodou podobný príkladu 813 o zaostalej krajine!

### 824. O balíčkoch

[25]

V podniku KRACH vyrábajú výrobky s hmotnosťou 1, 2, 3, 4, 5, 6, 7 alebo 8 kg. Od odberateľov prijímajú objednávky v tvare:

hmotnosť	1kg	2kg	3kg	4kg	5kg	6kg	7kg	8kg
počet kusov	$p_1$	$p_2$	$p_3$	$p_4$	$p_5$	$p_6$	$p_7$	$p_8$

a poštou im posielajú balíky s požadovaným počtom výrobkov. Pošta však doručuje len tie balíky, ktorých hmotnosť nepresahuje 8 kg. Preto si podnik KRACH u vás objednal program, ktorý načíta objednávku  $p[1..8]$  a navrhne, akým spôsobom treba výrobky baliť do balíkov, aby bol počet balíkov najmenší možný.

Napríklad pre  $p = 1, 2, 2, 0, 1, 0, 2, 0$  je jedno z možných riešení dať do 1. balíka 3 2 2; do 2. balíka 7; do 3. balíka 5 3; do 4. balíka 1 7. Celkovo sú potrebné 4 balíky.

### 825. Druhý príklad o hre BOXES

[21]

Spomeňte si na príklad 815 o hre BOXES. Iste ste sa pri jeho riešení potešili, že sa od vás nežiada kontrola korektnosti danej hernej situácie.

ÚLOHA: Napište program, ktorý načíta reprezentáciu určitej hernej situácie v hre BOXES. Zistí, či je korektná a ak áno, tak vypíše jeden z možných priebehov hry. Spôsob, akým sa priebeh hry zaznamená, si môžete ľubovoľne zvoliť, len dbajte na to, aby bol prehľadný a stručný.

### 831. O katastrofe vo východnej krajine

► [21]

Vo východnej krajine je  $n$  miest očíslovaných  $1, 2, \dots, n$ . Medzi všetkými mestami navzájom existovali spojenia a to buď priame, alebo nepriame (nepriame spojenie je spojenie dvoch miest cez konečný počet miest). Jedného pekného dňa sa východnou krajinou súčasne prehnali uragán, zemetrasenie, povodeň, hladomor a revolúcia, čo zapríčinilo, že sa krajina rozpadla na niekoľko izolovaných častí, medzi ktorými nie je žiadne spojenie. Zaujímavé by bolo zistiť, na koľko častí sa krajina rozpadla.

ÚLOHA: Je daný počet miest  $n$  východnej krajiny a  $k$  dvojíc miest  $(x_i, y_i)$ , kde  $1 \leq i \leq k$  a  $1 \leq x_i, y_i \leq n$ , medzi ktorými sa po katastrofe zachovalo priame spojenie (ak sa zachovalo priame spojenie mesta  $x$  s mestom  $y$ , zachovalo sa aj priame spojenie  $y$  s  $x$ ). Napište program, ktorý pre dané údaje zistí, na koľko izolovaných častí sa východná



krajina rozpadla. Za izolovanú časť považujte takú skupinu miest, že medzi každými dvomi mestami z tejto skupiny existuje spojenie (t.j. existujú mestá, cez ktoré sa môžu spojiť), a žiadne mesto z tejto skupiny nemá spojenie s mestom, ktoré nepatrí do tejto skupiny.

### 832. O dierach v doske

25

V ústave informatiky majú drevenú dosku zanedbateľnej hrúbky obdĺžnikového tvaru rozmerov  $n \times m$ . V doske je vyvŕtaných  $k$  bodových dier (t.j. dier zanedbateľnej veľkosti) so súradnicami  $[x_i, y_i]$ , kde  $1 \leq i \leq k$  (bod  $[0, 0]$  je umiestnený v ľavom dolnom rohu dosky, spodný okraj dosky určuje os  $x$  a ľavý okraj dosky určuje os  $y$ ).

ÚLOHA: Napíšte program, ktorý pre danú situáciu zistí rozmery maximálneho (čo do obsahu) obdĺžnika bez dier, ktorý sa dá z dosky vypilovať tak, že sa pili len rovnobežne s okrajmi dosky (t.j. v smere osí).

### 833. O spore v parlamente

► 25

Dve iniciatívne skupiny poslancov predložili v parlamente dva návrhy zákona. Prvá skupina predložila návrh A dĺžky  $n$  (v programátorskom ponímaní je návrh zákona jednorozmerné pole znakov danej dĺžky) a druhá skupina návrh B dĺžky  $m$ . Aby sa poslanci neháďali, dohodli sa, že z obidvoch návrhov vyškrtnú určité časti tak, aby boli návrhy po vyškrtnutí zhodné. Samozrejme im záleží na tom, aby škrtali čo najmenej.

ÚLOHA: Sú dané čísla  $n, m$  a polia znakov  $A[1..n]$ ,  $B[1..m]$ . Napíšte program, ktorý urobí návrh, ktoré znaky treba z polí vyškrtnúť tak, aby boli polia znakov po vyškrtnutí totožné. Snažte sa, aby bolo škrtaných znakov čo najmenej. Výstup môžete urobiť napríklad takto: budete predpokladať, že A a B neobsahujú zátvorky  $()$  a dáte do zátvoriek tie časti A a B, ktoré treba vyškrtnúť.

Keď A = 'moricko pisał basne v siestich rokoch' a B = 'horac sipel krasne v sestnastich rokoch'. Výstupom môže byť napríklad:

```
A = (m)or(i)c(ko) (p)i(sa)l (b)asne v s(i)estich rokoch
B = (h)or(a)c (s)i(pe)l (kr)asne v sest(nast)ich rokoch
pocet skrtani: 20
```

### 834. O vzbure v Hanibalovom tábore

► 30

Keď sa v roku 217 p.n.l. rozhodol kartáginský veliteľ Hanibal tiahnuť na Rím pochodom cez Alpy, jeho armáda sa vzbúrila a chcela iného veliteľa. Po krátkej hádke sa vojaci zhodli, že vojenskú prísahu zložia šľachticovi Hasdrubalovi a chystali sa k nemu vyslať skupinu poslov, v ktorej mal byť z každého práporu práve jeden vojak. Hanibal bol však bystrý človek. Podplatil otroka Sikima, aby presvedčil vzbúrených vojakov, že k Hasdrubalovi nemožno vyslať len tak hocijakých poslov. Keď vojaci začali rozmýšľať aký lesk by svojmu posolstvu dodali, Sikim ich nahovoril, aby vybrali takých poslov, ktorí budú mať navzájom rôzne mená. Kartáginci mali totiž početnú armádu s veľa prápormi, no ich kalendár mal málo mien. Vojaci sa dlhé hodiny snažili zostaviť takéto posolstvo, a keď sa im to nepodarilo, presvedčení, že je to zákrok nebies, vrátili sa s prosbou o odpustenie k Hanibalovi.

ÚLOHA: Napíšte program, ktorý načíta číslo  $n$ , prvky  $k$  množín  $m_1, m_2, \dots, m_k$ , ktoré obsahujú len prvky  $1, 2, \dots, n$  (t.j.  $m_i$  je podmnožina množiny  $\{1, 2, \dots, n\}$ ) a zistí, či je možné z každej množiny vybrať jeden prvok tak, aby vybrané prvky boli navzájom rôzne. V prípade, že to možné je, program musí vypísať aj výber prvkov.

### 835. O svetovej vojne

22

Na planéte Z sú štáty  $1, 2, \dots, n$ . Zvykom na tejto planéte je, že sa uzatvárajú vojenské dohody výlučne typu : „Ak štát  $x$  napadne štát  $y$ , potom my štát  $u$  napadneme štát  $v$ “.

ÚLOHA: Napíšte program, ktorý načíta počet štátov  $n$ ,  $k$  vojenských dohôd (t.j. štvoric  $(x_i, y_i, u_i, v_i)$ , kde  $1 \leq i \leq k$ ) a zistí, či existujú také dva štáty  $g$  a  $h$ , že ak  $g$  napadne  $h$ ,

tak na planéte Z vypukne svetová vojna (svetová vojna je stav, keď bojujú všetky štáty, t.j. keď každý štát niekoho napadol, alebo bol niekým napadnutý). V prípade, že také  $g$  a  $h$  existujú, program ich vypíše a okrem toho vypíše poradie dvojíc (útočník, obranca), v ktorom sa štáty budú napádať.

#### 841. O orientačnom behu

[21]

V okolí Bratislavy sa každoročne organizuje niekoľko orientačných behov. Pri orientačnom behu dostanú účastníci pred štartom poradie stanovišť, ktoré organizátori predtým v teréne vyznačili, a ich úlohou je prebehnúť cez všetky stanovištia v danom poradí. Pri pretekoch sa zvyčajne beží viac rôzne dlhých tratí, pričom trať a jej dĺžku organizátori určujú poradiem a počtom stanovišť, cez ktoré majú bežci prebehnúť. Za dĺžku trate sa považuje súčet priamych vzdialeností medzi stanovišťami, takže v skutočnosti bežci prebehnú trochu dlhšiu vzdialenosť (samozrejme za predpokladu, že nezablúdia). Na trať sa kladie požiadavka, že cez každé stanovište môže prechádzať len raz, s výnimkou štartovacieho a cieľového stanovišta - tie môžu byť totožné. (To však platí len pre trať a nie pre bežca. Ten môže prechádzať jedným stanovišťom aj viackrát, napríklad keď zablúdi, alebo keď sa jedno stanovište nachádza na ceste medzi dvoma inými.) Inak sa na trať nekladú žiadne požiadavky, takže sa môže napríklad aj „križovať“.

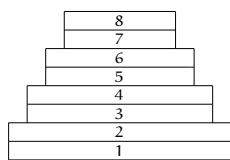
ÚLOHA: Napište program, ktorý načíta počet stanovišť  $n$ , súradnice stanovišť  $[x_i, y_i]$  pre  $1 \leq i \leq n$  (teda akúsi relatívnu zemepisnú dĺžku a zemepisnú šírku stanovišť) a navrhne (t.j. určí poradie stanovišť) najdlhšiu trať, akú možno pre dané stanovištia navrhnuť. (Výškové rozdiely sa pri výpočte vzdialenosti dvoch stanovišť zanedbávajú.)

Napríklad pre  $n = 4$  a stanovištia so súradnicami  $[x, y] = [0, 0], [2, 0], [10, 1], [10, -1]$  je výsledok 1, 3, 2, 4, 1.

#### 842. O Hanojských dvojvežiach

[M27]

V budhistickom kláštore v Hanoji majú tri veľké stĺpy. Na prvom stĺpe sú nastoknuté ťažké kamenné disky s otvorom uprostred. Tieto disky sú rôznych veľkostí (hmotností; u kameňa so stúpajúcim objemom rastie aj hmotnosť), pričom z každého druhu sú na stĺpe práve dva disky. Navyše sú tieto disky nastoknuté v takom poradí, aby bol vždy menší nad väčším, prípadne rovnako veľkým (lebo menší by sa pod väčším rozdrvil) a sú očíslované od najspodnejšieho ( $= 1$ ) po najvrchnejší ( $= 2n$ , kde  $n$  je počet druhov diskov).



Příklad pre  $N = 4$

Mnísi v tomto kláštore volajú takéto usporiadanie diskov Hanojská dvojveža a veria, že keď sa im podarí preložiť disky z prvého stĺpu na tretí, a to v takom poradí ako boli pôvodne na prvom stĺpe, nastane koniec sveta. Našťastie pre náš svet to podľa svojho náboženstva nemôžu robiť hocijako, ale vždy môžu preložiť len jeden disk z niektorého stĺpu na iný (takže na preloženie musia nutne použiť druhý stĺp). Pri prekladaní sú ďalej obmedzení tým, že nemôžu klást väčší disk na menší, lebo menší by sa pod väčším rozdrvil (klást na seba rovnako

veľké disky je však možné). Preto sa mnísi už mnohé roky snažia vypočítať, ako majú disky poprekladať, aby im to vyšlo<sup>8</sup>.

ÚLOHA: Napište program, ktorý načíta počet druhov diskov  $n$  a vygeneruje postupnosť preložení tvaru „zo stĺpa č.  $x$  na stĺp číslo  $y$  prenes disk číslo  $z$ “, po vykonaní ktorej by sme preložili dvojvežu z prvého stĺpu na tretí. Snažte sa, aby bol počet preložení minimálny a pokúste sa určiť vzorec pre počet preložení, ktoré navrhne váš program v závislosti od  $n$ .

<sup>8</sup> Pôvodný hlavolam vymyslel francúzsky matematik Edouard Lucas v roku 1883. Opriadol ho príbehom o Brahmovej veži, ktorá mala 64 rôzne veľkých diskov z rýdzeho zlata. Disky sa navliekali na tri diamantové ihlice. Na počiatku ich Boh umiestnil na prvú ihlicu a skupina mníchov ich má presunúť na tretiu, podľa rovnakých pravidiel ako v tejto úlohe. Keď sa im to podarí svet zanikne. E. Lucas, *Récreations mathématiques*, 4 zväzky, Ganthier-Villars, Paris 1891–1894, Reprint: Albert Blanchard, Paris 1960, zv. 3 s. 55–59

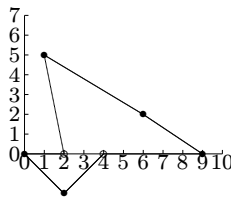
**843. O mnohouholníku**

25

Máme dané  $n$  a ľubovoľný  $n$ -uholník, ktorý je daný vymenovaním súradníc svojich vrcholov tak, ako po sebe nasledujú proti smeru chodu hodinových ručičiek počnúc vrcholom s najmenšou  $x$ -ovou súradnicou.

ÚLOHA: Napíšte program, ktorý z vrcholov daného (aj nekonvexného)  $n$ -uholníka vyberie tie vrcholy, ktoré tvoria konvexný mnohouholník s tou vlastnosťou, že obsahuje („pokrýva“) daný nekonvexný  $n$ -uholník. Výstupom programu je postupnosť poradí vybraných vrcholov z daného nekonvexného  $n$ -uholníka (táto postupnosť je rastúca).

Napríklad pre  $n = 7$  a vrcholy  $[0, 0]$ ,  $[2, -2]$ ,  $[4, 0]$ ,  $[9, 0]$ ,  $[6, 2]$ ,  $[1, 5]$ ,  $[2, 0]$  je výsledok 1, 2, 4, 5, 6.

**844. O prekladateľovi**

19

Napíšte program, ktorý načíta reťazec znakov, ktorý zodpovedá korektnému zápisu čísla od nuly po 999 999 v slovenskom alebo anglickom jazyku. Rozpozná, v ktorom jazyku je dané číslo zapísané a urobí preklad čísla do druhého jazyka. Predtým než začnete robiť samotný program, nezabudnite si presne definovať, aký reťazec pokladáte za korektný zápis čísla v anglickom a slovenskom jazyku.

Pokiaľ vám angličtina nevyhovuje, môžete použiť iný svetový jazyk (ruštinu však neodporúčame, iba ak by ste mali počítať s kódom aj pre azbuku).

Príklad: *Two hundred and fourteen thousand five hundred and twenty-seven* preložíme na *Dvestostrnast tisíc patsto dvadsatsedem a nula na zero*.

S medzerami a pomlčkami si hlavu netrápajte, iba ak by vás úloha zaujala a ak by sa vám chcelo otravovať vašich slovenčinárov a angličtinárov. Takisto nemusíte uvažovať mäčkene, ale ak ste maniaci, môžete sa o to pokúsiť.

**845. Miškova Super Úloha**

M50

S touto úlohou sa môžete dennodenne stretnúť v továrenskej výrobe, takže je vám už asi jasné, že pôjde o ťažkú a zaujímavú úlohu. Keď sa vyrába nejaký výrobok plošného tvaru (napríklad podošva, rôzne výrezy z plechu), väčšinou sa vyrezáva z obdĺžnikového kusu plošného materiálu (koža, plech). Samozrejme, ak chceme ušetriť, musí byť obdĺžnikový kus materiálu čo najmenší. Zovšeobecnenie a zjednodušenie vedie k nasledujúcej úlohe:

ÚLOHA: Je dané  $n$  a konvexný  $n$ -uholník.  $n$ -uholník je daný vymenovaním súradníc vrcholov, tak ako po sebe nasledujú proti smeru chodu hodinových ručičiek počnúc vrcholom s najmenšou  $x$ -ovou súradnicou (teda tak, ako v úlohe 843). Napíšte program, ktorý vypočíta súradnice štyroch vrcholov obdĺžnika s čo najmenším obsahom, ktorý obsahuje daný  $n$ -uholník.

**911. O Mersenových<sup>9</sup> číslach**

18

Napíšte program, ktorý pre dané prirodzené čísla  $m$  a  $n$ ,  $m \leq n$ , nájde nezáporné celé čísla  $k_1, k_2, \dots, k_n$  (t.j. jedno riešenie) také, že číslo  $2^{k_1} + 2^{k_2} + \dots + 2^{k_n}$  bude deliteľné  $m$ -tým Mersenovým číslom (t.j. číslom  $2^m - 1$ ). Príklad: Pre  $m = n = 3$  môže byť výsledok napríklad: 3, 1, 2, lebo  $2^3 - 1 = 7$  delí číslo  $2^3 + 2^1 + 2^2 = 14$ .

V prípade, že sa pre určité  $m$  a  $n$  úloha nedá riešiť, program o tom musí vypísať správu.

**912. O výstavbe mosta**

21

Vládca súostrovia Kiribati sa chcel zapáčiť svojmu ľudu a sľúbil mu postaviť most. Teraz musí svoj sľub splniť, ale chcel by, aby ho to stálo čo najmenej peňazí. Keďže tvar

<sup>9</sup> Mersenne Marin, 1588–1648, francúzsky mních, matematik a filozof.

morského dna medzi ostrovmi je na Kiribati približne rovnaký, množstvo peňazí potrebných na stavbu je priamo úmerné dĺžke mosta. O súostroví Kiribati sa vie, že žiaden ostrov neobsahuje zálivy (ostrovy sú konvexné útvary) ani lagúnu (vnútorné more).

ÚLOHA: V matici  $A$  rozmerov  $m \times n$  núl a jednotiek je zaznamenaná mapa súostrovia Kiribati ( $0 \dots$  more,  $1 \dots$  pevnina). Navrhňte program, ktorý vypíše túto mapu, pričom na nej vyznačí miesto, kde má vládca nechať postaviť sľúbený most. Musí ho to však stáť čo najmenej peňazí, t.j. dĺžka vyznačeného mosta musí byť najmenšia možná. Most na mape vyznačíte tak, že na mape prepíšete niektoré nuly (more) napríklad na dvojky (alebo na nejaké iné sympatické čísla predstavujúce most). Aby toto vyznačenie bolo mostom, musia byť splnené nasledujúce podmienky:

- Každé políčko mapy predstavujúce most susedí buď (a to celou stranou, nie rohom!) s políčkom predstavujúcim ostrov (s jednotkou) a s jedným políčkom predstavujúcim most, alebo susedí s práve dvoma políčkami predstavujúcimi most, alebo susedí s dvoma políčkami predstavujúcimi rôzne ostrovy.
- Most musí spájať dva rôzne ostrovy. (V prípade nejasnosti definície sa riadte intuitívnou predstavou mosta.)

Dĺžkou takto vyznačeného mosta je počet políčok, ktoré ho predstavujú.

### 913. O koláči

[21]

Na obdĺžnikovom plechu upiekli (mimochodom nie veľmi dobrý) koláč rozmerov  $m \times n$  (rozмеры koláča a plechu boli rovnaké) a rozrežali ho na  $k$  kusov. Všetky kusy boli obdĺžnikového tvaru, ale ich rozмеры sa navzájom značne líšili. Hostia sadli ku stolu s koláčom, ale keďže sa veľmi nemali k jedeniu, začali si krátiť dlhú chvíľu skladaním nakrájaných kusov koláča do pôvodného tvaru, teda do tvaru, aký mal na plechu. Pritom im najväčšie problémy robilo zistenie rozmerov plechu, na ktorom sa koláč piekol. Po dlhej a namáhavej práci sa dohodli na tom, že táto úloha nie je súca pre nich, ale pre účastníkov KSP.

ÚLOHA: Je daný počet  $k$  nakrájaných kusov koláča obdĺžnikového tvaru rozmerov  $x_i, y_i$ ,  $1 \leq i \leq k$ ). Napíšte program, ktorý zistí, aké mohli byť rozмеры plechu  $m \times n$ , na ktorom bol daný koláč upečený (stačí jedno riešenie). Všetky rozмеры sú celočíselné.

Výsledkom programu sú len rozмеры  $m$  a  $n$ , nežiada sa nájsť rozloženie daných kusov na plechu! Pri pečení je celý plech pokrytý cestom.

### 914. O Burundjskej abecede

► [21]

Keď sa obyvatelia Burundi zbavili belgických kolonizátorov, boli už natoľko poznačení ich nadvládou, že od nich prevzali francúzsky jazyk a abecedu. V návale nacionalizmu a protibelgických vášní sa rozhodli, že francúzku abecedu nahradia vlastnou, burundjskou. Aby si pritom nenarobili veľa problémov, vytvorili svoju abecedu z francúzskej tak, že iba zmenili poradie jej písmen.

ÚLOHA: Napíšte program, ktorý na vstupe prijíma  $k$  slov usporiadaných podľa burundjskej abecedy a vypíše jedno možné poradie písmen burundjskej abecedy (vypíše iba poradie písmen, ktoré sa nachádzali v slovách).

Napríklad pre slová **gda**, **gab**, **qek**, **da** je jedno poradie písmen v abecede **g, q, d, a, k, b, e**.

### 915. O Froscale

[25]

Froscale je jazyk sprevádzajúci 41. ročník matematickej olympiády, kategória P. Bližší opis tohto jazyka je uvedený nižšie.

ÚLOHA: Navrhňte prekladač tohto jazyka do Pascalu, t.j. program, ktorý načíta program vo Froscale a vytvorí k nemu príslušný ekvivalent v Pascale.

Samotný prekladač môže byť napísaný v ľubovoľnom prípustnom jazyku.

Riešiteľom, ktorí Pascal neovládajú: Urobiť prekladač Froscale do Basicu alebo do C by bolo oveľa ťažšie ako sa pozrieť do múdrej knihy, napríklad [32], ako Pascal vyzerá.

Ak by vás však študovanie Pascalu odradilo od riešenia tejto úlohy, ponechávame vám možnosť definovať si analogický jazyk, pracujúci s frontom, podobný Basicu alebo C (teda Frosic alebo Froc) a naprogramovať prekladač z tohto jazyka do Basicu alebo C.

#### Froscal — Frontový Pascal

Jazyk bez procedúr, premenných, syntaxou pripomínajúcou Pascal (aj so skokmi) a s jedinou pamäťovou štruktúrou — frontom — to je **Froscal**. Front je dátová štruktúra s premenlivou veľkosťou (nie nepodobná zásobníku), u ktorej sa vkladá na jeden koniec a vyberá z druhého. Presnejšie: Froscal je Pascal bez premenných (a teda aj bez priradenia), s **while** a **repeat** cyklami, príkazmi **case** a **if** a skokom **goto**. Má dve fiktívne premenné *INP* a *TOP* typu char a skrytú premennú front znakov. *INP* je práve čítaný znak zo vstupu a *TOP* je znak na začiatku frontu. Ďalšie príkazy: *NEXT* načíta nový znak zo vstupu do fiktívnej premennej *INP*, *PUT*(znak) alebo *PUT*(premenná) uloží znak, alebo hodnotu premennej *INP* alebo *TOP* na koniec frontu, *GET* vyhodí prvý prvok z frontu. Príkaz *write* má Pascalovskú definíciu, t.j. môže sa vypísať znak, premenná, reťazec alebo postupnosť takýchto výrazov. Špeciálnym znakom na vstupe a vo fronte je znak '?', ktorý na vstupe znamená presiahnutie vstupného slova a vo fronte znamená prázdny front. Ak *INP* = '?' (alebo *TOP* = '?'), potom príkaz *NEXT* (alebo *GET*) nič nespraví. Ani *PUT*('?') nič nevykoná.

Program pracuje tak, že na vstupe má slovo nad dopredu presne určenou abecedou (podmnožinou vypísateľných znakov) a na výstupe (*write*) môže vypísať riešenie problému. Testovať korektnosť vstupnej abecedy nie je nutné; ak je špecifikovaný formát vstupu, tak ho netreba testovať. Do frontu ukladať, testovať a vypisovať možno ľubovoľné vypísateľné znaky. Na začiatku práce programu je front prázdny (t.j. *TOP* = '?') a premenná *INP* obsahuje prvý znak zo vstupu.

Príklad: Program, ktorý má na vstupe slovo v tvare  $w_1\#w_2$ , kde  $w_1$  a  $w_2$  sú slová nad abecedou **a b c d**. Treba otestovať, či sa  $w_1 = w_2$ .

```
program TEST;
label 13;
begin
  { front síce vyprázdniť netreba, ale na precvičenie }
  while TOP <> '?' do GET;
  while not((INP = '#') or (INP = '?')) do
    begin { ešte sme vo  $w_1$ , treba uložiť }
      PUT(INP); NEXT
    end;
  if INP <> '#' then goto 13; { slovo má tvar  $w_1$  bez '#' }
  NEXT;
  while ((INP <> '?') and (TOP = INP)) do
    begin { ďalší znak sa rovná, ideme ďalej }
      GET; NEXT
    end;
  if ((INP = TOP) and (INP = '?')) then write('ANO')
  else 13: write('NIE')
end.
```

Orientačná gramatika jazyka Froscal:

```
<program> = program <meno>; [(dekl. skokov)] <telo programu>
<dekl. skokov> = label <skok>[, <skok>]*;
<skok> = <celé číslo>
<telo programu> = begin <príkazy> end.
```

```

⟨príkazy⟩ = ⟨nič⟩ | ⟨príkaz⟩[; ⟨príkazy⟩]
⟨príkaz⟩ = [(⟨skok⟩)* ⟨elem. príkaz⟩]
⟨elem. príkaz⟩ = begin ⟨príkazy⟩ end
| if ⟨podm⟩ then ⟨príkaz⟩[ else ⟨príkaz⟩]
| case ⟨výraz⟩ of
    (⟨selektor⟩: ⟨príkaz⟩)*
    [ else ⟨príkaz⟩]
end
| while ⟨podm⟩ do ⟨príkaz⟩
| repeat ⟨príkazy⟩ until ⟨podm⟩
| goto ⟨skok⟩
| write((write výraz)[, ⟨i>write výraz⟩])*
| PUT((výraz))
| GET
| NEXT
⟨podm⟩ = ⟨relácia⟩
| (⟨podm⟩) and (⟨podm⟩)
| (⟨podm⟩) or (⟨podm⟩)
| not(⟨podm⟩)
⟨relácia⟩ = ⟨výraz⟩⟨relačný znak⟩⟨výraz⟩
⟨relačný znak⟩ =
    = | < | <= | > | >= | <> | in
⟨výraz⟩ = INP | TOP | ⟨znak. konšt.⟩ | ⟨interval znakov⟩ | ⟨množina znakov⟩
⟨i>write výraz⟩ = INP | TOP | ⟨znak. konšt.⟩ | ⟨reťazec⟩
⟨poznámka⟩ = { ⟨text bez '⟩'⟩ }

```

Poznámka môže byť kdekoľvek rovnako ako v Pascale

## 921. O Kanárskych poliach

► 21

Je dané pole  $A[0..n]$  celých čísel a kladné celé číslo  $k$ . Nech váš program rýchlo preusporiada toto pole tak, aby pre každé  $i$  platilo:  $a_i \geq a_{k-i+j}$  pre  $1 \leq j \leq k$  (ak  $a_{k-i+j}$  existuje).

## 922. O Kiribatských daniach

21

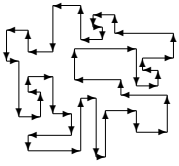
Vládca súostrovia Kiribati sa rozhodol vyrúbiť nové dane z rybolovu. Keďže bol múdry a spravodlivý, daň pre jednotlivé ostrovy chcel stanoviť podľa veľkosti ich pobrežia (čím viac pobrežia, tým viac možností lovu rýb). Dal si preto pre každý ostrov súostrovia vyhotoviť na štvorcový papier mapu, na ktorej zafarbené políčko predstavovalo pevninu a nezafarbené políčko more. Veľkosť pobrežia ostrova stanovil vládca ako počet zafarbených políčk na jeho mape, ktoré sa aspoň jedným rohom dotýkajú nezafarbeného políčka. Daňovým úradníkom neostalo nič iné, ako sa pustiť do práce podľa vládcových pokynov. Pot sa im lial z čela a preklínali chvíľu, keď sa odmietli zúčastniť programátorského školenia za oceánom.

ÚLOHA: Napište program, ktorý načíta rozmery mapy  $m$  (počet riadkov) a  $n$  (počet stĺpcov), mapu (jedného!!!) ostrova reprezentovanú maticou  $A[1..m, 1..n]$  núl a jednotiek (nula ... nezafarbené políčko ... more, jednotka ... zafarbené políčko ... pevnina) a vypočíta veľkosť pobrežia daného ostrova (podľa vládцovej definície).

Ostrov, môže mať aj zálivy, ale nemôže mať vnútorné more. Políčka pevniny susedia stranou alebo rohom.

## 923. O opitom zlatokopovi

► 25



Na Vyšnej Klondike sa objavilo zlato. Masy nezamestnaných Slovákov sa hrnú na Vyšnú Klondiku vykolikovať si svoju parcelku. Medzi prvými sa za vidinou ľahkého zbohatnutia vybral zlatokop Stano, zvaný Santo. Posilnený miestnou ohnivou vodou by na miesto ani nedošiel, nebyť jeho verného kamaráta Boba, zvaného Dlhý Banto. Na mieste Santo odborným okom (spod borovice) ohodnotil okolie a povelmi riadil triezveho Banta pri kolikovaní (Doprava! Krok! Ešte krok! Doľava! Krok! Ešte krok! Ešte krok! atď.). Ťažkosti nastali až dolu v Dolnom Kelčove v registračnom stredisku, kde si chceli zaregistrovať svoju parcelu. Jedna z najdôležitejších koloniek znela: „Plocha parcely: .....“. Spomenúť si však vedeli (presnejšie Banto si spomenul) iba na postupnosť Santových príkazov. Santo vyhlásil, že zistiť z toho plochu parcely musí byť hračka už aj preto, že o Bantovi je známe, že jeho krok meria 1 meter.

V súčasnosti už niekoľko dní sedia v predizbe registračnej miestnosti a na sakramentsky malom (!) kúsku papiera sa snažia vypočítať plochu svojej parcely, kým ich niekto nepredbehne.

ÚLOHA: Predbehnite ich, a napíšte program, ktorý rýchlo (a v čo najmenšej pamäti) vypočíta plochu Santovej a Bantovej parcely. Na vstupe dostane postupnosť Santových príkazov, pre jednoduchosť zaznamenaných v tvare čísel:

- 1 ... (otoč sa o  $90^\circ$ ) Doľava!
- 2 ... (otoč sa o  $90^\circ$ ) Doprava!
- 3 n ... Prejdi n krokov!

Napríklad postupnosť 3 1 2 1 1 3 2 1 3 1 1 3 2 zodpovedá postupnosti príkazov: Prejdi 1 krok! Doprava! Doľava! Doľava! Prejdi 2 kroky! Doľava! Prejdi 1 krok! Doľava! Prejdi 2 kroky! Teda zodpovedajúca parcela by mala tvar obdĺžnika  $2\text{m} \times 1\text{m}$ .

Dôležité! Z ich rozhovoru sa zistilo, že Banto skončil kolikovanie na tom istom mieste, kde začínal, a pritom nikdy neprekrížil svoju cestu.

## 924. O Burundijskej železnici

► 27

Keď belgickí kolonizátori opustili Burundi, nezostal tam po nich kameň na kameni. Burundijčania začínali všetko budovať odznova a v prvom rade sa vrhli na výstavbu železničnej siete. Tá mala podľa ich predstáv spájať všetky významné dediny (na tomto mieste by bolo vhodné poznamenať, že burundijské dediny boli síce veľké, ale bezmenné skupiny palmových príbytkov, a preto ich stavitelia železnice museli očíslovať číslami od 1 po  $n$ ). Lenže na koľajnice potrebovali kvalitné železo, a to bolo v Burundi veľmi vzácné. Zostavili preto maticu  $A[1..n, 1..n]$ , do ktorej na políčko  $A[i, j]$  napísali, aké množstvo železa je potrebné na vybudovanie železničného spojenia z dediny s číslom  $i$  do dediny s číslom  $j$  (Táto ich matica je symetrická). Ďalej si však už ani pri najlepšej vôli poradiť nevedeli, lebo belgickí kolonizátori ich nenaučili programovať. Železnicu nakoniec postavili len medzi niektorými dedinami, v dôsledku čoho vznikla v Burundi nenávisť medzi dvoma najpočetnejšími kmeňmi a skončilo sa to krvavou občianskou vojnou, ktorá trvá podnes.

ÚLOHA: Napíšte program, ktorý pre dané  $n$  a maticu  $A$  (s vyššie uvedeným významom) navrhne dvojice Burundijských dedín, medzi ktorými treba vybudovať železničné spojenie tak, aby sa z každej dediny dalo železnicou dostať do ľubovoľnej inej a aby množstvo železa potrebného na stavbu bolo najmenšie.

## 925. Tretí príklad o hre BOXES

25

Janko a Marienka sa hrali známu hru BOXES. Hra BOXES sa hrá na papieri, na ktorom je  $m \times n$  bodov usporiadaných do tvaru obdĺžnika.

V každom ťahu hráči spájajú dva dosiaľ nespojené susedné body horizontálnou alebo vertikálnou čiarou. Pravidlá samotnej hry sú pre nás teraz nepodstatné (kto by ich chcel poznať, nech si pozrie príklad 815).

Situácia:	Reprezentácia:	Vstup pre počítač:
o---o---o o---o o	1 1 0 1 0	'11010'
	1 1 1 0 1 0	'111010'
o o o---o---o o	0 0 1 1 0	'00110'
	1 1 1 1 1 0	'111110'
o o---o o---o---o	0 1 0 1 1	'01011'
	0 0 0 1 1 1	'000111'
o o o o---o	0 0 0 0 1	'00001'
	0 0 0 1 1 1	'000111'
o o o o---o---o	0 0 0 1 1	'00011'

Takže Janko s Marienkou sa hrali túto hru a keď už Janka delili od víťazstva len dva ťahy, Marienka, vidiac, že ak niečo nevymyslí, prehrá, sa opýtala: „Janko a koľko sme my vlastne nakreslili obdĺžnikov? Tých je tak veľa, že ja by som to veru nezrákala.“ Janko chcel samozrejme ihneď porátať všetky obdĺžniky a tak rátať, rátať, až obe deti zavolala ich mamička na obed. „A kto vlastne vyhral?“ opýtala sa mamička, keď sadli ku stolu. „Ale, nedohrali sme“, povedal cez slzy Janko a Marienka sa s chuťou pustila do fazuľovej polievky.

ÚLOHA: Napište program, ktorý načíta reprezentáciu určitej hernej situácie hry BOXES a spočíta, koľko obdĺžnikov hráči nakreslili.

Napríklad pre vyššie uvedenú situáciu by mal program zistiť, že hráči nakreslili osem obdĺžnikov.

### 931. O Santovej rúre

100

Santo a Banto úspešne vypočítali plochu svojej parcelky na Vyšnej Klondike a nič im už nebráni začať v ťažbe zlata. Ba predsa! Pri ryžovaní nutne potrebujú dostatok vody a Klondika je veľmi suchý kraj. Najbližší prameň sa nachádza dosť ďaleko od ich parcelky, a preto si v Dolnom Kelčove vybavili ešte aj povolenie na výstavbu podzemného vodovodného potrubia. Po úpornej práci v drsných podmienkach sa im podarilo vykopať výkop na každom mieste 1 meter hlboký a priamo vedúci k prameňu. Do tohto výkopu teraz potrebujú položiť rúru tak, aby po zasypaní výkopu netrčala nad povrch zeme. Klondika je však hornatá krajina, v dôsledku čoho sa rúra bude musieť občas rozrezať a zvariť, aby sa zalomila podľa stúpania alebo klesania terénu a nešla hlbšie ako 1m pod zem a zároveň nevyšla nad povrch zeme.

Keďže Santo je od prírody lenivý (a rezanie a zváranie rúr je ťažká práca), prikázal Bantovi, aby najprv vypočítal minimálny počet miest, na ktorých treba rúru rozrezať a zvariť, aby vedel, čo ho čaká.

Banto prešiel popri výkope a starostlivo si zaznačil každé miesto, kde sa mení sklon terénu, ako i miesto, kde výkop začína a miesto, kde výkop končí. Tieto miesta si očísloval číslami  $1, 2, \dots, n$  v takom poradí, v akom po sebe nasledujú, keď sa prechádza popri výkope od začiatku po koniec. Pre každé vyznačené miesto (očíslované číslom  $i$ ) si tiež vypočítal  $y_i$  - výškový rozdiel s miestom, kde výkop začína a taktiež  $x_i$  - vzdialenosť tohto vyznačeného miesta od začiatku výkopu. Teraz Banto smutne sedí a nevie, čo si počítať.

ÚLOHA: Pomôžte Bantovi! Napište program, ktorý pre dané  $n$ , vzdialenosti  $x_1, x_2, \dots, x_n$  a výškové rozdiely  $y_1, y_2, \dots, y_n$  vypočíta minimálny počet zalomení Santovej a Bantovej rúry. Upresnenie:  $y_1 = 0$ , ostatné hodnoty  $y_i$  sú buď záporné (ak sa miesto  $i$  nachádza pod úrovňou miesta 1), kladné (ak sa miesto  $i$  nachádza nad úrovňou miesta 1), alebo rovné nule (ak sa miesto  $i$  nachádza na tej istej úrovni). Uvedomte si pritom, že na



začiatku i na konci musí potrubie trčať zo zeme, teda počiatočný i koncový bod sú dané presne. Hrúbku rúry považujte za zanedbateľnú.

### 932. O Kiribatskom pakovači

[20]

Vládca súostrovia Kiribati sa rozhodol vybudovať historický archív, ktorý by pretrval dlhé-predlhé veky. Okrem iného sa v tomto archíve mali uchovávať aj mapy súostrovia Kiribati ako aj viacerých spriaznených súostroví. Týchto máp je však dosť veľa a navyše častá sopečná činnosť núti kartografov vytvárať stále novšie mapy. Preto po dohode so svojimi poradcami vydal nariadenie o konkurze na najlepší návrh, ako nejako zhustene uchovávať všetky tieto mapy.

ÚLOHA: Mapa je reprezentovaná maticou obsahujúcou iba nuly a jednotky, pričom jednotiek je dosť málo (menej ako 15%). Navrhните pakovač (a k nemu rozpakovač), ktorý danú maticu spakuje do vami navrhnutej zhustenej formy.

### 933. Telegram Bielym Légiam

[18]

patrame po spanieloch stop boli v amerike zmizli bez stopy stop asi u indianov stop indiani maju tenke koly stop možno ich umucili stop tych spanielov stop indianske hliadky sa skladaju z en bojovníkov stop keď hliadka chyti bielu tvar zmera ju a zakoduju do en krat tri matice stop kazdy bojovník robi jeden riadok a dava zajatcovi kody jedna dva a tri stop teda medicinman kmena dostane s kazdym zajatcom aj tabulku en krat tri v ktorej su tieto kody stop on potom spočíta ci zajatca budu mucit alebo ho pustia lebo nemozu vsetkych mucit lebo maju tenke koly stop vie sa len ze medicinman si k tabulke co mu prinesie hliadka prilozí este jeden riadok v ktorom je v prostrednom teda druhom stlpci diera stop velky manitou dovolil robit s tabulkou len dve veci stop mozu zrotovat hociktory riadok v lubovolnom smere stop mozu tiez posunut do diery cislo ktore je nad nou alebo pod nou stop tym sa samozrejme posunie aj diera stop ak sa da dosiahnut ze v kazdom povodnom riadku su cisla jedna dva a tri v tomto poradí a diera sa vrati na povodne miesto zajatca pustia lebo by im zlomil kol stop inak ma smolu stop chytili sme indiansku spojku stop ma so sebou vsetky tabulky stop pomozte nam zistit kolko spanielov umucili a kolki prezili stop díky stop

ÚLOHA: Keď dostali Biele Légie tento telegram, hneď im bolo jasné, že tých tabuliek bude veľa a ručne to nezrátajú. Nevedia však programovať, preto nás požiadali, aby sme pre nich urobili program. Program má na vstupe maticu  $A[1..n, 1..3]$ , ktorá má v každom riadku čísla 1, 2, 3 (v ľubovoľnom poradí). Výstupom má byť reťazec PREŽIL, ak zajatec s touto tabulkou prežil, alebo GAME OVER, ak ho Indiáni umučili. Naprogramujte tento postup.

### 934. O občianskej vojne v Burundi

[28]

V príklade 924 sa dejiny Burundi skončili krvavou občianskou vojnou medzi dvoma najpočetnejšími kmeňmi. Jeden z nich, kmeň Zest, obýval západnú časť Burundi a vyzýval bohov, ktorých symbolom je západ slnka. Druhý, kmeň Vast, obýval východnú časť krajiny a vyzýval bohov, ktorých symbolom je východ slnka.

Medzi ich územia sa tiahla zo severu na juh kľukatá hranica cez hustý prales, obývaný iba bojovníkmi toho-ktorého kmeňa. Po dlhých týždňoch krvavej vojny, v ktorej ani jedna strana nemohla získať prevahu nad druhou, vymyslel Veľký Šaman kmeňa Zest, aby sa najodvážnejší z odvážnych presunuli do niektorých dvoch zlomových bodov hranice (medzi ktorými leží iba nepriateľské územie), presekali sa priamo cez toto územie a po celej čiare, aby zapálili kúžlo. To ráno dymom zatieni vychádzajúce slnko a bojovníci kmeňa Vast sa na odrezanom území vzdajú v domnení, že ich bohovia opustili. Ako povedal, tak sa aj stalo. V noci najodvážnejší bojovníci presekali rovnú čiaru cez súperove územie, zapálili kúžlo a ráno na odrezanom území len zbierali zničených nepriateľov (ktorých potom kruto mučili).

Ich akciu však sledoval Veľký Šaman kmeňa Vast a poobede im odpovedali podobnou akciou, ktorá odrezala kus územia kmeňa Zest a kúzlom im zatienila zapadajúce slnko. Odvtedy sa vojna zmenila na neskôr nazývanú „Dymovú vojnu“: vždy cez noc kmeň Zest odrezal jednu časť územia kmeňa Vast, načo cez deň kmeň Vast odrezal jednu časť územia kmeňa Zest. Po niekoľkých takýchto útokoch si šamani uvedomili, že nech ich bojovníci bojujú akokoľvek, výsledok vojny to neovplyvní. Preto začali smerovať svoje útoky tak, aby sa vojna čo najskôr skončila.

ÚLOHA:

- Čo viedlo šamanov k tvrdeniu, že výsledok vojny nezmenia?
- Nech polia  $X[1..n]$  a  $Y[1..n]$  (pričom  $Y[i-1] < Y[i]$  pre každé  $2 \leq i \leq n$ ) predstavujú súradnice hranice. Navrhňte postupnosť útokov v tvare „Zest i j, Vast k l, ...“ tak, aby bola vojna trvala čo najkratšie.

### 935. Zobali vrabce zobali, konvexné obaly...

► [23]

Ku každej množine bodov v rovine existuje jej konvexný obal, teda najmenší konvexný mnohoúhelník obsahujúci dané body. Záhradkársky sa konvexný obal získa tak, že sa dané body omotajú špagátom a body, okolo ktorých sa špagát napína, tvoria konvexný obal danej množiny bodov.

- Máme dané dva disjunktné konvexné obaly dvoch množín bodov v rovine (t.j. ich konvexné obaly sa neprekrývajú). Oba sú dané vymenovaním súradníc svojich vrcholov v smere hodinových ručičiek. Napíšte čo najrýchlejší program, ktorý zistí konvexný obal zjednotenia týchto množín.
- V poliach  $X[1..n]$  a  $Y[1..n]$  máme uložené súradnice  $n$  bodov v rovine (poradie ich uloženia je náhodné). S využitím riešenia úlohy a) zistíte ich konvexný obal.

### 941. O tanečnom súbore Hatla-Patla

[21]

Tanečný súbor Hatla-Patla sa preslávil ktorom tanečníci nastúpia do radu a postupne polovica z nich predvedie zvláštny tanečný motív, zvaný Patla.

Počas tanca každý tanečník buď stojí a poskakuje na mieste obrátený tvárou k publiku, buď leží na nejakom inom tanečníkovi a tleska, alebo leží na zemi pod nejakým iným tanečníkom a do rytmu búcha hlavou o zem. Na začiatku sú všetci tanečníci v prvej polohe, t.j. stoja a poskakujú. Potom začnú vybraní tanečníci predvádzať Patlu a ak sa tanec vydarí všetci tanečníci ležia: prvá polovica na zemi a druhá polovica na tanečníkoch z prvej polovice. Patla spočíva v tom, že sa určený tanečník otočí k niektorému zo svojich susedov (podľa toho rozlišujeme Ľavú Patlu a Pravú Patlu), preskočí dvoch tanečníkov, nasledujúceho tanečníka zvalí na zem, ľahne si na neho a tleska (pričom zvalený tanečník začne búchať hlavou o zem). Preskok môže mať dve podoby: preskakovaní tanečníci obaja ležia alebo stoja. Ak ležia, tak sa preskakujú celkom ľahko, ak stoja, musia obaja urobiť podrep a vykriknúť „Patla!“ (pričom sa tanečník pri skoku môže odraziť od ich chrbtov). Ak by tanečník nemohol urobiť ani jednu verziu týchto dvoch preskokov, alebo by po preskoku nemal koho zvaliť na zem, Patla by sa mu nepodarila a celý tanec by sa pokazil.

ÚLOHA: Napíšte program, ktorý pre daný počet tanečníkov  $n$  vypíše, kedy má ktorý tanečník urobiť Ľavú či Pravú Patlu, aby sa tanec vydaril (prípadne vypíše, že sa to nedá). Predpokladajte, že tanečníci nastúpení v rade sú očíslovaní zľava doprava číslami 1 až  $n$ . Pre  $n = 8$  je výsledok napríklad:

najprv 5. tanečník urobí Ľavú Patlu,  
potom 3. tanečník urobí Pravú Patlu,  
potom 1. tanečník urobí Pravú Patlu,  
potom 8. tanečník urobí Ľavú Patlu.

Pre kontrolu uvádzame aj postupné fázy tanca (tanečníkov ležiacich na zemi znázorňujeme v zátvorkách, druhý v zátvorke leží na prvom):

1. fáza. 1 2 3 4 5 6 7 8

2. fáza. 1 (25) 3 4 6 7 8
3. fáza. 1 (25) 4 6 (73) 8
4. fáza. (25) (41) 6 (73) 8
5. fáza. (25) (41) (68) (73), tanec sa vydaril.

#### 942. O Burundijskom zjednotení

[21]

Po občianskej vojne sa Burundi rozpadlo na  $n$  maličkých štátikov. Každý mal svojho vládcu a meno, ale bolo ich toľko, že ich radšej budeme označovať číslami od 1 po  $n$ . Zavládol chaos a anarchia. Iba Veľký Mahdí (vládca v Burumburi) vedel, čo robiť, aby bolo Burundi opäť jednotným, mocným a nezávislým štátom. Rozhodol sa, že na znovuzjednotenie Burundi použije svoj obrovský zlatý poklad, predajnosť vládcov jednotlivých štátikov a starý burundijský zvyk – sľub spolupatričnosti.

Sľub spolupatričnosti má v Burundi veľkú tradíciu. Spočíva v tom, že dvaja vládcovia (pôvodne rodín, teraz štátikov) slávnostne sľúbia, že každý, kto je alebo kto sa stane poddaným jedného z nich, stáva sa automaticky aj poddaným toho druhého. Týmto sľubom tak vznikne akási „spoluvláda“.

Mahdího plán bol jednoduchý: svojimi peniazmi „dopomôže“ k uzavretiu toľkých sľubov spolupatričnosti, až nakoniec budú vládnuť všetci vládcovia nad celým územím Burundi. Ďalej plánoval pozvať k sebe všetkých vládcov na hostinu a tam ich dať všetkých, až na seba, pozabíjať. Tak by sa stal jediným vládcom nad celým Burundi. Veľký Mahdí teda starostlivo pripravil veľa poslov, každému dal vrece zlatých burundijských dukátov a poslal ich k ostatným vládcom intrigovať. Teraz sedí na tróne a čaká na ich správy. Posli budú postupne prichádzať a hlásiť: „Veľký Mahdí, oznamujeme Ti, že vládca krajiny X uzavrel s vládcom krajiny Y sľub spolupatričnosti“. Ak sa týmto sľubom podarilo spojiť dve územia, ktoré ešte neboli zjednotené, dá Veľký Mahdí príkaz posla bohato obdarovať. Ak však nie, tak poslovi beda: keďže zbytočne premrhal Mahdího peniaze, Mahdí ho dá utopiť.

ÚLOHA: Napište Mahdímu program, ktorý bude v cykle postupne načítavať správy poslov v tvare dvojíc  $X, Y$  a bude vypisovať, či má Mahdí dať posla odmeniť alebo utopiť. Keď program načíta správu  $0, 0$ , znamená to, že Mahdímu došli peniaze. Vtedy program skončí a oznámi, či už je Burundi zjednotené alebo nie. Snažte sa, aby jednotlivé odpovede na správy poslov dával váš program v čo najkratšom čase, lebo Mahdí je veľmi netrpezlivý.

#### 943. O malom lenivom krtkovi

[25]

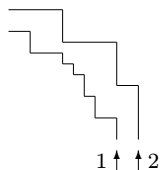
Malý lenivý krtko spal v brlôžku až do obeda. Jeho mamičku to nazlostilo, a tak si povedala: „Čo z môjho synčeka vyrastie, ak bude taký lenivý!“, a už ho aj išla budiť. „Vstávaj krtko!“, rázne sa mu prihovorila, „ak dneska nepochytáš chrobáčikov aspoň v jednom našom tuneli, tak ťa naskutku paličkou vyobšívam!“. Čo mal chudák krtko robiť, nechcelo sa mu, ale musel vstať.

Krtia rodina mala na rovine vybudovaných  $n$  brlôžkov (so súradnicami  $[x_1, y_1], \dots, [x_n, y_n]$ ), pričom medzi každými dvoma brlôžkami mala vykopaný priamy tunel. Každý deň krtica s krtom preliezli všetky tunely a zbierali to, čo mali najradšej – chrobáčikov. Teraz mal aj malý lenivý krtko zbierať, ale keďže bol celkom inteligentný, najprv pouvažoval: „Najmenej chrobáčikov, a teda aj najmenej práce, bude v najkratšom tuneli. Stačí mi teda zistiť, ktoré dva brlôžky sú k sebe najbližšie. Musím však na to prísť skôr, ako sa mamička začne o mňa znovu zaujímať, takže by som mal urobiť menej ako  $c \cdot n^2$  operácií, kde  $c$  je nejaká konštanta.“

ÚLOHA: Pomôžte krtkovi a naprogramujte mu to! Ak sa vám aj nepodarí nájsť algoritmus lepší ako kvadratický, nevadí, krtko sa poteší aj tomu, len vám dá za odmenu trošku menej chrobáčikov.

## 944. O valcočlovekovi

25



Valcočlovek je taký človek, ktorého tvar sa približuje tvaru valca s určitou výškou a šírkou. Výskyt ľudí takéhoto tvaru je veľmi vysoký napríklad medzi pracovníkmi francúzskych vínnych pivníc (čím viac vína pracovník skonzumuje, tým je širší).

Francúzske vínné pivnice sú spravidla dosť úzke, takže širší valcoľudia cez ne nedokážu ani prejsť, často sa v pivnici zaseknú a vínná pivnica sa stane nepoužívateľnou. Preto sa musí u každého pracovníka starostlivo sledovať jeho šírka, aby sa predišlo nehode.

Na to je samozrejme nutné vedieť, aký najširší valcočlovek je ešte schopný pivnicou sa pretiahnuť.

O francúzskych vínnych pivniciach je známe, že majú dva vchody spojené chodbou. Obe steny chodby sa vyznačujú tým, že nezatáčajú dvakrát po sebe tým istým smerom a vždy zatáčajú v pravom uhle. Navyše obe steny zatáčajú pri vchodoch prvý raz tým istým smerom. Vďaka tomu sa dá celá pivnica zakódovať pomocou šírky vchodu do pivnice a dvoch postupností, ktoré sa získajú nasledovne:

Prvý vinár sa postaví na ľavú stranu vchodu a druhý na pravú. Obidvaja opakovane merajú najbližší priamy úsek, až kým ich stena chodby nezatočí. Namerané hodnoty zapisujú do svojej postupnosti. Vchod, od ktorého začnú merať, zvolia tak, že obe steny zatáčajú prvý raz doľava.

ÚLOHA: Napište program, ktorý načíta šírku vchodu, dĺžku postupnosti prvého vinára, dĺžku postupnosti druhého vinára, obe tieto postupnosti a vypočíta, aký najširší valcočlovek dokáže kódovanou pivnicou bezpečne prejsť.

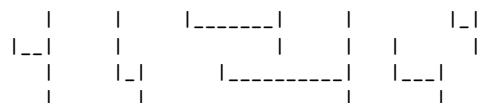
## 945. Štvrtý príklad o hre BOXES

25

Janko a Marienka sa hrali veľmi známu hru BOXES. Hra BOXES sa hrá na papieri, na ktorom je  $m \times n$  bodov usporiadaných do tvaru obdĺžnika. V každom ťahu hráči spájajú dva dosiaľ nespojené susedné body horizontálnou alebo vertikálnou čiarou. Pravidlá samotnej hry sú pre nás teraz nepodstatné (kto by ich chcel poznať, nech si pozrie príklad 815). Situácia hry BOXES je zobrazená v 925.

Takže Janko s Marienkou sa hrali túto hru a pre zmenu prehrával Janko. Keď videl, že ho od porážky už nič nezachráni, spomenul si na to, ako s ním Marienka naposledy vybabrala (925). A preto sa opýtal: „Marienka, čo myslíš, koľko je na hracom pláne štvoriek? Nepomohla by si mi ich porátať?“ Marienka sa na chvíľu zamyslela (no nie príliš dlho) a potom víťazoslávne povedala: „Ale Janko, no predsa .... A teraz to môžeme dohrať.“ „Ako sa to len mohlo Marienke podať?“ Rozmýšľal Janko, keď definitívne prehrál.

ÚLOHA: A čo si myslíte vy, milí programátori? Ako je to možné? Svoju odpoveď podporte efektívnym programom, ktorý bude v reprezentácii hernej situácie hry BOXES zisťovať počet štvoriek. Pre upresnenie: Janko si pod štvorkou predstavuje napríklad takéto obrázce:



Každá zo štyroch nožičiek štvorky môže byť ľubovoľne dlhá, ale nesmie mať nulovú dĺžku. Teda napríklad  $|_ |$  nie je štvorka. Odpoveď vášho programu pre vyššie uvedenú situáciu hry BOXES z príkladu 925 by teda mala znieť: 10.

## 1011. O Santovi a fľašiach

27

Zlatokopi z Vyšnej Klondiky radi chodievali do hostinca v Dolnom Kelčove posilňovať sa ohnivou vodou, hrať hazardné hry a uzatvárať stávky. I Santo a Banto si tam občas

radi zašli. Raz sa im podarilo vyhrať stávkou s krčmárom a za odmenu im krčmár postavil na pult do radu mnoho fliaš ohnivej vody rôzneho objemu a povedal: „Z týchto fliaš, ako tu v rade stoja, môžete vypiť (a potom prázdne vrátiť na pôvodné miesto) tie, ktoré sa vám páčia, ale tak, aby nakoniec v celom rade z každých troch po sebe idúcich fliaš bola aspoň jedna prázdna a aspoň jedna plná.“ Santo sa už-úž chcel rozbehnúť k najväčšej fliaši, keď ho Banto upozornil na to, že to nemusí byť najvýhodnejšie.

ÚLOHA: Pomôžte Santovi a napíšte program, ktorý načíta počet fliaš  $n$ , postupnosť ich objemov  $v_1, v_2, \dots, v_n$  (kde  $v_i$  je objem  $i$ -tej fľaše v krčmárovom rade) a poradí Santovi, ktoré fľaše má vypiť, aby preliel hrdlom najväčšie možné množstvo ohnivej vody.

#### 1012. O Kiribatskom chráme

[23]

Kiribatský chrám je podivuhodná stavba s pôdorysom tvaru rovnoramenného pravouhlého trojuholníka orientovaného (z náboženských dôvodov) pravým uhlom na sever. Na mape (kiribatská mapa je matica núl a jednotiek, kde nuly predstavujú more a jednotky pevninu) vyzerá chrám (vyznačený napríklad dvojkami) takto:

```

                2
              2 2 2
            2 2 2 2
          2 2 2 2 2
        2 2 2 2 2 2
      2 2 2 2 2 2 2
    2 2 2 2 2 2 2 2
  
```

Na obrázku sú uvedené chrámy v štyroch najmenších veľkostiach. Južná stena chrámu zaberá na mape vždy nepárny počet políčok a je vyjadrením veľkosti chrámu.

ÚLOHA: Napíšte program, ktorý načíta rozmery mapy, počet riadkov  $m$ , počet stĺpcov  $n$  (a maticu  $A[1..m, 1..n]$  predstavujúcu mapu súostrovia Kiribati a vyznačí na mape (napríklad dvojkami) miesto, kde Kiribatčania môžu svojim bohom postaviť najväčší chrám. Ak je takých miest v Kiribati viac, stačí vyznačiť ľubovoľné z nich.

#### 1013. O šikovnom učiteľovi

► [25]

Žiaci 4.A a 4.B triedy nastúpili v tesnej miestnosti do dvojradu, pričom v oboch radoch boli žiaci utriedení podľa výšky od najnižšieho po najvyššieho (oba rady tvorili tzv. rebrík). Učiteľ si ich dal zavolať preto, lebo chcel zistiť výšku 30. najvyššieho žiaka. Nechcel však, aby sa oba rady začali triediť podľa výšky, to by nastala priveľká panika a krik. Všimol si, že v oboch radoch je práve 30 žiakov, chvíľu striedavo pozeral naľavo a napravo. Potom z druhého radu vyviedol Jožka a zmeral si jeho výšku. Zaujímavé na tom bolo to, že na väčšinu žiakov sa ani nepozrel.

ÚLOHA: Sú dané dve polia  $A[1..n]$ ,  $B[1..n]$  rovnakej (!) dĺžky  $n$ . Obe polia sú vzostupne utriedené. Napíšte program, ktorý vypočíta hodnotu  $n$ -tého najväčšieho prvku oboch polí (t.j. hodnotu prvku, ktorý by stál na indexe  $n + 1$  v poli, ktoré by vzniklo zlúčením  $A$  a  $B$ ). Váš program by mal postupovať podobne efektívne ako šikovný učiteľ.

#### 1014. O smetiároch

[75]

Na sídlisku Číselníkov je  $n$  križovatiek očíslovaných od 1 po  $n$ . Križovatky sú pospájané ulicami rôznej dĺžky. (Pod ulicou v Číselníkovke rozumejúc súvislý úsek cesty neprerušenej križovatkou.) Taktiež každá ulica má v Číselníkovke svoje číslo. Popri každej ulici stoja smetiaky obyvateľov Číselníkov plné odpadkov. Smetiari, ktorí majú depo na križovatke s číslom 1, majú k dispozícii jediné smetiarske auto, pomocou ktorého musia každý deň všetky smetiaky vyprázdniť.

ÚLOHA: Napíšte program, ktorý pre nich naprojektuje trasu, po ktorej majú ísť, aby (vyšli z depa) vyprázdnili všetky smetiaky (a vrátili sa do depa) a pritom prešli čo najkratšiu trasu (a minuli čo najmenej drahého benzínu).

Váš program bude na vstupe prijímať počet križovatiek  $n$ , prirodzené číslo  $k$  a  $k$  štvoríc, pričom každá štvorica  $(c, i, j, dl)$  bude reprezentovať ulicu číslo  $c$ , dĺžku  $dl$ , spájajúcu

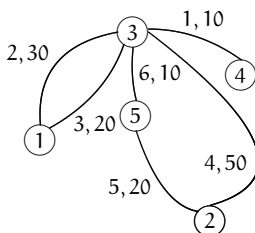
křížovatky  $i$  a  $j$ ,  $i < j$ . Výstupom vášho programu bude postupnosť čísel ulíc, ktorá obsahuje každú ulicu, pričom súčet dĺžok ulíc bude čo najmenší.

Příklad:

Vstup:

```
5
6
1 3 4 10
3 1 3 20
4 2 3 50
2 1 3 30
5 2 5 20
6 3 5 10
```

Výstup: Trasa: 2, 1, 1, 6, 5, 4, 3; Dĺžka trasy: 150



### 1015. O Batuchánových vyslancoch

30

Chán Zlatej hordy, Batu, vyslal zo svojho tábora na Volge na dvor Veľkého chána v Kambaluku (dnes Peking) troch svojich synov, aby mu blahoželali k jeho víťazstvu nad ríšou Sung. Po ceste mali ponavštevovať Batuových priateľov a odovzdať im jeho dary a pozdravy. Aby ich v rozľahlej Mongolskej ríši našli, dal Batu svojim synom zemepisné súradnice táborov svojich priateľov ako i zemepisné súradnice Kambaluku a svojho tábora na Volge. Vzhľadom na rozlohu Mongolskej ríše sa všetky zemepisné dĺžky Batuových priateľov nachádzali medzi zemepisnými dĺžkami Batuovho tábora a Kambaluku, pričom všetky boli navzájom rôzne.

„Chodte“, povedal Batu svojim synom, „a nezabúdajte na staré mongolské cestovné príslovie: ‘Nikdy neprekroč žiadny poludník viac ako dvakrát!’. Ponáhľajte sa však, lebo nadchádza chvíľa, keď dáme plemenám uhorským pocítiť naše šípy, preto chodte najkratšou cestou, ktorá neporušuje naše zvyky!“. Synovia si vzali k srdcu slová svojho otca, navštívili Veľkého chána i všetkých Batuových priateľov a vrátili sa k Volge ešte včas, aby sa pridali k Batuovej výprave do Uhorska. medzi dvomi miestami so zemepisnými súradnicami  $Sirka1$ ,  $Dlзка1$  a  $Sirka2$ ,  $Dlзка2$  je  $\sqrt{(Sirka2 - Sirka1)^2 + (Dlзка2 - Dlзка1)^2}$  (v XIII. storočí sa ešte všeobecne neprijímal názor, že Zem je guľatá).

ÚLOHA: Je dané prirodzené číslo  $n$ ,  $n > 2$  a polia  $Sirka[1..n]$ ,  $Dlзка[1..n]$ , pričom pre  $i < j$  platí, že  $Dlзка[i] < Dlзка[j]$ . ( $Sirka[1]$ ,  $Dlзка[1]$ ) sú zemepisné súradnice Batuovho tábora, ( $Sirka[n]$ ,  $Dlзка[n]$ ) Kambaluku a ostatné údaje sú zemepisné súradnice Batuových priateľov. Napíšte program, ktorý zistí, v akom poradí navštevovali Batuovi synovia jednotlivé miesta (t.j. nájde najkratšiu trasu, ktorá začína a končí v Batuovom tábore, prechádza Kambalukom a táborami Batuových priateľov a neprechádza žiadny poludník viac ako dva razy).

### 1021. O dôležitej osobe

23

V Číselníkovke žije  $n$  ľudí očíslovaných od 1 po  $n$ . Človek číslo  $i$  človeka číslo  $j$  buď pozná, alebo nepozná (sám seba každý pozná). Keď sa starostu Číselníkovy opýtate „Pozná človek číslo  $i$  človeka číslo  $j$ ?“, vie na položenú otázku odpovedať (ÁNO alebo NIE) pre ľubovoľné  $i$  a  $j$ ,  $1 \leq i, j \leq n$ . Keď sa ho však opýtate, či v Číselníkovke existuje osoba, ktorú všetci poznajú, ale ona pri tom nikoho okrem seba nepozná, pokrčí ramenami. Starosta vždy hovorí pravdu.

ÚLOHA: Napíšte program, ktorý bude starostovi kľásť otázky typu „Pozná človek číslo  $i$  človeka číslo  $j$ ?“, prijímať na ne odpovede typu ÁNO/NIE a v konečnom čase zistí, či v Číselníkovke uvedená osoba existuje alebo nie.

### 1022. O Patagónskom fjorde

23

Keď Fernando Magellan hľadal v rokoch 1519-20 v službách španielskeho kráľa a ne-

meckého cisára Karola V. prieliv do Južného mora<sup>10</sup> (dnes Tichý oceán), v zúfalej snahe prehľadával všetky fjordy na východnom pobreží Patagónie (dnes Argentína). Raz vplával do fjordu, ktorý sa veľmi často rozchádzal na dve ramená, ktoré sa ďalej zásadne nespájali. Preto bol nútený rozdeliť svoju flotilu a každej lodi dať sledovať iné rameno fjordu. Kapitán Trinidadu však nedal pozor a jeho loď uviazla na plytčine. Ihneď poslal za Magellanom na vlajkovú loď San Antonio čln s prosbou o pomoc. Keď sa správa dostala do Magellanových uší, pozorne si prezrel doterajšie hlásenia a povedal: „Je to mrzuté. Keby bol San Antonio alebo Trinidad na hocktorom inom mieste ako je, cesta (po vode) od jedného k druhému by merala menej legaus<sup>11</sup>.“

ÚLOHA: Napíšte program, ktorý načíta podstatné údaje o patagómskom fjorde a vypočíta dĺžku (vodnej) cesty, ktorú museli námorníci zo San Antonio podniknúť, aby pomohli uviaznutému Trinidadu.

Údaje o fjorde sú v tvare:

```

<fjord> = <podfjord>
<podfjord> = <dlzka ramena> legaus VLAVO <podfjord> VPRAVO <podfjord>
              | <dlzka ramena> legaus KONIEC
<dlzka ramena> = integer

```

Napríklad fjord, ktorý sa po 4 legaus delí na dve ramená dĺžky 2 legaus, pričom ľavé sa ešte delí na dve ramená dĺžky 1 legaus, sa zaznamená takto: 4 **legaus** **VLAVO** 2 **legaus** **VLAVO** 1 **legaus** **KONIEC** **VPRAVO** 1 **legaus** **KONIEC** **VPRAVO** 2 **legaus** **KONIEC**.

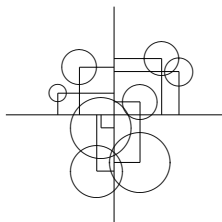
### 1023. O megalitickej kultúre

[30]

Iste poznáte megality vo Veľkej Británii. Čo však určite nevíete (lebo sme si to vymysleli) je fakt, že neďaleko nich žil dávno pradávno ľud, ktorý prinášal svojmu bohu plodnosti veľmi zaujímavé obete. Uprostred svojho sídliska mali postavenú sochu boha plodnosti. Obeta spočívala v tom, že sa šaman postavil k soche, náhodne odpočítal  $S$  krokov na sever,  $J$  krokov na juh,  $Z$  krokov na západ a  $V$  krokov na východ, až sa dostal na pole, na ktorom pestovali obilie. Podľa postavenia slnka stanovil  $r$  a ostatní všetku pôdu na

$r$  krokov od miesta (t.j. kruh), kde šaman stál, znehodnotili tak, že sa už nikdy nedala obrobiť. Znehodnocovanie pôdy prebiehalo rôzne, bohatší ju posýpali kameňmi, chudobnejší iba udupávali.  $S, J, Z, V$  a  $r$  boli každý rok rôzne, takže po niekoľkých rokoch musel ľud toto miesto opustiť. Pri sťahovaní sa rozdelil na mnoho skupín, z ktorých dobrá polovica padla za obeť útrapám a ľudožrútom.

ÚLOHA: Napíšte program, ktorý načíta počet rokov  $n$ , počas ktorých sa konali uvedené obete, čísla  $S_i, J_i, Z_i, V_i$  a  $r_i$ , kde  $1 \leq i \leq n$  a vypočíta, aká plocha (vyjadrená v šamanových krokoch štvorcových) pôdy bola pri nich znehodnotená.



### 1024. O dopravnom ihrisku

[35]

V Číselníkovke si obyvatelia postavili dopravné ihrisko. Najprv vyznačili  $n$  miest očíslovaných od 1 po  $n$ , a potom ich podľa plánu pospájali cestami (tam, kde sa potom zbíhalo viac ciest, urobili križovatku). Plán bola obyčajná matica  $A[1..n, 1..n]$  núl a jednotiek, kde  $A[i, j] = 1$ , keď miesta  $i$  a  $j$  mali byť spojené priamou cestou a  $A[i, j] = 0$ , keď nie. (Žiadna priama cesta spájajúca dve miesta neprechádza cez tretie.  $A[i, i] = 0$  pre všetky  $1 \leq i \leq n$ .)

<sup>10</sup> Dostal sa so svojou flotilou až na Filipíny, kde ho zabili domorodci (1521). Iba jednej lodi jeho flotily (Victoria, pod vedením Sebastiána del Cano) sa podarilo dostať späť do Španielska (1522). [2]

<sup>11</sup> Legaus = španielsko-portugalská námorná míľa, t.j. jednotka dĺžky.

Keď bolo všetko hotové, zmocnila sa starostu zlá predtucha: „Občania, budeme my môcť na tomto ihrisku cvičiť jazdu do osmičky?“

ÚLOHA: Napíšte program, ktorý načíta počet miest  $n$ , plán ihriska  $A[1..n, 1..n]$  a zodpovie na starostovu otázku (ÁNO alebo NIE). Nezabudnite tiež pridať úvahu o efektívnosti vášho algoritmu, prípadne odvodenie zložitosti.

Na to, aby ste sa mohli dať do práce, treba si ešte sformalizovať pojem „jazda do osmičky“. Pod „osmičkou“ si budeme predstavovať postupnosť miest tvaru  $b, u_1, u_2, \dots, u_k, b, v_1, v_2, \dots, v_l, b$ , kde

- 1.)  $k \geq 2, l \geq 2$
- 2.) sú každé dve po sebe idúce miesta spojené priamou cestou
- 3.) existuje index  $r$ , že pre každý index  $s$ ,  $1 \leq s \leq l$  platí  $u_r \neq v_s$
- 4.)  $u_i \neq b$  a  $v_j \neq b$  pre každé  $1 \leq i \leq k$  a  $1 \leq j \leq l$
- 5.) pre každé tri po sebe idúce miesta  $m_1, m_2, m_3$  platí  $m_1 \neq m_3$ .

#### 1025. O Santovi a dynamite

[25]

Raz sa Santo opil a dostal nešťastný nápad: vyhodiť celé svoje nálezisko zlata do vzduchu. Porozhádzoval po ňom preto šúľky dynamitu a začal ich spájať zápalnou šnúrou, aby ich mohol odpáliť. Keď to uvidel Banto, rýchlo začal zápalné šnúry zasa rozpájať, lebo ho nelákala vidina, že z jeho jediného majetku ostanú kúdky prachu.

ÚLOHA: Napíšte program, ktorý načíta počet šúľkov dynamitu  $n$  (šúľky sú očíslované od 1 po  $n$ ) a ďalej postupne číta vstupy tvaru SANTO SPOJIL  $i$   $j$  alebo BANTO ROZPOJIL  $i$   $j$ , kde  $i, j$  sú čísla šúľkov, alebo KONIEC. Keď sa načíta KONIEC, ukončí sa beh programu. Vždy keď sa Santovi podarí spojiť všetky šúľky dynamitu (a teda môže zrealizovať svoj opilecký sen), program zareaguje na daný vstup a upozorní Banta.

#### 1031. U holiča

[18]

U holiča bolo  $2n + 1$  stoličiek, na ktorých sedelo  $n$  mladých a  $n$  starých ľudí, jedna stolička bola prázdna. Stoličky boli očíslované od 1 po  $2n + 1$ . Ten, kto sedel na stoličke s menším číslom, sa dostal „pod kombajn“ prv, ako tí, čo sedeli na stoličkách s väčším číslom.

Keď to zbadal holič, vyhlásil, že všetci mladí musia prepustiť výhodnejšie miesta starším, inak strihať nebude. A navyše, on tu nechce mať žiaden neporiadok, takže mladí sa so starými povymieňajú pekne poporiadku: vždy niekto vstane a sadne si na voľnú stoličku, tým sa jeho stolička uvoľní, na ňu si sadne niekto ďalší, atď. Je jedno, kto kde bude sedieť počas vymieňania, nakoniec však musia sedieť starší „pred“ mladšími.

ÚLOHA: Napíšte program, ktorý načíta  $n$ , pre každú stoličku  $i$  načíta, kto na nej sedí ( $-1$  je mladý,  $0$  je nikto,  $1$  je starý) a vypíše postupnosť čísel stoličiek, z ktorých majú tí, čo na nich sedia, vstať a sadnúť si na (v tej chvíli) voľnú stoličku, aby po vykonaní týchto presunov boli všetci starí pred mladými (na polohe voľnej stoličky nezáleží). Bolo by dobré, keby váš program vypísal vždy postupnosť minimálnej dĺžky (dokážte!). Taktiež by sme sa potešili, keby bol váš program lineárny.

#### 1032. O švajčiarskej pechote

[18]

Základný prvok švajčiarskej pechoty bol štvorec  $n \times n$  kopijníkov (pod veľkosťou tohto štvorca budeme rozumieť rozmer  $n$ ), ktorých kopije smerovali vždy von z tohoto štvorca, takže predstavovali pre nepriateľskú rytiersku jazdu, útočiacu z ktorejkoľvek strany, neprekonateľnú prekážku. Celá pechota pozostávala z takýchto štvorcov pre  $n = 1, 2, \dots, k$  (neobsahovala dva rovnaké štvorce), pričom  $k$  záležalo od veľkosti pechoty.

Každý kopijník patriaci k švajčiarskej pechote mal svoje identifikačné číslo, pričom všetky identifikačné čísla boli prirodzené ( $1, 2, 3, \dots$ ) a navzájom rôzne. Identifikačné číslo jediného kopijníka vo štvorci veľkosti 1 bola jednotka. Ak mal nejaký kopijník identifikačné číslo  $id$ , buď to bola jednotka, alebo existoval v pechote kopijník s identifikačným číslom



id – 1. Pri udeľovaní identifikačných čísel platila ďalej zásada, že každý kopijník zo štvorca menšej veľkosti mal menšie číslo ako ktorýkoľvek kopijník zo štvorca väčšej veľkosti.

ÚLOHA: Napište program, ktorý načíta identifikačné číslo kopijníka švajčiarskej pečoty a vypíše veľkosť štvorca, do ktorého patrí. Program by mal pracovať dostatočne rýchlo pre ľubovoľné identifikačné číslo do  $10^9$ .

### 1033. O listine kráľov

[21]

Pri archeologickom výskume sa našla do kameňa neznámym slabičným písmom vytesaná listina v neznámom jazyku. Podľa ilustrácií išlo evidentne o listinu kráľov dosiaľ neznámeho národa. Dalo sa teda predpokladať, že najfrekvencovanejšie slovo v listine bude slovo kráľ.

Na prvý pohľad bolo jasné, že vodorovný klin oddeľuje jednotlivé slová a že písmo sa číta sprava doľava a zdola nahor. Nálezcovia preto listinu dôkladne okopírovali a každému znaku pre slabiku pridelili číslo od 1 do 138. Každé slovo listiny zaznamenali do jedného riadku súboru listina.in ako postupnosť čísel znakov jeho slabík oddelených medzerami. Žiadne slovo nemalo viac ako 5 slabík.

ÚLOHA: Napište program, ktorý prečíta súbor listina.in a zistí všetky pravdepodobné zápisy slova kráľ.

Napríklad pre súbor listina.in

2 11 11

1 122 12

8 8 1 1 8

2 3

1 122 12

2 3

výstup: 1 122 12, 2 3.

### 1034. O lenivých novinároch

▶ [35]

Niektorí novinári sú takí leniví, že do článku napíšu jednu pasáž aj viackrát, aby ľahko dosiahli požadovanú dĺžku. Šéfredaktori preto potrebujú program, ktorý pre daný text článku  $T[1..Dlžka]$  ( $Dlžka \leq 1000$ ) zistí najdlhší úsek, ktorý sa v článku opakuje na inom mieste. V prípade, že je takých úsekov viac, zistí najľavejší z nich.

Napríklad pre  $Dlžka = 13$ ,  $T = \text{abedabehdabeh}$  je výsledkom dabeh

### 1035. O hľadaní bodu

[??]

Vymenovaním proti smeru chodu hodinových ručičiek je daný konvexný  $n$ -uholník. Napište program, ktorý pre dané  $n$  a konvexný  $n$ -uholník nájde bod  $C$  taký, že maximum zo vzdialeností od tohoto bodu do všetkých vrcholov  $n$ -uholníka je minimálne.

Napríklad pre 5-uholník  $[0, 0], [2, -1], [4, 0], [3, 1], [1, 1], [0, 0]$  je  $C = [2, 0]$ .

### 1041. Kombinačné číslo

[25]

Napište program, ktorý načíta celé čísla  $n$  a  $k$ ,  $0 \leq k \leq n \leq 100$  a vypočíta presnú hodnotu kombinačného čísla  $\binom{n}{k}$ . Pripomíname, že  $\binom{n}{k} = n! / (k!(n-k)!)$ , kde  $x! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot x$ . Napríklad pre  $n = 100$  a  $k = 50$  je výsledok 100891344545564193334812497256.

### 1042. Katastrofa v Kiribati

[21]

Následkom zvyšovania obsahu ropy a olejov v morskej vode sa všetky koralové ostrovy pomaly, ale isto rozpadávajú (a potápajú). Ani Kiribati táto katastrofa neobišla. Z predchádzajúcich príkladov viete, že mapu Kiribati si možno predstaviť ako maticu núl a jednotiek, kde jednotky sú pevnina a nuly more. Katastrofu si potom možno predstaviť tak, že sa každý rok rozpadnú a potopia tie časti ostrovov, ktoré sú na mape reprezentované políčkami, ktoré aspoň jednou stranou susedia s morom.

ÚLOHA: Napište program, ktorý načíta rozmery mapy  $m$  a  $n$ , mapu s  $m$  riadkami a  $n$  stĺpcami núl a jednotiek a zistí, za koľko rokov klesne prvý (dnešný!!!) ostrov celý pod hladinu.

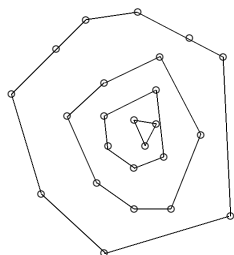
## 1043. Santove vrstvy

27

Možno sa pamätáte, že Santova a Bantova parcelka bola už po prvom vykolikovaní kade-tade posiatá kolíkmi. Jedného slnečného dňa sa v krčme v Dolnom Kelčove Santo dozvedel, že bol vyhlásený nový pozemkový zákon. Zákon obsahoval nasledujúce paragrafy:

- §1. Zakazuje sa zatĺkať nové kolíky za účelom vyznačenia parcely.
- §2. Ruší sa doterajšie rozdelenie zlatokopeckých parciel. Nové parcely sa vyznačia špagátom, ktorý sa musí obopínať o kolíky zatĺčené pred dňom vyhlásenia tohto zákona, nemusí však použiť všetky kolíky. Špagát musí byť napnutý. Špagát sa nesmie vetviť a nesmie prechádzať popod alebo ponad iný špagát (vyznačujúci inú parcelu, viď §4).
- §3. Nové parcely môžu mať iba jedného majiteľa, a to toho, kto natiahol parcelu vyznačujúci špagát. Viď §4.
- §4. Parcela je priestor vnútri špagátu, s výnimkou parciel nachádzajúcich sa v tomto priestore.

Banto (ako starý abstinent) nebol vtedy v krčme, takže sa dosť začudoval nad tým, že Santo hneď po návrate z krčmy obohnal okolo ich parcelky špagát. Myslel si, že sa zase opil, ale Santo mu víťazoslávne oznámil, že sa práve stal jediným majiteľom celej ich parcelky, pretože v zmysle nového zákona si práve vyznačil parcelu s najväčšou možnou plochou, nakoľko to ich kolíky umožnili.



Banto sa však nedal ľahko odbiť. Preštudoval si nový zákon a v zmysle §4 zobral špagát a obohnal kolíky, ktoré Santo nepoužil (uvedomte si, že všetky tieto kolíky museli byť nutne v Santovej parcele). Takisto dbal na to, aby si vyznačil najväčšiu parcelu. Víťazoslávne potom Santovi svoj čin oznámil. Santo ihneď vbehol do už Bantovej parcely a

urobil Bantovi to isté, čo on jemu. A Banto znova Santovi. Tak to išlo, až sa minuli kolíky, t.j. ostali menej ako tri kolíky, prípadne zvyšné kolíky ležali na priamke.

ÚLOHA: Napište program, ktorý načíta počet kolíkov  $n$ , súradnice kolíkov  $x_i, y_i, 1 \leq i \leq n$  a vypočíta, koľko parceliek vlastní Santo a koľko Banto po tomto slnečnom dni. (Program by nemal mal byť väčšej než kvadratickej zložitosti.)

## 1044. Vekslovanie

20

Pred hotelom Kyjev stojí  $n$  vekslákov. Každý vekslák  $i$  mení menu  $A_i$  na menu  $B_i$  v pomere  $p_i : q_i$ . Momentálne stojíte pred hotelom  $i$  vy a máte  $k$  slovenských korún. Chcete ich zameniť za US \$, samozrejme v čo najvýhodnejšom kurze. Využiť môžete aj viacnásobnú výmenu skrz rôzne meny. Veksláci však musia tiež z niečoho žiť, takže sa vám nemôže stať, že by vám suma v akejkoľvek mene po viacnásobnej výmene stúpala (po viacnásobnej výmene môžete dostať opäť pôvodnú menu, ale neoplatí sa vám to).

ÚLOHA: Napište program, ktorý načíta vyššie uvedenú špecifikáciu vekslákov a vypíše najvýhodnejší kurz koruny a dolára. (Na sume  $k$  nezáleží, vzhľadom na to, že neuvažujeme zaokrúhľovanie.) Meny sa uvádzajú v medzinárodne zaužívaných skratkách nepresahujúcich desať znakov. Napríklad:

```
1. vekslák: Sk -> DM 18:1
2. vekslák: DM -> US$ 5:3
3. vekslák: Sk -> US$ 31:1
Výstup: 30:1
```

**1045. Darmožráči**

21

V Číselníkovke žije  $n$  občanov, očíslovaných od 1 po  $n$ . Každý z nich žije svojou prácou niekoľko spoluobčanov (môže sa stať aj to, že niekto nežije nikoho a niekto žije sám seba). Keď občan  $i$  žije občana  $j$ , hovoríme, že  $Z(i, j)$ . Ďalej hovoríme, že občan  $i$  žije v konečnom dôsledku občana  $j$ , keď existujú občania  $u_1, u_2, \dots, u_k, k \geq 1$  takí, že  $Z(i, u_1) \& Z(u_1, u_2) \& \dots \& Z(u_k, j)$ , alebo keď  $Z(i, j)$  a tento fakt zapisujeme  $Z^*(i, j)$ . Darmožráčom je ten občan  $d$ , ktorého žije v konečnom dôsledku viac občanov ako on sám v konečnom dôsledku žije, t.j. pre ktorého platí  $|\{k : Z^*(i, k)\}| < |\{l : Z^*(l, i)\}|$ . ( $|X|$  predstavuje počet prvkov množiny  $X$ .)

ÚLOHA: Napíšte program, ktorý načíta počet občanov  $n$ , zoznam dvojíc čísel občanov (prvý v dvojici žije druhého, zoznam je ukončený dvojicou 0, 0) a vypíše čísla všetkých darmožráčov.

**z1011. O čiapočkách**

11

Z rozprávok iste poznáte Červenú Čiapočku. V našom rozprávkovom lese však okrem nej vystráajú aj Fialová, Modrá, Žltá, Oranžová, Okrová, Tyrkysová Čiapočka a mnohé iné, každá celá v oblečení svojej farby. Práve sa im stala galiba:

Pri hre na chytačku odložili na kopu svoje čiapočky, aby sa im nezašpinili. Keď bola hra v najlepšom, zjavil sa znenazdajky vyhladnutý vlk, ledva sa držal na nohách. Čiapočky sa prefakli, každá chytró zobrala z kopy jednu čiapočku a vzali nohy na plecia. Keď boli v bezpečí, zistili, že skoro každá z nich zobrala z kopy cudziu čiapočku. Každá si chce tú, čo má, okamžite vymeniť z rúčky do rúčky s nejakou inou Čiapočkou za svoju pôvodnú. Dá sa to?

ÚLOHA: Predpokladajme, že v lese je  $n$  rôznofarebných oblečených Čiapočiek. Pre jednoduchosť každej farbe priradíme číslo z intervalu od 1 po  $n$ . Napíšte program, ktorý načíta počet Čiapočiek  $n$  a pole  $C[1..n]$ , kde  $C[i]$  je číslo farby čiapočky, ktorú drží v ruke Čiapočka, ktorej oblečenie má farbu číslo  $i$ , a vypíše, či je možné, aby si Čiapočky vymenili vyššie uvedeným spôsobom svoje čiapočky tak, aby mala každá svoju pôvodnú.

**z1012. O nešťastnom čísle**

??

Pred medzištátnym futbalovým zápasom nastúpilo mužstvo trénera Mamojku do radu, aby si vypočulo hymny. Poverčivý Mamojka sa vždy obával svojho nešťastného čísla. Preto aj zakázal svojim hráčom nosiť dresy s týmto číslom. Keď uvidel, ako jeho mužstvo stojí v rade, vydesilo ho, že súčet čísel dresov niekoľkých vedľa seba stojacich hráčov sa môže náhodou rovnať jeho nešťastnému číslu. Prv ako stihol preveriť, či je to tak, hráči sa rozišli hrať a Mamojka si do konca zápasu od strachu obhrýzol všetky prsty. Bola Mamojkova obava opodstatnená alebo nie?

ÚLOHA: Napíšte program, ktorý načíta presný počet hráčov  $n$ , pole  $C[1..n]$ , kde  $C[i]$  je číslo dresu  $i$ -teho hráča v rade a Mamojkovo nešťastné číslo  $k$  a zistí, či boli Mamojkove obavy opodstatnené.

**z1013. O horskom nosičovi**

►18

Horský nosič Pišta sa chystá na cestu. Jeho náklad tvoria rovnako veľké a približne rovnako ťažké súdky rôznych tekutín (rôzne oleje, sirupy, alkoholické nápoje, ...). Z  $n$  súdkov (očíslovaných číslami 1, 2, ...,  $n$ ), ktoré má k dispozícii, dokáže však vyniesť len  $k$ ,  $1 \leq k \leq n$ , a je mu jedno, ktoré súdky to budú.

ÚLOHA: Napíšte program, ktorý načíta čísla  $n$  a  $k$  a vypíše všetky možné  $k$ -tice súdkov, ktoré mohol Pišta zobrať na cestu, a počet týchto  $k$ -tic. (Na poradí  $k$ -tic nezáleží, každú  $k$ -ticu však musí program vypísať práve raz.) Napríklad pre  $n = 3$ ,  $k = 2$  vypíše váš program: (1,2) (1,3) (2,3).

**z1014. O pretláčaní**

13

Na turnaji v pretláčaní sa stretlo  $n$  navzájom rôzne silných silákov. Po každom pretláčaní majú obaja borci možnosť dostatočne si oddýchnuť, takže každé pretláčanie vyhrá silnejší borec (keby niekto už pred turnajom vedel, kto z borcov je najsilnejší, vedel by aj, kto turnaj vyhrá).

ÚLOHA: Napíšte program, ktorý navrhne rozpis zápasov (podobný ako je uvedené nižšie v príklade) taký, aby po jeho vykonaní bolo všetkým divákovi jasné, kto z borcov je najsilnejší a kto najslabší. Rozpis by mal obsahovať čo najmenej zápasov. (Skúste nájsť vzorec, ktorý pre dané  $n$  udáva počet zápasov v rozpise vášho programu.)

Napríklad pre  $n = 4$  môže byť rozpis takýto (pozor!, pre  $n = 4$  existuje aj lepší rozpis):

zápas č. 1	...	borec č. 1 - borec č. 2
zápas č. 2	...	borec č. 3 - víťaz zápasu č. 1
zápas č. 3	...	víťaz zápasu č. 2 - borec č. 4
zápas č. 4	...	borec č. 3 - borec č. 4
zápas č. 5	...	porazený v zápase č. 1 - porazený v zápase č. 4

**z1021. O čiapočkách II**

10

Iste viete, že v našom rozprávkovom lese spolu s Červenou čiapočkou vystráajú aj Fialová, Modrá, Žltá, Oranžová, Okrová, Tyrkysová Čiapočka a mnohé iné, každá celá v oblečení svojej farby. Práve sa im opäť stala galiba:

Pri hre na schovávačku odložili na kopy svoje farebné čiapočky, aby si ich pri hre nestratili. Keď bola hra v najlepšom, zjavil sa znenazdajky veľký medveď. Ešte sa nestačil uložiť na zimný spánok. Čiapočky sa prefakli, každá chytrá zobrala z kopy jednu čiapočku a vzali nohy na plecia.

Keď boli v bezpečí, zistili, že skoro každá z nich drží v ruke cudziu čiapočku. Každá Čiapočka sa preto rýchlo chytila voľnou rukou svojej pôvodnej čiapočky. Koľko kruhov čiapočky utvorili? Za kruh sa považuje aj jedna Čiapočka, ktorá oboma rukami drží svoju pôvodnú čiapočku.

ÚLOHA: Predpokladajme, že v lese je  $n$  rôznofarebne oblečených Čiapočiek. Pre jednoduchosť každej farbe priradíme číslo z intervalu od 1 po  $n$ . Napíšte program, ktorý načíta počet Čiapočiek  $n$  a pole  $C[1..n]$ , kde  $C[i]$  je číslo farby čiapočky, ktorú drží v ruke Čiapočka s farbou oblečenia číslo  $i$ , a vypíše, koľko kruhov (cyklov) Čiapočky utvorili.

**z1022. O Perzskej kráľovskej ceste**

10

Perzská kráľovská cesta<sup>12</sup> spájala mesto Susy v Médii s mestom Sardy v Lýdii. Na V. storočie pred našim letopočtom to bola veľmi moderná cesta, ktorá mala na významných miestach vybudované stanice s možnosťou nocľahu a s dostatočným množstvom čerstvých koní. Keď „kráľ kráľov“ Xerxes<sup>13</sup> plánoval zhromažďovanie vojsk na výpravu proti Grékom, potreboval vedieť dĺžku cesty medzi ľubovoľnými dvomi stanicami. Z plánov cesty však vedel iba vzdialenosť medzi dvomi susednými stanicami.

ÚLOHA: Poradte „kráľovi kráľov“ spôsob, akým si najefektívnejšie zráta chýbajúce údaje, t.j. napíšte program, ktorý načíta počet staníc  $n$ , postupnosť  $L_1, L_2, \dots, L_{n-1}$ , kde  $L_i$  predstavuje dĺžku cesty medzi  $i$ -tou a  $(i+1)$ -vou stanicou, a vypočíta dĺžky ciest medzi ľubovoľnými dvomi stanicami.

**z1023. O horskom nosičovi II**

► 20

Horský nosič Pišta sa opäť chystá na cestu. Jeho náklad však tvoria rôzne potraviny s rozličnou cenou a rozličným objemom, ktoré sú dobre deliteľné (saláma, syr, slanina, chlieb, atď.). Z  $n$  druhov (očíslovaných číslami  $1, 2, \dots, n$ ), ktoré má k dispozícii, si musí

<sup>12</sup> Dal ju postaviť Dareios I (522–486 p.n.l.). Cesta je 5m široká a 2500km dlhá.

<sup>13</sup> bol synom Darea

vybrať len niektoré, lebo sa mu všetky nezmestia do batohu s objemom  $v$ . V jeho záujme je vybrať predmety tak, aby za ne na chate dostal čo najviac peňazí.

ÚLOHA: Napíšte program, ktorý načíta objem Pištovho batohu, počet druhov potravín  $n$ , ich objemy a ceny a vypočíta, koľké časti si má Pišta odrezať z každého druhu potravín, aby za ne dostal čo najviac peňazí. Napríklad pre batoh s objemom 3, dve potraviny, pričom prvej je 10 jednotiek a jednotková cena je 5 a duhej je 1 jednotková cena je 6, treba zobrať  $1/5$  prvej a druhú všetku.

#### z1024. O Santovi a burze

[13]

Keď Santo a Banto našli na Vyšnej Klondike prvé zlaté hrudky, vybrali sa do Dolného Kelčova investovať ich, lebo nechávajú si väčšiu hotovosť u seba, sa na Klondike rovnalo istej záhube. Dolnokelčovská burza ponúkala na predaj akcie iba dvoch spoločností: Ťažobnej spoločnosti, ktorá zlatokopom dodávala technické vybavenie a Whisky company, ktorá im dodávala ohnivú vodu. Santo by bol samozrejme najradšej všetky peniaze investoval do Whisky company, ale Banto ho presvedčil, že tak by mohol o svoj majetok rýchlo prísť. Zašli preto k cigánke, ktorá bola známa svojimi jasnovidckými schopnosťami a nechali si prorokovať ceny akcií oboch spoločností na najbližších  $n$  dní (ceny na burze sa vždy menili iba o polnoci). Cigánka pozrela do svojej sklenenej gule a vyčítala z nej postupnosť  $t_1, t_2, \dots, t_n$  cien akcií Ťažobnej spoločnosti ako i postupnosť  $w_1, w_2, \dots, w_n$  cien akcií Whisky Company.

ÚLOHA: Napíšte program, ktorý Santovi a Bantovi vypočíta, ako majú kupovať a predávať svoje akcie, aby po  $n$  dňoch znásobili svoj majetok čo najviac. Predpokladajte pritom, že cigánka sa vo svojej predpovedi nepomýlila. Nezabúdajte, že všetok Santov a Bantov majetok musí byť stále investovaný v akciách, t.j. ak Santo a Banto nejaké akcie predajú, tak ešte v ten deň musia za všetky utržené peniaze nejaké akcie nakúpiť, s výnimkou  $n$ -tého dňa, kedy sa všetky akcie speňažia. Ďalej predpokladajte, že na Dolnokelčovskej burze sa dajú kúpiť i časti akcií ( $1/2$  akcie a pod.) a uvedomte si, že vzhľadom na tento predpoklad je výpočet vášho programu nezávislý od sumy peňazí, ktorú Santo a Banto dostali za svoje zlaté hrudky. Ceny akcií sú reálne čísla.

#### z1031. O binárnych vektoroch

[13]

Binárny vektor rozmeru  $n$  je  $n$ -tica núl a jednotiek. Napíšte program, ktorý pre dané  $n$  vypíše všetky binárne vektory dĺžky  $n$  (každý práve raz) v takom poradí, že dva po sebe idúce vektory a prvý a posledný vektor sa líšia v jedinej číslici<sup>14</sup>.

Pre  $n = 3$  vypíše program napríklad:

$(0,0,0), (0,1,0), (0,1,1), (0,0,1), (1,0,1), (1,1,1), (1,1,0), (1,0,0)$ .

#### z1032. O horskom nosičovi III

[15]

Horský nosič Pišta sa chystá na cestu. Má k dispozícii batoh s objemom  $k$  a pätnásť (tento raz nedeliteľných) predmetov očíslovaných od 1 po 15 s objemami  $v_1, v_2, \dots, v_{15}$  a cenami  $c_1, c_2, \dots, c_{15}$ . Napíšte program, ktorý Pištovi poradí čísla predmetov, ktoré si má do batohu dať, aby sa mu tam jednak vošli a aby tam mal tovar najväčšej možnej hodnoty.

#### z1033. $a..ab..bc..c$

[13]

Je dané slovo  $w$ , t.j. pole znakov, dĺžky  $n$ . Napíšte program, ktorý nájde najväčšie  $k$ , pre ktoré existuje v tomto slove podslovo tvaru  $a..ab..bc..c$  zložené z  $k$  áčok, z  $k$  béčok a z  $k$  céčok za sebou.

Napríklad pre slovo  $aaabbbbcccccaaaabbcccc$  je  $k = 2$ .

<sup>14</sup> Takáto postupnosť vektorov sa nazýva Grayov kód po Frankovi Grayovi.

**z1034. „Skladanie“ skupín Čiapočiek****11**

Predpokladajme, že v našom rozprávkovom lese vystrája  $2n$  Čiapočiek, pričom ku každej z  $n$  farieb, očíslovaných od 1 po  $n$ , existujú práve dve Čiapočky s oblečením tejto farby. Všetky Čiapočky sa rozdelili do dvoch skupín tak, že v ani jednej skupine neboli dve Čiapočky rovnakej farby. Potom obe skupiny začali tancovať čiapočkovský tanec na rôznych miestach čistinky. Na začiatku tanca v oboch skupinách Čiapočky vyhodili svoje čiapočky do vzduchu, každá chytila nejakú čiapočku a potom sa ľavou rukou chytila pravej ruky Čiapočky, ktorá mala kabátik tej istej farby ako ňou chytená čiapočka. Čiapočky takto vytvorili kruhy, roztočili ich a ujúkali. Niektoré Čiapočky, ktorým sa podarilo chytiť svoju vlastnú čiapočku, sa otáčali na mieste. Takto Čiapočky tancovali a ujúkali, až sa obe skupiny dostali na dohľad. Vtedy si každá Čiapočka pozrela farbu kabátika Čiapočky, ktorú drží ľavou rukou. Potom očami vyhládala v opačnej skupine Čiapočku, ktorá mala kabátik takej istej farby, preniesla svoj zrak na kabátik Čiapočky, ktorú táto Čiapočka drží ľavou rukou a dobre si zapamätala jeho farbu. Na povel si potom Čiapočky v oboch skupinách pustili ruky a prechytili sa tak, že ľavou rukou chytili tú Čiapočku vo svojej skupine, ktorej farba kabátika bola zhodná so zapamätanou farbou. A ujúkali a točili sa až do večera.

ÚLOHA: Napíšte program, ktorý načíta  $n$  a pre obe skupiny pre všetky Čiapočky farbu čiapočiek, ktoré na začiatku tanca chytili a znázorní obe tanečné skupiny pred a po prechytení.

Napríklad pre  $n = 5$ 

1. skupina

=====

1.Čiap. chytila čiap. farby 2  
 2.Čiap. chytila čiap. farby 3  
 3.Čiap. chytila čiap. farby 4  
 4.Čiap. chytila čiap. farby 1  
 5.Čiap. chytila čiap. farby 5

1. skupina pred prechytením

=====

1 -> 2 -> 3 -> 4 -> 1  
 5 -> 5

1. skupina po prechytení

=====

1 -&gt; 3 -&gt; 5 -&gt; 4 -&gt; 2 -&gt; 1

2. skupina

=====

1.Čiap. chytila čiap. farby 2  
 2.Čiap. chytila čiap. farby 3  
 3.Čiap. chytila čiap. farby 1  
 4.Čiap. chytila čiap. farby 5  
 5.Čiap. chytila čiap. farby 4

2. skupina pred prechytením

=====

1 -> 2 -> 3 -> 1  
 4 -> 5 -> 4

2. skupina po prechytení

=====

1 -&gt; 3 -&gt; 2 -&gt; 4 -&gt; 5 -&gt; 1

1->2->3->4 je ten istý kruh Čiapočiek ako 3->4->1->2, ale nie ten istý ako 4->3->2->1 !

Skúste nájsť zadanie:

- pre čo najmenšie  $n$ , pre ktoré sú obe skupiny po prechytení rôzne,
- pre čo najväčšie  $n$ , pre ktoré sú obe skupiny po prechytení rovnaké (triviálne i netriviálne).

**z1041. Sám od seba samého seba****15**

Napište, alebo aspoň vymyslite, ako by sa dal napísať program (v jednom z jazykov PASCAL, C, BASIC, prípadne v ich odrodách), ktorý po spustení vypíše svoj zdrojový text. Program nesmie používať prácu so súbormi, v BASICu nesmie používať príkazy *LIST* a *DATA*. Akýkoľvek výstup môže realizovať len pomocou *write*, *writeln* (PASCAL), *printf* (C), *PRINT* (BASIC). Jednou vetou, svoj zdrojový text musí poznať sám od seba a nesmie sa naň pýtať OS.

**z1042. Hoare**

[11]

Je dané  $n$  prvkové pole  $A[1..n]$  celých čísel a index  $i$ ,  $1 \leq i \leq n$ . Označme  $p$  hodnotu prvku  $A[i]$ . Napište program, ktorý preusporiada toto pole (t.j. povymieňa jeho prvky) tak, že hodnota  $p$  bude na nejakom indexe  $j$  (t.j.  $p = A[j]$ ) a bude platiť:

- pre všetky  $k$ ,  $1 \leq k < j$ ;  $A[k] \leq p$  a súčasne
- pre všetky  $k$ ,  $j < k \leq n$ ;  $A[k] \geq p$ .

**z1043. AB-slová**

[14]

AB-slovo je slovo zložené výlučne z písmen A a B, napríklad AA, ABB, BBB. Napište program, ktorý na úvod prečíta zo vstupu postupnosť rôznych AB-slov, oddelených jednou medzerou. Potom bude v cykle zo vstupu prijímať AB-slovo, na ktoré zareaguje jedným z nasledujúcich spôsobov:

- ak prijal prázdne slovo, ukončí svoju činnosť,
- ak prijal slovo, ktoré nebolo v úvodnej postupnosti, vypíše reťazec **nepoznám** a pokračuje,
- ak prijal slovo, ktoré bolo v úvodnej postupnosti, vypíše jeho poradie v tejto postupnosti a pokračuje.

Napríklad pre úvodnú postupnosť AAAA ABAB máme: Vstup Výstup

AAA	nepoznám
ABAB	2
AAAA	1

**z1044. Vzorcotvorca**

[14]

Asi viete, že  $1 + 2 + 3 + \dots + n = n(n+1)/2$ . Už menej z vás pozná vzorec  $1^2 + 2^2 + 3^2 + \dots + n^2 = n(n+1)(2n+1)/6$ . Ľahko si však pre akékoľvek zvolené  $k$  vyrátate vzorec pre súčet  $1^k + 2^k + 3^k + \dots + n^k$ , ak vám prezradíme, že vzorec má tvar polynómu  $k+1$ . stupňa, t.j.  $a_{k+1}n^{k+1} + a_k n^k + \dots + a_1 n + a_0$ . Je to však ľahké urobiť program?

ÚLOHA: Napište program, ktorý pre dané  $k$  vypíše vzorec pre súčet radu  $k$ -tych mocnín. Desatinné čísla vo vzorci stačí vypísať na dve desatinné miesta (pokúste sa ich nahradiť zlomkami).

Například pro  $k = 2$  vypíše  $0.33*n^3 + 0.50*n^2 + 0.16*n^1 + 0.00*n^0$ .

**1111. O arite otáznika**

[23]

Aritou operátora nazývame počet operandov, ku ktorým sa viaže (napríklad  $+$  má aritu 2). Ďalej definujeme P-výraz nasledujúcimi pravidlami:

- P-výrazom je 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- keď  $O$  je operátor s aritou  $n$  a  $v_1, v_2, \dots, v_n$  sú P-výrazy, P-výrazom je aj  $O v_1 v_2 \dots v_n$ .  $O$  môže byť jeden z nasledujúcich operátorov:  $\$$  s aritou 1,  $+$   $-$   $*$   $/$  s aritou 2,  $F$   $G$   $H$  s aritou 3 a  $?$  s neznámou (ale pevnou) aritou.
- nič iné P-výrazom nie je.

ÚLOHA: Napište program, ktorý zistí, či daná postupnosť znakov tvorí P-výraz. V prípade, že tvorí, vypíše aj aritu otáznika.

Například +22 F+22\*33F123 \$\$\$\$1 ???1 ??222 \$? sú P-výrazy a +aa F1234 \$22 \$ ???11 +?12?123 nie sú P-výrazy.

**1112. O súčte čísel**

► [17]

Na vstupe máme postupnosť celých čísel ukončenú nulou. Napište program, ktorý načítá túto postupnosť a vypíše jej súčet.

Obmedzenie: V celom programe môžete použiť iba procedúry bez parametrov a jedínú jednoduchú globálnu premennú.

## 1113. O Kiribatských plavidlách

100

Pobrežné vody okolo súostrovia Kiribati sú dosť búrlivé, preto si domorodci stavajú zvláštne plavidlá tak, aby mali čo najväčšiu stabilitu. Kiribatské plavidlo sa skladá z  $n$  malých plátí očíslovaných 1 až  $n$ , z ktorých niektoré dvojice sú pozväzované rôzne dlhými palicami.

ÚLOHA: Váš program má načítať počet malých plátí  $n$ , počet palíc  $m$  a  $m$  trojíc čísel  $A_i, B_i, D_i$  s významom: plát  $A_i$  je zviazaná s plátom  $B_i$  palicou dĺžky  $D_i$ . Výstupom je ANO, ak takto zadané plavidlo je funkčné (t.j. existuje v rovine rozloženie plátí tak, aby vyhovovali vzdialenosti), alebo NIE, ak nie je.

Palice sa uvádzajú vždy na jedno miesto plte, takže plte môžete považovať za body

## 1114. O Santovom pohľade na sídlisko

21

V súbore domy.dat sú reprezentované domy tvoriace jedno sídlisko. Všetky domy majú obdĺžnikový pôdorys, pričom všetky steny každého domu ležia buď na poludníkoch alebo rovnobežkách (predpokladajte, že sídlisko sa nachádza na Slovensku a nie napríklad na severnom póle). V prvom riadku súboru domy.dat sa nachádza počet domov  $n$  ( $0 \leq n \leq 100$ ). Nasledujúcich  $n$  riadkov špecifikuje domy č.1, č.2, ..., až č. $n$ . Pre  $1 \leq i \leq n$ ,  $i$ -ty riadok udáva postupne zemepisnú dĺžku najzápadnejšieho poludníka, ktorý domom prechádza (označme  $x_i$ ), zemepisnú šírku najjužnejšej rovnobežky, ktorá domom prechádza (označme  $y_i$ ), dĺžku domu v smere rovnobežiek (označme  $dlzkax_i$ ), dĺžku domu v smere poludníkov (označme  $dlzkay_i$ ) a výšku domu v metroch (označme  $v_i$ ). Pre lepšiu orientáciu je zemepisná dĺžka vyjadrená počtom stotisícín stupňa od zemepisnej dĺžky najzápadnejšej steny na sídlisku, zemepisná šírka počtom stotisícín stupňa od dĺžky najjužnejšej steny na sídlisku a dĺžky stien domu v stotisícinách stupňov. Pri predpoklade, že sídlisko bude mať ľudské rozmery, bude teda približne:  $0 \leq x_i < 1000$ ,  $0 \leq y_i < 800$ ,  $0 < dlzkax_i < 1000$ ,  $0 < dlzkay_i < 800$  a  $0 < v_i \leq 50$ , pričom sú  $x_i$ ,  $y_i$ ,  $dlzkax_i$ ,  $dlzkay_i$  a  $v_i$  reálne čísla.

Na juh od sídliska sa nachádza pozorovateľ Santo. Je od sídliska vzdialený natoľko, že bočné steny domov sa javia také malé, že ich jeho oko zanedbáva. To znamená, že keď urobí napríklad krok doľava vidí sídlisko presne také isté ako predtým.

ÚLOHA: Napíšte program, ktorý vypíše všetky čísla domov, z ktorých Santo aspoň kúsok vidí. Čísla budú vypísané v utriedenom poradí.

Keď za sebou stoja dva rovnaké domy, zadný nevidno. Pri sto prvkoch nie je quicksort najrýchlejšie triedenie

## 1115. O ACM

30

ACM je programátorská súťaž, ktorá je zaujímavá tým, že jediným rozhodcom na nej je počítač. Úlohy, ktoré majú súťažiaci riešiť majú do posledného bajtíku predpísaný tvar vstupu a výstupu. Riešenia nesmú nič čítať ani vypisovať na terminál, všetko čítajú zo vstupného súboru meno.in a vypisujú do výstupného súboru meno.out (každá úloha má vopred pevne stanovené meno).

Riešiteľ, ktorý sa už nazdáva, že nejakú úlohu vyriešil, nahrá ju so stanoveným menom na disketu a zanesie na testovanie. Opravovateľom oznámi číslo svojho družstva, názov riešenej úlohy a jazyk (PASCAL alebo C), v ktorom je úloha napísaná. Opravovatelia založia jeho disketu do mechaniky testovacieho počítača a spustia špeciálny testovací program. Ten skopíruje riešenie z diskety na disk testovacieho počítača, preloží ho pomocou dohodnutého prekladača (tpc, tcc, bcc) a spustí s vopred pripraveným vstupným súborom (ktorý riešiteľ nepozná). Ak testovaný program nevypočíta výsledky v stanovenom časovom limite, testovací program oznámi, že riešenie je neefektívne, inak výsledok porovná s vopred pripraveným vzorovým výstupom. Ak si úplne zodpovedajú (bajt po bajte) oznámi, že riešenie testom prešlo a je OK. Inak oznámi chybu.



ÚLOHA: Navrhnite testovací program pre súťaž ACM. Máte teda vyrobiť balík programov spolu s návodom na používanie (!), ktorý v konečnom dôsledku umožní organizátorom súťaže testovanie správnosti riešení.

Podmienky:

1. Testovanie prebieha na PC-AT s operačným systémom DOS 5.0
2. Opravovatelia majú dopredu pomenovanú každú úlohu najviac osemznakovým menom, pripravené vzorové vstupy `meno.in` a vzorové výstupy `meno.tst` (tie sa budú porovnávať s `meno.out`, ktoré vytvárajú riešenia) a odhadnuté časové limity pre jednotlivé úlohy.  
Môžete (ale nemusíte) predpokladať nasledovné obmedzenia:
3. Testované programy sú v jedinom jazyku napríklad v PASCALe.
4. Testované programy sú preložiteľné (kompilátor ich úspešne preloží).
5. Testované programy bežia v reálnom čase (nie je potreba násilne prerušiť ich beh).
6. Nezáleží na čase riešenia, každé riešenie je efektívne.

V prípade nejasností sa obráťte na THELP.

#### 1121. O sťahovaní študentov

[21]

Na internáte je  $n$  dvojposteľových izieb, v ktorých býva  $2n$  študentov. Každému študentovi je pridelené číslo internátneho preukazu (tzv. ČIP, od 1 po  $2n$ ) a číslo izby, kde je ubytovaný. Keďže boli títo študenti ubytovaní mimoriadne demokratickým spôsobom, málokto je spokojný so svojim spolubývajúcim.

ÚLOHA: Napíšte program, ktorý načíta počet izieb  $n$ , pole `Izba[1..2n]`, kde `Izba[i]` je číslo izby, na ktorej býva študent s ČIP  $i$  a pole `Kamarat[1..2n]`, kde `Kamarat[i]` je ČIP študenta s ktorým chce študent s ČIP  $i$  bývať. V prípade, že je niekomu jedno s kým chce bývať je jemu prislúchajúca hodnota v poli `Kamarat` nulová. Váš program vypíše postup preubytovania pozostávajúci zo správ typu študent ČIP  $i$ , izba  $j$  sa vymení so študentom ČIP  $ii$ , izba  $jj$ , alebo správou Nedá sa vyhovieť všetkým. Vo vašom riešení by nemali chýbať úvahy o optimálnosti použitého algoritmu.

#### 1122. Binár

[17]

Napíšte program, ktorý načíta kladné číslo  $n$  a vypíše nasledujúci riadok:

„dvojkový zápis  $n$ “ (2) = „desiatkový zápis  $n$ “ (10)

Napríklad pre  $n = 10$  vypíše: 1010(2) = 10(10).

Obmedzenie: V celom programe môžete použiť iba procedúry bez parametrov a jedinú jednoduchú globálnu premennú.

#### 1123. O byrokratoch

[23]

Kde bolelo, tam bolelo, bola raz jedna krajina, v nej kopa byrokratov a bolelo to dosť. Ak človek niečo chcel, musel najprv získať mnoho pečiatok, a na to, aby dostal niektorú pečiátku, musel získať najprv mnoho iných. Našťastie tam boli i také pečiatky, ktoré sa dali získať samostatne.

ÚLOHA: Pomôžte ľuďom v tejto nešťastnej krajine a napíšte pre nich program, ktorý načíta počet pečiatok, pre každú pečiátku nulou ukončený zoznam pečiatok, ktoré treba na to, aby ste ju získali, ďalej načíta číslo pečiatky  $P$  a vypíše všetky postupy ako možno pečiátku  $P$  získať (prípadne smutnú správu **Prestahujte sa do inej krajiny**, ak sa pečiátka získať nedá).

Napríklad pre vstup: 5 2 4 3 0 5 3 0 4 0 0 1 je výstup: (5 4 3 2 1) (4 5 3 2 1) (4 3 5 2 1)

#### 1124. O ilegálnej organizácii

► [25]

$n$  ľudí pracuje v ilegálnej organizácii. Ako sa na ilegálov patrí, každý z nich pozná najviac troch ďalších. Neoslovujú sa menom, ale zásadne číslami od 1 po  $n$ .

Dôležitý ilegál je taká osoba, ktorej zadržanie poriadkovými silami spôsobí rozpadnutie organizácie aspoň na dve časti, medzi ktorými nie je žiadne spojenie.

ÚLOHA: Napište program, ktorý načíta počet ilegálov  $n$ , ďalej pre každého ilegála čísla spolupracovníkov, ktorých pozná, a zistí, či je medzi ilegálmi aspoň jeden dôležitý ilegál.

### 1125. Chodiace písmenká

21

Iste poznáte vírus Padajúce písmenká. Vašou úlohou bude naprogramovať podstatnú časť jeho novej verzie, ktorej dáme pracovný názov Chodiace písmenká. Každú sekundu si každé písmenko na obrazovke vygeneruje náhodne smer, ktorým sa chce uberať. Všetky písmenká sa potom naraz(!) pohnú svojím smerom. Keď chce viac písmeniek vstúpiť na to isté miesto, náhodne sa rozhodne ktoré to bude a ostatné ostanú stáť.

Napríklad z  $\begin{matrix} AB \\ CD \end{matrix}$  kde  $\begin{matrix} A \text{ chce ísť doprava,} \\ C \text{ chce ísť hore,} \end{matrix}$   $\begin{matrix} B \text{ chce ísť dole} \\ D \text{ chce ísť doľava} \end{matrix}$  bude  $\begin{matrix} CA \\ DB \end{matrix}$ .

ÚLOHA: Urobte program, ktorý po spustení realizuje jedno posunutie písmenok.

Ďalej môžete, ale nemusíte urobiť jeho rezidentnú verziu, ktorá sa zavesí na prerušenie od systémových hodín, chodenie písmeniek aktivuje o 12.00 a potom každú sekundu posúva písmenká (písmenká idú na obed).

### 1131. O osemsmerovke

30

Sú dané: rozmery osemsmerovky  $n, m$ , matica znakov  $A[1..n, 1..m]$  reprezentujúca osemsmerovku, počet slov  $k$  a  $k$  slov osemsmerovky.

ÚLOHA: Napište program, ktorý osemsmerovku, zadanú týmito údajmi vyrieši. (Riešenie osemsmerovky sa číta po riadkoch od najvrchnejšieho a od najľavejšieho znaku v riadku.) Predpokladajte, že žiadne slovo, ktoré sa bude škrtáť, nie je prefixom iného (napríklad, nie sú tam naraz slová „ale“ a „alebo“).

### 1132. Zátvorky

17

Napište program, ktorý zo vstupu číta postupnosť znakov obsahujúcu len '(', ')', '<', '>', '{', '}' ukončenú medzerou ' ' a vypíše ANO/NIE podľa toho, či je táto postupnosť dobre uzátvorkovaným výrazom.

Obmedzenie: jediná premenná typu char.

### 1133. O stupni čísla

23

Pre prirodzené čísla (a nulu) definujeme stupeň čísla  $x$  nasledovne:

$$\text{st}(x) = \begin{cases} \text{st}(0) = 0 \\ \text{st}(x) = 1 & \text{ak } 1 \leq x \leq 9 \\ \text{st}(x) = \text{st}(f(x)) + 1 & \text{ak } x \geq 10 \end{cases}$$

kde  $f(x)$  je súčin všetkých cifier čísla  $x$  (v desiatkovej sústave).

ÚLOHA: Napište program, ktorý pre dané  $k$  vypíše najmenšie číslo stupňa  $k$ .

Tieto čísla  $s_k$  veľmi rýchlo rastú, snažte sa napísať časovo čo najefektívnejší program. Nemusíte robiť aritmetiku veľkých čísel, pre hodnotenie je podstatný algoritmus.

### 1134. O Kocúrkove

75

V Kocúrkove majú  $n$  domov, pričom chcú postaviť cesty medzi niektorými dvojicami domov. Keďže Kocúrkovčania neoplývajú zbytočnou inteligenciou, nemajú radi križovatky, lebo na nich často zabúdlia. Mali by sme im pomôcť zistiť, či sa dajú cesty medzi domami postaviť tak, aby sa žiadne dve neprekrížili.

ÚLOHA: Napište program, ktorý načíta počet domov  $n$ ,  $k$  dvojíc domov, ktoré majú byť spojené cestou a vypíše, či sa dajú medzi nimi postaviť cesty bez križovatiek. Pomôcka: v dvoch nasledujúcich prípadoch sa to nedá urobiť:

1. prípad:  $n = 5$ ,  $k = 10$ , dvojice:  $[1, 2]$ ,  $[1, 3]$ ,  $[1, 4]$ ,  $[1, 5]$ ,  $[2, 3]$ ,  $[2, 4]$ ,  $[2, 5]$ ,  $[3, 4]$ ,  $[3, 5]$ ,  $[4, 5]$ .
2. prípad:  $n = 6$ ,  $k = 9$ , dvojice:  $[1, 4]$ ,  $[1, 5]$ ,  $[1, 6]$ ,  $[2, 4]$ ,  $[2, 5]$ ,  $[2, 6]$ ,  $[3, 4]$ ,  $[3, 5]$ ,  $[3, 6]$ .

## 1135. O formátovači

► 30

Iste ste sa už naučili, že programy je dobre písať v nejakom presne stanovenom formáte, napriek tomu, že kompilátoru to je jedno. Problém vzniká, keď určitý programový produkt robí viac programátorov, ktorí majú odlišné spôsoby písania programov. V takýchto programoch možno použiť program zvaný formátovač, ktorý program zapísaný ľubovoľným spôsobom prepíše do zvoleného formátu. Napríklad program vľavo upraví na program vpravo

<pre>var s:string; begin   readln(s);   if s='KSP' then     begin       write('to uz ano');       writeln     end   else writeln('nic zaujimave'); end.</pre>	<pre>var s : string; begin   readln (s);   if s = 'KSP' then begin     write ('to uz ano');     writeln   end else     writeln ('nic zaujimave') end.</pre>
---	---

ÚLOHA: Navrhnete a naprogramujete formátovač pre vami zvolený jazyk (PASCAL alebo C). Kľúčovým miestom vášho riešenia bude analýza možností zápisu programov a z nej vychádzajúci návrh spôsobu zadávania formátu (formátovač by mal umožniť užívateľovi zadať ním používaný tvar).

Zadávať tvar ide samozrejme len do istej miery. Stačí štandardný PASCAL, C. Môžete predpokladať, že formátovaný program je syntakticky správny.

## 1141. O permutáciách II

20

Napište program, ktorý na vstupe prijme čísla  $n$ ,  $k$  ( $k \leq n!$ ) a vypíše  $k$  náhodne vybraných permutácií čísel  $1, 2, 3, \dots, n$ . Pritom nesmie jednu permutáciu vypísať viackrát a pravdepodobnosť výberu vypísanej  $k$ -tice musí byť rovnaká ako pravdepodobnosť výberu ľubovoľnej inej  $k$ -tice. Pokúste sa zdôvodniť správnosť Vášho algoritmu.

## 1142. O delení siedmimi

19

Napište program, ktorý na vstupe prijme z riadku postupnosť čífer a vypíše ANO alebo NIE podľa toho, či je číslo (v desiatkovej sústave) zostavené z týchto číslíc deliteľné siedmimi alebo nie. Čífy sú v opačnom poradí, t.j. prvá ide cifra pri  $10^0$ , ďalej cifra pri  $10^1$ ,  $10^2$ , ... atď. Za poslednou cifrou (ktorou je cifra pri najvyššej mocnine 10, rôzna od nuly) už nie sú v riadku žiadne medzery (t.j. hneď po prečítaní poslednej cifry nastane EOLN).

Na program sa kladú dve obmedzenia: program môže používať len jednu premennú typu `integer` a program musí dokázať spracovať i 40 ciferné číslo za niekoľko stotín sekundy.

Pomôcka:  $(a \cdot b) \bmod c = ((a \bmod c) \cdot (b \bmod c)) \bmod c$

## 1143. O prešmyčkách

23

Slovo LODE je prešmyčka zo slova DELO alebo aj zo slova DOLE. Vašou úlohou bude napísať program, ktorý na vstupe prijme počet slov  $k$  a  $k$  slov a rozdelí tieto slová do skupín tak, že v každej skupine je každé slovo prešmyčkou ostatných. Žiadne slovo nie je dlhšie ako 10 znakov a  $k \leq 3000$ . Slová patriace do jednej skupiny budú vypísané v jednom riadku, slová nepatriace do jednej skupiny budú v rôznych riadkoch. Poradie slov v skupine bude rovnaké ako na vstupe. Podobne bude i poradie prvých slov skupinách bude rovnaké ako na vstupe.

Napríklad pre  $k = 10$  a slová: LODE MASO DELO SAKO OSMA DOLE SAMO KOSA PETRA PATER program vytvorí skupiny:

LODE DELO DOLE  
MASO OSMA SAMO  
SAKO KOSA  
PETRA PATER

**1144. O poradí**

[21]

V súťaži ACM sú riešenia súťažiach testované dvojkoľovo. Najprv sa v prvom kole priebežne testujú všetky riešenia (niektoré i viackrát) a potom sa lepšie z nich o testujú v druhom kole. Za každé testovanie dostane súťažiaci určitý počet bodov. Testy druhého kola sú zamerané na špeciálne prípady, takže o konečnom poradí rozhodujú body udelené v prvom kole a len v prípade rovnosti bodov z prvého kola rozhodujú body z kola druhého. V prípade, že bol program otestovaný v prvom kole viackrát berie sa za platný najvyšší dosiahnutý počet bodov. Testovania neprebiehajú na jednom mieste, takže keď sa výsledky testovaní prinesú na vyhodnotenie, tvoria súbor, zložený z viet tvaru: Meno (20B), body (4B), kolo (1B), ktoré sú úplne pomiešané. Napíšte program, ktorý načíta súbor uvedeného tvaru a vypíše výsledkovú listinu súťaže ACM.

**1145. O preprocesore**

[27]

Preprocesor je program, ktorý umožňuje užívateľovi používať pri písaní programu v nejakom programovacom jazyku vlastnú syntax. Kód obsahujúci nielen zápisy v programovacom jazyku, ale i direktívy definujúce vlastnú syntax a zápisy v tejto syntaxi preprocesor preloží do kódu, ktorý bude obsahovať iba zápisy v programovacom jazyku. Ten sa potom dá použiť na ďalšie spracovanie. Preprocesor nie je závislý na používanom programovacom jazyku, závislý je iba od spôsobu definovania vlastnej syntaxe. Náš preprocesor bude rozumieť jedinej direktíve tvaru

```
\def\výraz1{výraz2}
```

Kdekoľvek za touto direktívou použijeme v kóde `\výraz1`, bude tento výraz nahradený výrazom `výraz2`. Má to však jeden háčik. Výraz bude môcť obsahovať aj podvýrazy `?X *X`, kde `?X` predstavuje ľubovoľný znak a `*X` ľubovoľný neprázdny reťazec znakov. Veľké písmeno nasledujúce za `? a *` bude slúžiť na označovanie znaku alebo reťazca, aby sme mohli vo výraze `výraz2` upresniť, kde sa má tento výraz premiestniť. Aby sme mohli vo výraze používať i znaky `?, *, { a }` budeme ich písať ako `\?, \*, \{ a \}`. Potom znak `\` musíme písať ako `\\`.

Pri používaní direktív musí byť priradenie znakov a reťazcov symbolom `?X, *X` vždy jednoznačné a nikdy nesmie dôjsť k zacykleniu.

Takže napríklad kód:

```
\def\ a(?X,*Y){a(?X,\ a(*Y))}\\\def\ a(?X){?X}\ a(1,2,3,4,5,6)
```

bude preložený preprocesorom ako

```
\ a(1,\ a(2,\ a(3,\ a(4,\ a(5,6))))))
```

Naprogramujte vyššie popísaný preprocesor a zamyslite sa nad tým ako by sa dal tento preprocesor zdokonaľiť.

**1211. O Martowovi**

[18]

Raz Matematicko-fyzikálnu fakultu navštívil len tak mimochodom mimozemšťan Martow a zanechal tam na pamiatku  $n$ -rozmernú kocku. Študenti sa rozhodli ju dôkladne preskúmať (to znamená každý jej vrchol), ale, ako to už s  $n$ -rozmernými kockami býva, veľmi

rýchlo stratili orientáciu, ukazovali jeden cez druhého a hádali sa, ktorý vrchol už skúmali a ktorý ešte nie.

Preto sa rozhodli, že kocku budú skúmať systematicky: dôležité je, aby preskúmali skutočne všetky vrcholy, každý práve raz. Aby sa v kocke nestratili, musia vždy na ďalší skúmaný vrchol prejsť prstom po hrane.

ÚLOHA: Napíšte program, ktorý načíta číslo  $n$  a vypíše postupnosť súradníc vrcholov v takom poradí, ako ich môžu študenti skúmať.

$n$ -rozmerná kocka sa skladá z  $2^n$  vrcholov so súradnicami  $(x_1, x_2, \dots, x_n)$ , kde  $x_i$  je 0 alebo 1. Dva vrcholy sú spojené hranou práve vtedy, keď sa ich súradnice líšia práve v jednej zložke. Typickým príkladom  $n$ -rozmernej kocky je štvorec ( $n = 2$ ).

#### 1212. O brčkavých zátvorkách

[20]

„Vnorené!“, kričal Brutus, ako zmyslov zbavený. „Nevnorené!“, oponoval Frutus. „Ty sám si nevorené“, Brutus na to. „Tvoju hlavu plešivú!“, povedal Frutus. Medzitým kamaráti vyšli pred krčmu a Frutus vybil Brutovi zub.

O čom sa to Brutus a Frutus tak vášnivo rozprávali? Hádali sa, aké zátvorkovanie v komentároch používa nová verzia  $\omega$ -pascalu v.8.3 $\beta$ . V kompilátore s nevoreným zátvorkovaním je za poznámku považovaná časť od ľavej komentárovej zátvorky ( $\{$ ) až po najbližšiu pravú komentárovú zátvorku ( $\}$ ) - takto sa to napríklad kompiluje v Turbo Pasmale. V kompilátore s vnoreným zátvorkovaním je za poznámku považovaná časť od ľavej komentárovej zátvorky po jej zodpovedajúcu pravú komentárovú zátvorku - takýmto spôsobom skompilujeme ako poznámku napríklad toto:  $\{\{\{\text{Hello world!}\}\}\}$ .

ÚLOHA: Napíšte program, ktorý môže byť skompilovaný oboma druhmi kompilátorov pascalu a po spustení vypíše  $\text{\texttt{ANO}}$ , ak bol skompilovaný kompilátorom s vnoreným zátvorkovaním a  $\text{\texttt{NIE}}$ , ak bol skompilovaný kompilátorom s nevoreným zátvorkovaním.

#### 1213. O Pažravom Riškovi

[23]

Pažravý Riško je na prázdninách u svojej svokry na Jahodníku. Keďže na Jahodníku rastie veľa jahôd, má svokra v komore na polici  $n \times n$  jahodových kompótov rôznych objemov poukladaných do štvorca. Riško chce zjesť čo najviac jahôd z kompótov (je lenivý si ísť nazbierať jahody sám) tak, aby si to svokra nevšimla (toto je možné zabezpečiť tak, že z každého štvorca  $2 \times 2$  kompótov bude zjedený najviac jeden kompót). Teraz chce vedieť koľko najviac jahôd môže zjesť.

ÚLOHA: Zostavte program, ktorý načíta  $n$  a pole  $n \times n$  objemov kompótov a vypíše maximálny objem kompótov, ktoré môže Riško zjesť tak, aby si to svokra nevšimla.

#### 1214. O ovocnom sade

[21]

Santo a Banto vlastnia parcelu  $n \times n$ , na ktorej rastú jablone a hrušky. Po veľmi krutej hádke sa rozhodli o parcelu súdiť. Sudca spravodlivo rozhodol, že si majú svoju parcelu rozdeliť na dve súvislé časti rovnaké tvarom aj obsahom tak, aby na Santovej časti rástli iba jablone a na Bantovej iba hrušky (lebo Santo má rád jablkovicu a Banto zasa hruškovicu). Santo a Banto ako obvykle vymeriavali, zatĺkali kolíky, ale doteraz sa im parcelu rozdeliť nepodarilo.

ÚLOHA: Napíšte program, ktorý načíta  $n$  a pole čísel  $n \times n$ , v ktorom jednotky reprezentujú jablone, dvojky hrušky a nuly nevysadené miesta a vypíše pole čísel  $n \times n$ , pričom jednotky budú reprezentovať Santove pozemky a dvojky Bantove pozemky.

Časť je súvislá, keď medzi každými jej dvoma políčkami  $i, j$  existuje cesta  $v_1, v_2, \dots, v_k$  taká, že  $v_1 = i$  a  $v_k = j$  a pre všetky  $l = 1 \dots k - 1$  políčka  $v_l$  a  $v_{l+1}$  susedia hranou.

#### 1215. O kvalitnej tlačiarňi

[21]

V softférovom družstve SoDr si kúpili novú tlačiareň. Vyznačuje sa tým, že tlačí veľmi kvalitné písmenká, čo vyžaduje veľa pamäti a preto si dokáže zapamätať len tvary najviac

k písmeniek. Ak teda tlačia na takejto tlačiarňi nejaký súbor, musia veľmi často použiť operáciu  $DownLoad(x, y)$ , kde  $x$  je miesto v pamäti tlačiarne ( $1 \dots k$ ), kam sa má uložiť tvar písmenka  $y$  (písmenko, ktoré bolo na tomto mieste sa automaticky vymaže z pamäti tlačiarne). Ak potom tlačiareň má vytlačiť nejaké písmenko, vie, kde ho v pamäti nájde, dôležité je však, aby tam bolo (v opačnom prípade sa tlačiareň zasekne). Na začiatku je v pamäti náhodný obsah (a teda ho nemožno nijakým spôsobom využiť). Operácia  $DownLoad$  je veľmi pomalá a preto je potrebné jej použitie minimalizovať.

ÚLOHA: Napíšte program, ktorý načíta číslo  $k$  a meno súboru a vytlačí tento súbor na tlačiarňu družstva SoDr. Predpokladajte, že máte už predprogramovanú procedúru  $DownLoad$  tak, ako je popísaná vyššie.

Pokúste sa zamyslieť a napísať aj niečo o tom, ako efektívne (čo sa týka počtu  $DownLoad$ ov) váš program pracuje pre rôzne typy súborov.

### 1221. O výskumnom ústave

[23]

Santova manželka Santovka pracuje vo Výskumnom ústave pre označovanie výrobkov. Jej najnovší výskum sa zaoberá novým druhom zariadenia, ktoré na značený výrobok tlačí  $n$ -ticu rôznych číslic vždy v inom poradí. Prístroj ešte nie je úplne automatizovaný. Preto musí Santo, ktorý ho skúša, meniť poradie číslic ručne. Santovka prikázala Santovi, aby do večera vytlačil na papier všetky možné permutácie, ktoré môže stroj vytlačiť (ak chce dostať večeru). Santo je veľmi lenivý a rozhodol sa, že dve bezprostredne za sebou vytvorené permutácie sa budú líšiť jednou výmenou susedných prvkov. Teraz sedí na zemi a rozmýšľa, ako to môže urobiť.

ÚLOHA: Napíšte program, ktorý pre dané  $n$  vypíše všetky permutácie  $n$  prvkov tak, aby sa každé dve susedné permutácie líšili jednou výmenou dvoch susedných prvkov.

### 1222. O indiánskom nárečí

[35]

V Brazílskom pralese žije skupina Indiánov, ktorých jazyk je naozaj veľmi zvláštny. Dve slová sú považované za zhodné, ak je jedno cyklickým posunom druhého. Napríklad  $ABC$  a  $BCA$  sú zhodné slová, ale  $ABC$  a  $CBA$  nie sú. Jazykovedec sa ho rozhodol naučiť. Najväčšie problémy má zo zisťovaním, či sú dve slová zhodné.

ÚLOHA: Napíšte program, ktorý načíta dve slová a zistí, či sú v jazyku Indiánov zhodné, alebo nie.

### 1223. O PNI

[23]

V podniku na integrácie (PNI) vyrábajú rôzne obvody, ktoré sú charakterizované počtom vstupných nožičiek a počtom výstupných nožičiek. Dva integrácie možno zospájať do jedného (vznikne nám nový integráč), keď prvý má rovnaký počet výstupných nožičiek ako druhý vstupných nožičiek. Inžinieri zistili, že na overenie správnosti zospájovania treba pre každú dvojicu vstupná nožička nového integráču – výstupná nožička nového integráču odmerať napätie na všetkých spojoch. Jedno odmeranie trvá šikovnej meračke jednu sekundu. Celkovo, ak mal prvý integráč  $I_1$  vstupných a  $O_1$  výstupných nožičiek a druhý integráč  $I_2 (= O_1)$  vstupných a  $O_2$  výstupných nožičiek, bude zisťovanie správnosti trvať  $I_1 * O_1 * O_2$  sekúnd. V podniku podľa objednávky spájajú zákazníkovi  $n$  vhodných integráčov zaradom do jedného v požadovanom poradí, pričom ich spájajú vždy po dvojiciach a pri spájkovaní overujú správnosť. Vo výslednom produkte musia byť integrácie v požadovanom poradí, nie je však dôležité, v akom poradí ich spájkovali. Vhodné poradie spájovania môže výrazne znížiť čas potrebný na testovanie správnosti spojov.

ÚLOHA: Napíšte program, ktorý pre zadané  $n$ , načíta počty vstupných nožičiek integráčov  $I[1..n]$  počty výstupných nožičiek integráčov  $O[1..n]$  a vypíše poradie, v akom treba integrácie spájať, aby bol čas potrebný na testovanie minimálny. Zistite aj tento čas.

Napríklad pre vstup:  $n = 6$ ,  $I = [4, 2, 3, 1, 2, 2]$ ,  $O = [2, 3, 1, 2, 2, 3]$  je výsledkom takáto postupnosť spájania:

Spoj integráče B a C.  
 Spoj integráče A a BC.  
 Spoj integráče D a E.  
 Spoj integráče DE a F.  
 Spoj integráče ABC a DEF.  
 Na testovanie bolo treba 36 sekúnd.

#### 1224. O burundijských stavbároch

30

Po úmrtí starého kráľa nastal v krajine zmätok a mocné rody zápasili o trón. Po dlhotrvajúcich bojoch sa v krajine stal novým kráľom Ubugulundibumbuluku. Pri bojoch sa zničili všetky cesty medzi mestami. Ubugulundibumbuluku sa rozhodol, že vybuduje novú cestnú sieť, pričom presne určil, koľko ciest má do ktorého mesta viesť (podľa toho, ako mu miestne rody pomáhali pri ťažkom boji). Ešte chce, aby sa dalo dostať z každého mesta do každého. Stavbári teraz rozmýšľajú ako dané cesty postaviť, aby splnili kráľové podmienky. Keby aspoň vedeli, či sa to dá, hneď by sa im ľahšie rozmýšľalo. Preto si kúpili počítač a chcú program, ktorý im to zistí.

ÚLOHA: Napíšte program, ktorý pre zadané  $n$  a počty ciest, ktoré majú viesť do jednotlivých miest zistí, či je možné postaviť cestnú sieť tak, aby spĺňala kráľove požiadavky.

#### 1225. O počítačoch na mieru

25

V softvérovom družstve SoDr sa okrem iného vyrábajú aj tzv. počítače na mieru. Zákazník príde, povie si, čo by mal jeho počítač vedieť robiť, SoDráci napíšu potrebný softvér, nainštalujú ho na počítač a pošlú ho zákazníkovi.

Aby bol taký počítač čo najlacnejší, musí mať práve toľko pamäti, koľko dané programové vybavenie potrebuje (inak by to buď nefungovalo, alebo by počítač bol zbytočne drahý). Preto SoDráci pri každom svojom programe potrebujú zistiť, koľko vlastne pamäti potrebujú premenné v programe.

ÚLOHA: Napíšte program, ktorý spočíta veľkosť pamäti, ktorá sa použije na definíciu jednej premennej. Definícia premennej je zapísaná v takejto forme:

```

<id>      je sekvencia písmen dlhá najviac 255 písmen
<integer> je celé číslo medzi -10000 a 10000

<message> = <data>
<data> = <id> : <type>
<type> = <record> | <array> | <string> | <enum> | <range>
<record> = { <data>+ }
<array> = array[ <range> ] of <type>
<string> = string ( <integer> )
<enum> = ( <id-list> )
<id-list> = <id> | ( <id> , <id-list> )
<range> = [ <integer> .. <integer> ]
```

Samozrejme, počet bielych znakov<sup>15</sup> medzi jednotlivými skupinami písmen nehrá žiadnu úlohu<sup>16</sup>.

Počet bitov potrebných na uloženie sa počíta nasledovne:

```

record    súčet veľkostí jednotlivých položiek
array     počet prvkov poľa násobený počtom bitov potrebných na uloženie ich typu
string    dĺžka vynásobená siedmymi
enum      najmenší počet bitov, ktorými môžeme rozlíšiť všetky id
range     najmenší počet bitov, ktorými môžeme rozlíšiť všetky hodnoty z daného intervalu
```

<sup>15</sup> medzera, nový riadok, tabulátor

<sup>16</sup> ak sa vám táto definícia zdá príliš zložitá, porovnajte ju s definíciou premenných v PASCALe

Vstupom do programu je definícia premennej. Vstup je napísaný správne, netreba kontrolovať jeho syntaktickú korektnosť. Výstupom z programu je počet bitov potrebných na uloženie danej premennej. Napríklad

Vstup: *year* : [1970..2030]

Výstup: 6

Vstup:

```
team : {
  meno : string(14)
  clenovia : array[1..3] of {
    pohlavie : (muz, zena)
    meno : string(20)
    vek : [16..30]
  }
  umiestnenie : [1..40]
}
```

Výstup: 539

Vstup : *menaporotcov* : array[1..3] of string(20)

Výstup: 420

### 1231. Brutus, Frutus a partície

23

„6+3+1!“, kričal Brutus, ako zmyslov zbavený. „6+2+2!“, oponoval Frutus. „Ty sám si 6+2+2“, Brutus na to. „Tvoju hlavu plešivú!“, povedal Frutus. Medzitým kamaráti vyšli pred krčmu a Brutus vybil Frutovi zub.

O čom sa to Brutus a Frutus tak vášnivo rozprávali? Hádali sa, ako vyzerá desiatu partícia z čísla 10 v Peknom usporiadaní. Partíciou nazveme každý rozklad čísla na prirodzené sčítance usporiadané nerastúco (napríklad  $10 = 6 + 3 + 1$ ). Pekným usporiadaním partícií nazveme také usporiadanie, kde partícia  $a_1 + a_2 + \dots + a_k$  je pred partíciou  $b_1 + b_2 + \dots + b_l$  práve vtedy, keď pre nejaké  $u$  platí:

$$a_1 = b_1, a_2 = b_2, \dots, a_u = b_u, a_{u+1} > b_{u+1}$$

Napríklad partície čísla 6 zoradené v Peknom usporiadaní sú: 6, 5+1, 4+2, 4+1+1, 3+3, 3+2+1, 3+1+1+1, 2+2+2, 2+2+1+1, 2+1+1+1+1, 1+1+1+1+1+1.

ÚLOHA: Napište program, ktorý načíta čísla  $n$  a  $k$  a vypíše  $k$ -tu partíciu čísla  $n$  v Peknom usporiadaní.

Pravdu mal v tomto prípade Frutus.

### 1232. O Santových kartičkách

18

Kedysi dávno starý PrasaNto pre účely rodinnej hry vlastnoručne vystrihol 32767 kartičiek a napísal na ne čísla od 1 po 32767 (na každú inú). Už-už sa chcel pustiť do hry, keď tu zrazu zafúkal vietor a všetky kartičky rozfúkal po okolí. PrasaNto ich rýchlo pozbieral a keď ich spočítal, zistil, že jedna chýba.

Odvtedy sa vždy v sobotu večer zide celá rodina u Santovcov na verande a pokúšajú sa zistiť, ktorá kartička sa stratila, aby ju mohli znovu napísať a konečne si zahrať starú rodinnú hru. Santovi to samozrejme už lezie na nervy a do peňaženky (keďže vždy príde aj Banto a stiahne mu celý bar), tak chce tomuto zvyku učiniť rázny koniec.

ÚLOHA: Na vstupe je 32766 rôznych čísel od 1 po 32767. Napište program, ktorý načíta tieto čísla a v čo najkratšom čase a s čo najmenšou pamäťou nájde chýbajúce číslo (t.j. to číslo z intervalu  $\langle 1, 32767 \rangle$ , ktoré sa medzi číslami na vstupe nevyskytuje).



**1233. O Kiribatských aerolíniách**

[25]

Na Kiribatských ostrovoch sa rozhodli zanechať plavbu na trstinových lodičkách a prepojiť ostrovy leteckým spojením. Na každom ostrove vybudovali letisko. Problém je však v tom, že priame spojenia sú obmedzené dĺžkou doletu lietadla. Lietadlo síce môže urobiť medzipristátie na nejakom inom letisku, natankovať a pokračovať ďalej, ale týmto spôsobom sa predlžuje letová dráha (lietadlo medzi dvoma letiskami letí vždy po priamke, letová dráha medzi dvoma letiskami je ich priama vzdušná vzdialenosť).

ÚLOHA: Zostavte program, ktorý načíta súradnice jednotlivých letísk, dĺžku doletu lietadla a dve letiská  $s$  a  $t$  a vypíše najkratšiu možnú letovú dráhu z  $s$  do  $t$  (tj. postupnosť všetkých medzipristátí lietadla).

**1234. O modrých prilbách v Burundi**

► [25]

Kedže nárok kráľa Ubugulundibumbuluku na trón bol od začiatku veľmi pochybný, vytvorila sa v meste Bujumbura skupina obyvateľov, ktorí si zvolili vlastného prezidenta Ntybantuganyu a začali vytvárať územie novej Burundijskej republiky. Územie republiky má tvar mnohouholníka (nie nutne konvexného!).

Organizácia spojených národov sa v rámci nezasahovania do vnútorných záležitostí suverénnej krajiny rozhodla vyslať na územie Burundi špeciálne vycvičenú parašutistickú jednotku modrých prilieb. Keď takýto parašutista zoskočí na zem, musí rýchlo zistiť, či sa nachádza na území republiky, alebo kráľovstva.

ÚLOHA: Napíšte program, ktorý na vstupe dostane mnohouholník (súradnice jeho vrcholov v protismere hodinových ručičiek) a súradnice niekoľkých miest, do ktorých pristáli parašutisti a vypíše pre každé takéto miesto, či leží vnútri daného mnohouholníka, alebo nie.

**1235. O okienkovom systéme**

[30]

V softférovom družstve SoDr majú nového zákazníka, ktorý si kúpil počítačovú sieť so zbierkou najrôznejších terminálov najrôznejšieho typu, rozmeru a veku. Programátorom sa horko – ťažko podarilo naprogramovať niekoľko základných výstupných operácií tak, aby boli použiteľné na všetkých termináloch. To však nestačí, produkty softférového družstva SoDr sa totiž vyznačujú dobrým užívateľským rozhraním.

ÚLOHA: Napíšte pre softférové družstvo knižnicu pre prácu s oknami. Vaša knižnica by mala zvládnuť tieto funkcie:

Vytvorenie okna – vytvorí prázdne okno (t.j. žiadne rámčeky) v stanovenom obdĺžniku na obrazovke terminálu. Novovytvorené okno je ždy aktívne. Nezabudnite, že sa pritom môžu prekryť nejaké časti už vytvorených okien!

Zaktívnenie okna – dané okno sa „vysunie“ na povrch (to znamená, že žiadna jeho časť nebude zakrytá iným oknom).

Zápis znaku do aktívneho okna – na dané súradnice v okne (vzhľadom k ľavému hornému okraju) sa zapíše znak. Môžete počítať s tým, že sa výstup na obrazovku bude vykonávať iba pomocou tejto operácie.

Zatvorenie aktívneho okna – aktívne okno sa zruší (zmaže sa z obrazovky a obnovia sa všetky časti iných okien, ktoré zrušené okno zakrývalo)

Pre prácu s terminálom môžete použiť len tieto procedúry a funkcie. Deklarácie sú uvedené v jazyku Pascal, ak by ste programovali v inom jazyku, použite podobné.:

**procedure** *Write*(*ch:char*) – vypíše znak *ch* na aktuálnu pozíciu kurzora a posunie ho na nasledujúci stĺpec (prípadne prvý stĺpec nasledujúceho riadku).

**procedure** *GotoXY*(*x,y:byte*) – presunie kurzor na riadok *y* a stĺpec *x*.

**function** *GetRows:byte* – vráti počet riadkov terminálu

**function** *GetColumns:byte* – vráti počet stĺpcov terminálu

Nezabudnite, že ku knižnici treba dodať aj podrobný popis fungovania jednotlivých procedúr a použitých dátových štruktúr (teda nielen ich použitia).

**1241. O blšom tanci**

[30]

Riaditeľ hilbertovho cirkusu sa rozhodol rozšíriť program o nové číslo. Na konkurze zvíťazil Blší tanec. Tento tanec tancuje naraz  $n$  blšiek, označených číslami 1 až  $n$ , ktoré sú zostavené na políčkach očíslovaných od 1 po  $n$ . Z každého políčka vedie práve jedna šípka na nejaké políčko, pričom do každého políčka vedie práve jedna šípka.

Na začiatku každá blška sedí na políčku s rovnakým číslom ako má ona. Potom v každej sekunde preskočia blšky (všetky naraz) po šípkách na svoje nové políčka. Tanec končí, keď sú blšky rozmiestnené tak, ako na začiatku. Riaditeľ potrebuje, aby číslo trvalo najdlhšie ako sa dá a preto treba vhodne nakresliť šípky.

ÚLOHA: Napíšte program, ktorý pre dané  $n$  vypíše, ako treba nakresliť šípky, aby číslo trvalo najdlhšie ako sa dá. Program by mal tiež vypísať, ako dlho bude číslo trvať.

Pre  $n = 5$  je výsledok 1-2 2-1 3-4 4-5 5-3 a číslo bude trvať 6 sekúnd. Pre  $n = 8$  je výsledok 1-2 2-3 3-1 4-5 5-6 6-7 7-8 8-4 a číslo bude trvať 15 sekúnd.

**1242. O továrni na ceruzky**

[22]

V továrni na ceruzky HOK-NI-RO vyrábajú veľmi kvalitné ručne vyrezávané ceruzky rôznych dĺžok. Potom ich balia po  $n$  kusoch, pričom v krabičke sú ceruzky vždy usporiadané podľa dĺžky. V rámci automatizácie kúpili do továrne triedičku. Triedička je vybavená pravitkom na meranie dĺžok ceruziek a rukou, ktorá umožňuje vymeniť dve ceruzky. Problém je v tom, že ruka je poškodená a dokáže vymeniť iba dve také ceruzky, medzi ktorými je práve jedna iná ceruzka. V továrni potrebujú vedieť, či pre danú skupinu ceruziek sa tieto dajú usporiadať na triedičke.

ÚLOHA: Napíšte program, ktorý pre dané  $n$  a  $n$  dĺžok ceruziek vypíše, či sa tieto dajú usporiadať na triedičke.

Pre  $n = 4$  a dĺžky ceruziek 2, 3, 1, 4 sa ceruzky nedajú usporiadať. Pre  $n = 3$  a dĺžky 3, 2, 1 sa ceruzky dajú usporiadať.

**1243. O veľkej poľovačke**

[27]

Rodina Santovcov sa pohádala s rodinou Bantovcov kvôli portréту starej Santovky. Rodiny vytiahli staré pušky a nastala poľovačka.

Situácia v meste sa stala natoľko neprehľadná, že šerif o občanoch nevie, do ktorej rodiny patria. Aby to zistil poslal svojich pomocných šerifov na pozorovacie stanovištia, aby mu odtiaľ posielali informácie, kto na koho vystrelil (nemusel nutne trafiť). Z týchto informácií chce šerif zistiť príslušnosť jednotlivých občanov k rodinám. Šerif vie, že konfliktu sa zúčastňujú iba dve rodiny a že vo vnútri rodiny nie sú žiadne spory (t.j. členovia jednej rodiny na seba nestrielať).

ÚLOHA: Napíšte program, ktorý načíta  $n$  dvojíc mien (každá dvojica mien obsahuje informáciu, kto na koho vystrelil) a vypíše rozdelenie občanov do rodín. V prípade, že informácie sú nedostačujúce na jednoznačné zaradenie členov do rodín, program vypíše: MÁLO INFORMÁCIÍ. V prípade, že sa podľa daných informácií nedajú rozdeliť občania do dvoch rodín, program vypíše: CHYBNÉ INFORMÁCIE.

Vstup: Santo,Banto Banto,Jim Jim,Santo

Výstup: CHYBNÉ INFORMÁCIE

Vstup: Santo,Banto Jim,Bill

Výstup: MÁLO INFORMÁCIÍ

Vstup: Mary,Santo Bill,Banto John,Jim Bill,John Mary,Bill

Výstup: Santo,Jim,Bill vs John,Mary,Banto

**1244. O n-trise**

[27]

Programátor Ó Veľký sa rozhodol napísať modifikáciu známej hry tetris. Od pôvodného tetrisu sa nová verzia bude líšiť v tom, že padajúce dieliky sa nebudú skladať zo štyroch,

ale z  $n$  štvorčekov. Teraz už tretí deň sedí a na štvorčekový papier kreslí všetky možné dieliky, ktoré sa dajú zložiť z  $n$  štvorčekov a stále si nie je istý, či ich má všetky.

ÚLOHA: Napište program, ktorý pre dané  $n$  vypíše všetky dieliky, ktoré sa dajú zložiť z  $n$  štvorčekov a ich počet. Dieliky vzniknuté rotáciou považujeme za rovnaké.

Pre  $n = 4$  program vypíše:

```
## #      #  ##  ##  #
## ### ### ##   ##  ####  ####
```

Existuje 7 rôznych dielikov.

#### 1245. O veľkých číslach

► [30]

Raz si u softvérového družstva SoDr pracovníci Matematicko — matematickej fakulty Matematickej univerzity objednali program, ktorý by im umožnil pracovať s veľkými číslami. Čísla s ktorými chcú pracovať nie sú dlhšie ako 257 cifier, môžu byť kladné aj záporné.

ÚLOHA: Napište pre pracovníkov MMF MU knižnicu, ktorá im umožní pracovať s celými číslami, ktorých desiatkový zápis môže byť dlhý až 257 cifier. Vaša knižnica musí obsahovať tieto procedúry a funkcie (deklarácie sú uvedené pre jazyk Pascal, ak by ste programovali v inom jazyku, použite podobné):

$GAdd(a, b, c, err)$  – sčítanie ( $c := a + b$ )

$GSub(a, b, c, err)$  – odčítanie ( $c := a - b$ )

$GMul(a, b, c, err)$  – násobenie ( $c := a \times b$ )

$GDiv(a, b, c, r, err)$  – celočíselné delenie ( $c := a \div b; r := a \bmod b$ )

$GAbs(a, c)$  – absolútna hodnota čísla ( $c := \text{abs}(a)$ )

$GInput(a, err)$  – vstup veľkého čísla  $a$

$GOutput(a)$  – výstup veľkého čísla  $a$

Vo všetkých prípadoch premenná  $err$  po návrate z procedúry obsahuje nulu, ak operácia prebehla v poriadku a kód chyby, ak počas operácie nastala nejaká chyba (napríklad: výsledok presiahol rozsah veľkých čísel apod.)

#### 1311. O kiribatských klebetniciach

[23]

Kiribatské ženičky rady klebetia. Problém je v tom, že keď si chce jedna Kiribatčanka poklebetiť s inou, zväčša sa musí doplaviť na ostrov, kde býva.

Žiaľ, iba medzi niektorými dvojicami ostrovov funguje kyvadlová prievoznícka doprava a tak ženička v túžbe po nových zaujímavých informáciách musí často aj niekoľkokrát prestupovať. Navyše niektorí prievozníci sú strašne leniví a tak je ich loďka tak pomalá, že ženička zvolí radšej inú cestu aj za cenu prestupovania. Ženičky by potrebovali takú tabuľku, ktorá by povedala, ako dlho trvá najkratšia možná cesta pre každú dvojicu ostrovov.

ÚLOHA: Napište program, ktorý načíta počet ostrovov  $n$  a zoznam kyvadlových „liniek“ (každá linka je určená dvoma číslami ostrovov  $1 \dots n$  a časom prevozu) a vypíše tabuľku, v ktorej pre každú dvojicu ostrovov bude uvedený čas najkratšej novej cesty. Predpokladajte, že čas potrebný na čakanie na kyvadlovú dopravu je zanedbateľný.

Napríklad pre  $n = 3$  a linky (1,2,10) (1,3,34) (2,3,100) je výsledok:

```
0 10 34
10 0 44
34 44 0
```

#### 1312. O SoDr-e

[17]

V softvérovom družstve SoDr majú opäť kopec problémov s novým zákazníkom. Ten si kúpil počítač, ktorý mal k dispozícii iba jedinú premennú, do ktorej je možné uložiť jediný znak.

Zákazník chcel umiestniť počítač na hranicu. Keď prejde niekto z jednej strany hranice na druhú, počítač dostane na vstup znak **A**, ak prejde z druhej strany na prvú, počítač

dostane na vstup znak B. Ak je večer na vstupe rovnaký počet písmen A aj B, každý je doma a všetko je v poriadku. V opačnom prípade je potrebné urýchlene vyhlásiť poplach.

ÚLOHA: Na vstupe je postupnosť znakov A a B ukončená medzerou. Napíšte program, ktorý zistí, či je v postupnosti rovnaký počet písmen A aj B. Nezabudnite, že váš program smie používať jedinú premennú typu **char** a aj jej obsah možno meniť iba načítaním novej hodnoty zo vstupu.

### 1313. O malom lenivom krtkovi II

25

Malý lenivý krtko II. spal v brlôžku až do večere. Jeho mamičku to nazlostilo a tak si povedala: „Čo z môjho synčeka vyrastie, keď bude taký lenivý!“, a už ho aj išla budiť. „Vstávaj krtko II.!,“ rázne sa mu prihovarila, „ak dneska nepochytáš chrobáčekov vo všetkých našich brlôžkoch, tak ťa naskutku paličkou II. vyobšívam!“ Čo mal chudák krtko II. robiť, nechcelo sa mu, ale musel vstať.

Krtia rodina mala vybudovaných  $n$  brlôžkov, pričom každé dva brlôžky boli spojené tunelmi. Každý deň krtica s krtom preliezli všetky brlôžky a zbierali to, čo majú najradšej – chrobáčky. Teraz mal aj malý lenivý krtko II. zbierať. Ale keďže bol celkom inteligentný, najprv pouvažoval: „Najlepšie, keď cez každý brlôžtek prejdem práve raz a pôjdem po takých chodbách, že súčet ich dĺžok bude najmenší.“ Samozrejme malý lenivý krtko II. chcel svoju cestu skončiť opäť vo svojom brlôžku, aby mohol spať až do ďalšej večere.

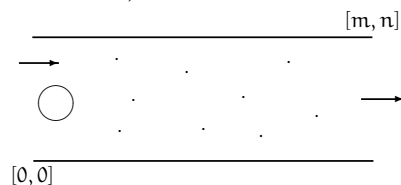
ÚLOHA: Napíšte program, ktorý načíta počet brlôžkov  $n$  a dĺžku tunela medzi každými dvoma brlôžkami a vypíše, aká dlhá bude cesta malého lenivého krtka II.

### 1314. O valcočlovekovi II

▶ 27

Valcočlovek je taký človek, ktorého tvar sa približuje tvaru valca s určitou výškou a polomerom. Výskyt ľudí takéhoto tvaru je veľmi vysoký napríklad medzi pracovníkmi francúzskych vínnych pivníc (čím viac vína pracovník skonzumuje, tým je širší). Francúzske vinne pivnice sú veľké podzemné sály tvaru obdĺžnika podopierané tenkými stĺpmi. Strany obdĺžnika sú orientované v smere svetových strán. Často sa stáva, že valcočlovek sa zasekne medzi stĺpy a vína pivnica sa stane nepoužívateľnou (zaseknutý valcočlovek zle vplýva na kvalitu vína). Preto si valcočlovek musí vždy dobre rozmyslieť, cez ktorú pivnicu môže ísť.

ÚLOHA: Napíšte program, ktorý načíta reálne rozmery pivnice  $m, n$ , počet stĺpov  $p$  a reálny polomer valcočloveka  $r$ . Potom načíta súradnice stĺpov (pričom  $[0.00, 0.00]$  je juhozápadný roh pivnice,  $[m, n]$  severovýchodný) a napíše, či môže tento valcočlovek prejsť pivnicou od západnej steny po východnú.



### 1315. Frutus, Brutus a zúrivejší výrazy

27

„Napíšem!“ kričal Brutus jedného večera v útulnej krčmičke. „Nenapíšeš!“ oponoval Frutus. „A just napíšem!!“ nedal sa Brutus. „Ani keby si sa zderivoval!!!“ nedal sa presvedčiť Frutus.

Tentokrát sa hádali o tom, kto napíše zúrivejší výraz. Nakoniec svoj spor rozhodli stávkou. Každý napísal na servítku svoj výraz a začala sa hádka o tom, ktorý z nich je zúrivejší. Pri rozhodovaní musia vedieť, či náhodou tie výrazy nie sú rovnaké (v tom prípade by si asi navzájom vyrazili zuby).

ÚLOHA: Napíšte program, ktorý načíta dva výrazy skladajúce sa z jednopísmenových premenných, celočíselných konštánt, znamienok  $+$ ,  $-$ ,  $*$  a zátvoriek  $($  a  $)$  a vypíše, či sú ekvivalentné. Dva výrazy sú ekvivalentné, ak pre každú kombináciu hodnôt premenných nadobúdajú rovnakú hodnotu. Môžete predpokladať, že výrazy sú korektne zadané.

Príklad:

$$A * (A - B) + A * C$$

$$A * A - A * (B - C)$$

výstup: ANO

**1321. O kráľovstvách na Kiribati**

[25]

Kiribatské deti sa strašne radi hrávajú na kráľov a kráľovné – niet divu, veď kiribatské súostrovie má toľko ostrovov, že každý obyvateľ môže vládnuť aspoň jednému.

Raz sa im dostala do rúk mapa celého súostrovia. Každý ostrov bol zobrazený jednou bodkou. Deti sa začali hrať hru, pri ktorej si každý vybral kráľovstvo obdĺžnikového tvaru, ktorému by chcel vládnuť. Nezáležalo však na rozlohe kráľovstva, ale na počte ostrovov – ten kto ich má najviac, stane sa Najmocnejšo-Najhrôzostrašnejšo-Najsilnejším panovníkom.

ÚLOHA: Napište program, ktorý načíta súradnice ostrovov na mape (reálne čísla). Ďalej načíta kráľovstvá, pričom kráľovstvo je dané súradnicami ľavého horného a pravého dolného rohu (celé čísla v rozsahu 0 až 100) obdĺžnika a pre každé kráľovstvo vypíše počet ostrovov, ktoré sa v ňom nachádzajú.

**1322. Brutus a Frutus v knižnici**

► [25]

Brutus a Frutus po návšteve u zubára navštívili aj knižnicu. Vybrali si knihu „Zbierka úloh KSP“ a začali v nej listovať. Keď si konečne zo zbierky vybrali ten najzaujímavejší príklad a chceli sa pozrieť na jeho vzorové riešenie, zistili, že vzorové riešenie je z knižky vytrhnuté.

ÚLOHA: Na vstupe je číslo  $n$  a  $n$  celých čísel v rozsahu  $1 \dots n^2$ . Utriedte tieto čísla.

Triedenie so zložitou  $O(n \log n)$  nie je ani zďaleka najlepšie riešenie.

**1323. Banto a dostavníky**

[23]

Banto chce navštíviť starú mamu Prabantovku, ktorá býva až hen v Nalomenej Trieske. Preto musí vedieť, ako chodia v Nalomenej Trieske dostavníky. Prabantovka nevie, ktorý dostavník kedy a kam chodí, ale vždy, keď jej prehrkoce nejaký pod oknami, veľmi ju to rozruší, lebo dostavníky robia veľký hluk. Večer si Prabantovka všetky vzrušujúce veci (teda aj prechod dostavníka) zapíše do denníka. Chudák Banto by aspoň chcel vedieť, koľko rôznych liniek chodí cez Nalomenú Triesku. Preto požiadal Prabantovku, aby mu poslala výpis z denníka. Teraz sedí nad listom od starej mamy a nevie si rady.

ÚLOHA: Napište program, ktorý načíta obsah denníka Prabantovky a vypíše, koľko najmenej dostavníkových liniek chodí cez Nalomenú Triesku.

Výpis z denníka Prabantovky obsahuje začiatok sledovaného obdobia, počet dostavníkov, ktoré za sledované obdobie prešli popred Prabantovkine okná a ďalej pre každý dostavník deň, kedy prešiel popod okná. Prabantovka zásadne nepíše dátumy, ale počet dní od jej narodenia. Sledované obdobie trvá 60 dní. Zápisy sú chronologicky usporiadané.

Dostavníková linka sa vyznačuje tým, že dostavníky chodia v pravidelných intervaloch počas celého sledovaného obdobia. Každá linka prejde popod Prabantovkine okná počas sledovaného obdobia aspoň dvakrát. Môžete predpokladať, že cez Nalomenú Triesku nechodí viac ako 17 liniek a že za sledované obdobie neprešlo popod Prabantovkine okná viac ako 300 dostavníkov.

Príklad:

Vstup:

25500 17

25500 25503 25505 25513 25513 25515 25521 25526 25527 25529

25537 25539 25539 25545 25551 25552 25553

Výstup:

3 linky

(s intervalmi 13, 12 a 8 dní, prvý dostavník z linky ide v deň

25500, 25503 a 25505)

**1324. O vikingskom cestovateľovi**

[20]

V roku 1101 sa vikingský cestovateľ Uwe Písgarson doplavil až k brehom Južnej Ameriky. Len čo vystúpil zo svojej lode, zajali ho domorodci a chystali sa ho obetovať (t.j. obedovať). Po dlhom prosení ich Uwe presvedčil, aby mu dali ešte jednu šancu. Náčelník ho zaviedol do obrovskej miestnosti, v ktorej bol na stene veľký, ťažký kamenný kotúč s kolíkmi po obvodě pripomínajúci lodné kormidlo. Kotúč sa dal otáčať oboma smermi. Najspodnejší z kolíkov bol natretý červenou farbou. Na niektorých kolíkoch boli pripnuté zlaté disky, zvyšné disky ležali na zemi. „Pozri !“ povedal náčelník Uwemu a ukázal na mozaiku na protiľahlej stene. Znáznorňovala rovnaký kotúč ako v miestnosti, červený kolík opäť smeroval nadol, iba disky boli inak rozmiestnené. „Ak do východu Slnka budú na obidvoch stenách rovnaké kotúče, si voľný. Inak.“

Písgarson si pozornejšie obzrel kotúč. Zistil, že ho vládze otočiť o jedno miesto v oboch smeroch, alebo pripnúť (alebo odobrať) disk na spodný kolík (na iné kolíky nedočiachol).

Len čo osamel, dal sa do práce. Začal rýchlo otáčať kotúč, pripínať a odoberať disky. Kotúč bol však veľmi ťažký, a tak sa Uwe rýchlo unavil. Nad ránom od vysilenia odpadol a úlohu nesplnil.

ÚLOHA: Napíšte program, ktorý zachráni úbohého Uwe Písgarsona pred krutou smrťou. Váš program načíta počet kolíkov  $n$ , potom reprezentáciu kotúča t.j. postupnosť núl a jednotiek (kolíky bez disku a kolíky s diskom) počnúc červeným kolíkom v smere hodinových ručičiek a nakoniec reprezentáciu mozaiky (rovnakým spôsobom ako kotúč).

Výstup bude postupnosť písmen L, P, V (otočiť o jedno miesto v smere hod. ručičiek, proti smeru chodu hodinových ručičiek, výmena disku na spodnom kolíku), ktorá zabezpečí, že Uwe splní úlohu a nezomrie od únavy (t.j. dĺžka výstupnej postupnosti má byť minimálna).

Kotúč je veľmi ťažký a Uwe zomrie veľmi rýchlo. Diskov je k dispozícii dostatočný počet.

**1325. O politikoch**

[20]

Politici to vôbec nemajú jednoduché. Celé dni musia vysedávať na schôdzach parlamentu, zasadnutiach rôznych komisií a výborov, podchvíľou sa poriadka nejaká tlačová konferencia, alebo sa novinári snažia získať interview. Ale to všetko by sa ešte dalo vydržať. Najhoršie je, že taký politik musí veľmi často písať politické prejavy. Je to horšie ako domáce úlohy zo slohu.

Veľmi dôležité je, aby mal prejav požadovanú dĺžku, pričom dĺžka prejavu sa meria v počte riadkov. Všetky riadky (okrem posledného) obsahujú 60 písmen. Takisto by mal byť prejav zakaždým iný – ak by politik vždy použil ten istý prejav, hrozí, že si to za nejaký čas niekto všimne (je však veľmi pravdepodobné, že by to trvalo aspoň jedno volebné obdobie).

ÚLOHA: Napíšte program, ktorý na vstupe dostane predpísaný počet riadkov politického prejavu a vypíše čo najpôsobivejší politický prejav predpísanej dĺžky. Vo svojom prejave môžete použiť nasledovné symboly: <DP> – dramatická pauza, :-> – podmanivý úsmev, 8-( – zdravé rozhorčenie, <NSVPS> – nadšené súhlasné výkriky z poslaneckej snemovne.

**1331. O snaživom Tomášovi**

[27]

Ako iste všetci viete, Tomáš je veľmi snaživý študent, a preto keď boli zverejnené rozvrhy, rozhodol sa, že si zapíše najviac prednášok ako len môže. Dlhو dumał nad rozvrhom a nič. Veľmi sa preto trápi. Nikdy si nie je istý, či rozvrh, ktorý si vymyslel, má najviac prednášok ako len môže mať. Za týždeň už schudol 5 kíł. Pomôžte mu, než bude mať zápornú hmotnosť a niekam nám uletí.

ÚLOHA: Napíšte program, ktorý načíta začiatky a konce prednášok a vypíše Tomášovi najväčší počet prednášok, ktoré si môže zapísať (samozrejme nemôže prísť na dve pred-

nášky naraz, ani prísť neskoro, či odísť priskoro z prednášky) spolu s príkladom takéhoto rozvrhu.

Vstup: Každá prednáška v tvare Pon 07:20–09:45 v novom riadku. Dni v týždni sú Pon Uto Str Stv Pia, v sobotu a nedeľu prednášky nie sú.

Výstup: Počet prednášok v rozvrhu. Zoznam zapísaných prednášok, každá v samostatnom riadku

Nezabudnite na dôkaz správnosti algoritmu.

Príklad:

Vstup:	Výstup:
Pon 07:20–09:45	2
Uto 09:30–10:00	Pon 07:20–09:45
Pon 09:00–10:35	Uto 09:30–10:00

### 1332. O Števovej návšteve v Indii

[21]

Števo sa potuloval svetom a keď už bol v Ázii, spomenul si, že je hladný. Zastavil sa preto v dedine Maharádzdžiád. Prišiel do obchodu a chcel si kúpiť veľa jedla na ďalšiu cestu. Keď už stál v rade všimol si, že na pokladni je napísané  $1 + 1 = 10$ . Spočiatku nevedel, čo to znamená. No po chvíli mu bolo všetko jasné. „V tomto obchode počítajú v inej číselnej sústave!“, skríkol Števo. Tak bežal do ďalšieho. Tam počítali zasa v inej sústave. Na každom obchode bolo napísané výraz tvaru  $A + B = C$ .

Števo nalistoval zodpovedajúcu stranu v svojej príručke mladých svištov a zistil, že takýto výraz slúži na určenie číselnej sústavy, v ktorej sa v danom obchode počíta. Števo nevie po indicky, preto v obchode nakupuje tak, že ukáže, čo chce a na papier napíše, koľko toho chce. Na to však potrebuje vedieť, v akej sústave sa v danom obchode počíta. Keď to nezistí, je odsúdený na smrť vyhladovaním.

ÚLOHA: Napíšte program, ktorý zistí, v ktorých číselných sústavách platí  $A + B = C$ . Predpokladajte, že cifry sú len  $0, 1, 2, \dots, 9, a, b, \dots, z$ .

Príklad vstupu:  $1+1=10$

Výstup: 2

### 1333. O Kiribatských náleziskách

[27]

Na Kiribatských ostrovoch nedávno objavili pod zemou veľké zásoby rumu. Inžinieri navrhli miesta, kde postaviť vrtné veže, problémom však je preprava rumu z týchto veží. Spoločnosť, ktorá rumové polia vlastní sa rozhodla postaviť jednu veľkú rúru idúcu cez celé pole z východu na západ a ku každej veži postaviť prípojku k tejto veľkej rúre. Kam ale postaviť veľkú rúru, aby sa na prípojky minulo čo najmenej materiálu? Nad tým treba veru triežvo pouvažovať!

ÚLOHA: Napíšte program, ktorý načíta  $n$  - počet ťažných veží, súradnice jednotlivých ťažných veží a vypíše  $y$ -ovú súradnicu kam postaviť veľkú rúru, tak, aby súčet dĺžok prípojok bol minimálny.

Príklad:

Vstup:	Výstup:
5	17
10 1	
15 20	
20 17	
10 7	
10 10	

### 1334. O telefónnom víruse

[21]

Vírus Prepheekunecz, ktorý infikuje moderné telefóny, vždy v piatok poprehadzuje význam tlačítek telefónu, t.j. napríklad po stlačení 1 sa vytočí číslo 5 a podobne, pričom je však telefón schopný vytočiť ktorékoľvek číslo.

Každý majiteľ telefónu má našťastie svoj súkromný telefónny zoznam, na základe ktorého môže zistiť, ako sú tlačítka poprehadzované a to tak, že vytáča rôzne čísla a podľa toho, kto sa mu ozve zistí, aké číslo vlastne vytočil.

ÚLOHA: Napište program, ktorý na čo najmenej vytočení zistí, ako sú poprehadzované tlačítka telefónu. Program má k dispozícii funkciu `vytoc(st : string) : integer`, ktorá dostane na vstupe vytáčané číslo (v poprehadzovaných tlačítkach) a vráti buď číslo volanej osoby, alebo 0, ak osoba s takým (správnym) číslom nie je v súkromnom telefónnom zozname. Telefónny zoznam máte k dispozícii ako pole `TZ[1..n]`, kde `TZ[i]` je (správne) telefónne číslo osoby  $i$ . Ak nie je možné zistiť poprehadzovanie tlačítok, vypíšte o tom správu.

Příklad:

Čísla sú prehádzané takto: (2,3,5,0,9,1,8,4,6,7)

Zoznam obsahuje čísla: [123,456,7890]

Možný postup vytáčania:

`vytoc('350')` vráti 1

`vytoc('918')` vráti 2

`vytoc('4672')` vráti 3

`vytoc('123')` vráti 0 (lebo neexistuje nikto s číslom 501)

### 1335. O SoDr a meteorológoch

??

V softvérovom družstve majú najnovšie klientov – meteorológov. Taký meteorológ keď príde k počítaču, chcel by vedieť, v akom rozmedzí uhlov v posledných  $n$  minútach fúkal vietor. Výsledok merania smeru vetra je k dispozícii každú minútu (zadáva sa v stupňoch).

Rozmedzie uhlov možno určiť len ako súvislý oblúk. Oblúk je určený dvoma číslami  $a$  a  $b$ , ktoré určujú hranice oblúku. Orientácia oblúku je v protismere hodinových ručičiek od  $a$  k  $b$ . Napríklad oblúk (0,90) určuje štvrtkruh a oblúk (90,0) trištvrtkruh.

ÚLOHA: Na vstupe sú čísla  $n$ ,  $k$  a  $n+k$  celých čísel z intervalu  $[0, 360)$ , ktoré sú výsledkami meraní postupne v časoch  $1, 2, \dots, n+k$ . Napište program, ktorý vypíše rozmedzia uhlov pre časy  $n, n+1, \dots, n+k$  vždy za posledných  $n$  minút. Môžete predpokladať, že  $k$  je aspoň 3.

Například pre  $n = 3$ ,  $k = 3$  a merania: 0,30,60,90,120,150 je výsledkom (0,60) (30,90) (60,120) (90,150).

### 1341. Brutus a Frutus v knižnici II

25

Brutovi a Frutovi sa v knižnici zapáčilo a tak sa tam opäť vybrali (však koho by bavilo stále navštevovať zubára). Čírou náhodou si znovu vybrali knihu „Zbierka úloh KSP“ a začali v nej listovať. Keď si konečne zo zbierky vybrali druhý najzaujímavejší príklad a chceli sa pozrieť na jeho vzorové riešenie, zistili, že nejaký zurvalec pre istotu vytrhol všetky vzorové riešenia.

ÚLOHA: V súbore je daných veľa čísel typu `longint`. Napište program, ktorý vytvorí súbor s tými istými číslami uloženými v poradí od najmenšieho po najväčšie. Čísel je tak veľa, že sa určite nezmestia do pamäti počítača. Predpokladajte, že na disku máte miesta dosť.

### 1342. O univerzitnom knihovníkovi

75

„Oook!“, povedal knihovník, keď doniesli ďalších  $n$  kníh. V univerzitnej knižnici bolo doteraz  $n$  kníh od rôznych autorov z najrôznejších odvetví mágie. Ku každej knihe teraz priviezli druhý diel a uložili ich na policu za pôvodnými knihami. Privezené knihy sú pritom usporiadané rovnako ako pôvodné. Knihovník by mal rád banán (lepšie povedané, chcel by banán).

ÚLOHA: V poli  $A$  veľkosti  $2n$  (polica) sú uložené postupne čísla  $a_1, \dots, a_n$  (prvé diely kníh),  $b_1, \dots, b_n$  (druhé diely kníh). Napište procedúru, ktorá preusporiada pole  $A$  tak, aby



v ňom boli prvky v poradí  $a_1, b_1, a_2, b_2, \dots, a_n, b_n$  (teda prvý a druhý diel vždy pri sebe). Nepoužívajte žiadne pomocné pole (knihovník nemá žiadnu ďalšiu „pomocnú“ policu).

### 1343. O čarodejnom písacom stroji

27

Čarodejnica Magica má vo svojej kancelárii prečudesný písací stroj, ktorým si zapisuje svoje kúzla na netopierie kože. Hlavná časť stroja som Ja – ľudská ruka s náramkom, na ktorom sú napísané písmenká (rovnaké písmenká môžu byť na náramku aj viac krát). Môžem sa točiť okolo osi určenej prostredníkom tak, že nad kožou sa objavujú iné písmenká, ktoré, ak čarodejnica stlačí červené tlačítko, vytlačím. Som z dlhého písania veľmi unavená a preto som sa rozhodla, že napíšem program, ktorý moju prácu uľahčí a pre dané poradie písmeniiek na náramku a požadovaný text vypíše postupnosť krokov ako sa mám točiť, aby bol počet otočení minimálny. Keďže mi chýba hlava, mohli by ste mi pomôcť.

ÚLOHA: Napíšte program, ktorého vstupom je najprv postupnosť písmeniiek na náramku (pričom prvé písmenko zodpovedá písmenku, ktoré je práve nad kožou) a potom v ďalšom riadku veta, ktorú treba napísať. Výstupom je postupnosť príkazov DOLAVA, DOPRAVA, STLAC a MEDZERA realizujúca danú vetu, alebo FUJ ak sa veta nedá s daným náramkom napísať.

Príklad:

Vstup:

```
abacd
aba dac
```

Výstup:

```
STLAC DOPRAVA STLAC DOLAVA STLAC MEDZERA DOLAVA STLAC DOLAVA
DOLAVA STLAC DOPRAVA STLAC
```

### 1344. O kachličkovaní

40

Veveričkám Anke a Hanke v Hornom-Dolnom je v ich domčeku zima a preto si u slona objednali slonovinové kachličky obdĺžnikového tvaru rozmeru  $1v_n \times 2v_n$  ( $1v_n$  = jedna veveričia nožička). Hrochovi na úrad priniesli obrázok svojej izbičky nakreslený na štvorcovom papieri so štvorčekmi s hranou  $1v_n$  a chcú vedieť, či sa dá vykachličkovať. Hroch sa práve zobudil a preto sa mu nechce veľmi rozmýšľať a vôbec.

ÚLOHA: Napíšte program, ktorý načíta tvar miestnosti v tvare mapy skladajúcej sa z medzier a znakov # (takýto krížik znázorňuje štvorček podlahy veľkosti  $1v_n \times 1v_n$ ) a vypíše, či sa dá miestnosť vykachličkovať, alebo nie.

Príklad:

Vstup:

```
# #
#####
# #
```

Výstup:

NIE

### 1345. O 276. zákazníkovi

25

Do softférového družstva SoDr dnes prišiel jubilejný 276. zákazník, ako obvykle, so zaujímavým problémom. Zákazník bol študentom nemenovanej vysokej školy a ako to tak už u vysokoškolákov býva, nevedel pracovať so zlomkami. Preto potreboval knižnicu, ktorá by mu túto prácu umožnila. Zákazník zaplatil, vzal si so sebou bloček z registračnej pokladne a tu máte úlohu.

ÚLOHA: Navrhňte a napíšte knižnicu, ktorá bude umožňovať prácu so zlomkami. Knižnica by mala zvládnuť základné aritmetické operácie, ako aj procedúry pre vstup a výstup.

Pokiaľ sú oba operandy aj výsledok vo vami určenom rozmedzí, nemal by váš program vyhlásiť pretečenie.

## 1411. O tybloni

► 40

V Nalomenej Trieske rastie čudesa tybloň. Ak ste takýto strom ešte nevideli, nevadí, ani my. Trabantomka povedala, že jeho plodmi sú pochopiteľne tyblká. Tyblko vyzerá skoro ako onbko, ibaže je dvojrozmerné, teda má tvar ľubovoľného mnohouholníka, ktorý má v každom vrchole malé modré nič (keď je tyblko zrelé). Keďže každé tyblko je iné, pri jeho rozdeľovaní nastávajú problémy. Treba ho totiž rozdeliť tak, aby pri rezaní vznikli len časti trojuholníkového tvaru, pričom v každom vrchole takéhoto trojuholníka je malé modré nič (v opačnom prípade môže pri konzumácii vzniknúť nepríjemná alergická reakcia).

ÚLOHA: Napíšte program, ktorý pre daný mnohouholník vytvorí jeho ľubovoľnú trianguláciu. Triangulácia je také rozdelenie mnohouholníka na neprekrývajúce sa trojuholníky, že vrcholy každého trojuholníka sú vrcholmi mnohouholníka. Mnohouholník je zadaný počtom vrcholov  $n$  a vymenovaním súradníc jeho vrcholov po obvodu proti smeru hodinových ručičiek. Výstup programu bude pozostávať zo zoznamu vytvorených trojuholníkov.

Napríklad pre  $n = 5$  a  $[0, 0], [2, 2], [1, 2], [2, 3], [0, 3]$  je výsledok  $([0, 0], [2, 2], [1, 2])$   $([0, 0], [1, 2], [0, 3])$   $([1, 2], [2, 3], [0, 3])$ .

## 1412. O dlhočizných dážďovkách

30

Ježkovia to mali vždy jednoduché. Keď sa chceli zmestiť do malého priestoru, proste sa schúlili do kľbka. Dážďovky to majú omnoho ťažšie. Vedia sa totiž zohýbať len vo svojich „kľboch“, ktoré majú medzi článkami. Našťastie dážďovky sú dosť šikovné – vedia sa dokonca zohnúť v kľbe tak, že ich články sa spätne prekrývajú...

ÚLOHA: Napíšte program, ktorý pre zadané dĺžky článkov dážďovky zistí, na akú minimálnu dĺžku sa môže dážďovka poskladať. Dážďovka je priamka, ktorá pri poskladaní môže byť v kľboch ohnutá o  $0^\circ$ , alebo o  $180^\circ$ . Články dážďovky po sebe nasledujú v takom poradí, v akom sú na vstupe zadané.

Napríklad pre 5 článkov s dĺžkami 1, 3, 2, 3, 2 je dĺžka poskladanej dážďovky 4.

## 1413. O naladenej strune

25

Usilovný Tomáš si nechal za ťažké peniaze naladiť klavír. Keďže ladič býva až hen za lesom, musel si Tomáš klavír dotlačiť späť do brlôžka sám. Aby sa klavír nerozladil, nesmie ním Tomáš otáčať – odstredivá sila pôsobiaca pri takomto pohybe na struny by ich natiahla a klavír by bol znovu rozladený.

Každý z vás už iste aspoň raz v živote tlačil klavír lesom a preto si viete predstaviť, čo to znamená. Lepšie je si najskôr zistiť, či sa vôbec klavír lesom pretlačiť dá. Les je zo severu aj z juhu ohraničený dvoma riekami, ktoré tečú tesne pri najsevernejšom a pri najjužnejšom strome lesa. Chudák Tomáš teda potrebuje zistiť, či môže klavír presunúť z východu na západ (bez otáčania).

ÚLOHA: Napíšte program, ktorý pre zadaný počet stromov, ich súradnice a rozmery klavíra (obdĺžnika) vypíše, či je možné daný klavír lesom preniesť. Klavír je na začiatku na hony vzdialený od lesa a hrany jeho kolmého priemetu na zem sú rovnobežné so súradnicovými osami. Stromy majú zanedbateľný polomer. Klavír (ani Tomáš) nevie plávať.

Ak sa Vám zdá úloha príliš ľahká, vezmite do úvahy koncertné krídlo – také koncertné krídlo má inú polohu strún a preto ho smelo možno v lese aj otáčať.

Napríklad pre 6 stromov so súradnicami  $[0, 1], [3.5, 3.5], [0.5, 3], [2, 0], [0, 5], [2, 1.5]$  a klavír s veľkosťou strán  $x = 3, y = 2$  sa klavír lesom preniesť dá.

## 1414. The Streets of San Benátky

► 25

Benátčan s dlhým a nezrozumiteľným menom má problém. Mestský úrad mu nariaďil rozhodnúť, ktoré z mostov sú natoľko významné, že ich absencia by dlhodobo ochromila dopravnú situáciu v meste. Most je významný, ak v prípade, že by spadol by v meste existovali aspoň dve dôležité miesta, medzi ktorými by neexistovalo žiadne spojenie. Treba si

uvedomiť, že v Benátkach nemajú cesty, ale iba kanály a mosty nad nimi (a po kanáloch prirodzene nemožno chodiť, lebo by mal človek veľmi rýchlo mokré topánky).

ÚLOHA: Pomôžte Benátčanovi s dlhým a nezrozumiteľným menom a napíšte program, ktorý načíta počet dôležitých miest  $n$ , počet mostov  $m$  a ďalej  $m$  dvojíc čísel dôležitých miest, pričom každá dvojica znamená, že tieto dôležité miesta sú priamo spojené mostom a potom vypíše všetky významné mosty v Benátkach.

Napríklad pre 7 dôležitých miest, 8 mostov: (1, 2), (2, 3), (4, 7), (1, 4), (3, 6), (2, 4), (5, 6), (3, 5) sú významné mosty (2, 3), (4, 7).

#### 1415. O Števovej sieti

[35]

Števo sa so svojimi kamarátmi vybral na výlet do lesa. Chodili po lese, chodili, až boli tak unavení, že ďalej chodiť nevládali. I rozhodli sa, že si spravia obedovú prestávku.

Čo môžu robiť študenti Matematicko-fyzikálnej fakulty v lese, keď si spravia obedovú prestávku? To je jednoduché. Každý vyliezol na nejaký strom a vytiahol z batohu chlieb, slaninu, cibuľu a notebook. I omrzelo ich len tak každý sám obedovať, do LCD displeja hľadieť, nuž pospájali počítače náhodným spôsobom do siete. Teraz sedia a nevedia, čo si so sieťou počať, lebo nemajú tušenia, ako poslať jeden druhému po sieti e-mail.

ÚLOHA: Každý počítač je spojený s niekoľkými inými počítačmi, s každým osobitnou linkou. Každý počítač má svoje linky očíslované od 1 po  $n$ , kde  $n$  je počet jeho liniek. Každý počítač v sieti má jednoznačne priradené meno (reťazec max. 10 písmen, ktorý neobsahuje medzery – napríklad FERDO). Ak sú počítače A a B spojené linkou, môže napríklad počítač A poslať správu počítaču B, správa sa zaraď do mailboxu počítača B.

Predstavme si takýto spôsob posielania e-mailov. Počítač pošle e-mail niektorou svojou linkou. Počítač na druhom konci linky e-mail prevezme, zistí pre koho je, a ak nie je pre neho, pošle ho niektorou svojou linkou ďalej (prerútuje ho). Aby e-maily zbytočne dlho neblúdili po sieti (trebárs aj na veku-vekov), každý počítač musí vedieť pre každý iný počítač v sieti, ktorou cestou e-mail poslať, aby sa na najmenší počet prerútovaní dostal do cieľa (tieto údaje sú uložené v takzvanej *rútovacej tabuľke*).

Napíšte program, ktorý (pokiaľ sa napchávajú slaninou) spustia všetci Števovi kamaráti (Števo je egocentrický – kamaráti sa aj sám so sebou) na svojich laptopoch, pričom po jeho skončení bude mať každý z počítačov k dispozícii rútovaciu tabuľku (program beží na každom počítači nezávisle).

Pre komunikáciu medzi počítačmi máte k dispozícii tieto procedúry (pre jazyk C, prípadne iný jazyk zadefinujte procedúry s podobnou štruktúrou; *smes* je to isté čo **string**, iba s neobmedzenou dĺžkou):

*Send(sprava : smes; linka : integer)* – počítač pošle správu *sprava* po linke *linka*.

*Receive(var sprava : smes; var linka : integer)* – vyberie z mailboxu jednu správu a vráti v premennej *sprava* jej obsah a v premennej *linka* údaj o tom, z ktorej linky správa prišla. Ak žiadna správa v mailboxe nie je, *Receive* čaká až kým tam nejaká nebude.

*NumberOfLinks(var n : integer)* – do premennej *n* uloží počet liniek, ktoré vedú z počítača.

*MyName(var s : string[10])* – do premennej *s* uloží meno počítača.

Snažte sa, aby počítače skončili skôr, ako študenti dojedia svoju slaninu (Števo slaninu neje, namiesto nej je kaleráb). Linky sú pomalé a preto sa snažte, aby ste toho neprenášali zbytočne veľa.

#### 1421. O koláči II

[21]

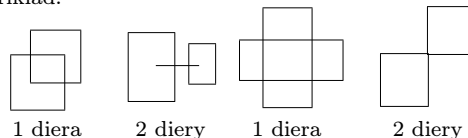
Na obdĺžnikovom plechu upiekli (mimochodom nie veľmi dobrý) koláč veľmi veľkých rozmerov. Koláč sa navyše na krajoch pripálil a tie držali na plechu ako prilepené kanagom.

Hostia sa už zišli, nikto nevie čo im ponúknuť, tak sa rozhodli, že predsa len niečo z toho plechu vydolujú. Zobrali nôž a začali do koláča rezať zvislé a vodorovné úsečky, ktoré sa nedotýkali pripálených okrajov.

Potom plech preklopili, vysypali kúsiky rýchlo na tanier a odniesli ich hostom. Zaujímavou otázkou však zostáva, koľko dier zostalo v pôvodnom koláči.

ÚLOHA: Daných je  $n$  začiatočných a koncových bodov jednotlivých úsečiek vyrezaných do koláča. Napíšte program, ktorý zistí, koľko dier zostalo v koláči po vyklopení plechu.

Príklad:



#### 1422. O univerzitnom knihovníkovi II

[35]

„Oook!“ začudoval sa knihovník, keď zistil, že dnes sú na polici zase iné knihy. Včera polica obsahovala niekoľko prvých a druhých dielov Lindenmayerovho zborníka magických formúl. Aj dnes tam boli prvé a druhé diely toho istého zborníka, ale určite sa líšil ich počet a rozmiestnenie. Keď knihovník niekoľko dní knihy pozoroval, zistil, že ich zmeny nie sú náhodné. Každú noc z každého 1. dielu vznikne skupina kníh  $u$  a z každého druhého dielu vznikne skupina kníh  $v$ , pričom  $u$  a  $v$  sú každú noc tie isté. Tak sa rozhodol dať Mrakoplašovi úlohu. Napísal mu na papier nejaké poradie prvých a druhých dielov a prikázal mu zistiť, či niektorý deň bude na polici taká istá skupina prvých a druhých dielov ako je na papieri (naľavo aj napravo od tejto skupiny môžu byť na polici ešte ďalšie knihy).

ÚLOHA: Pomôžte Mrakoplašovi a napíšte program, ktorý zistí odpoveď na knihovníkovu úlohu. Program načíta postupnosti  $u$ ,  $v$ ,  $w$ ,  $x$  pozostávajúce z čísel 1 a 2, kde  $u$  je skupina kníh, na ktoré sa každú noc zmení kniha 1,  $v$  je skupina kníh, na ktoré sa zmení kniha 2,  $w$  je dnešný stav police a  $x$  je knihovníkov zoznam ( $u$  a  $v$  môžu byť aj prázdne postupnosti).

Napríklad nech  $u = 12$ ,  $v = 212$  a  $w = 21$ . Každú noc sa každá 1 zmení na 12 a každá 2 sa zmení na 212. Na druhý deň budú na polici knihy 21212 a na tretí deň 21212212212. V prípade, že  $x = 1221$ , odpoveď na knihovníkovu otázku bude **ano**, lebo takáto skupina sa nachádza v postupnosti kníh tretieho dňa. V prípade, že  $x = 11$ , odpoveď bude **nie**, lebo takáto skupina nikdy nemôže vzniknúť.

#### 1423. O vrabcovi Čin-činovi II

[20]

Vrabc Čin-čin zase priletel neskoro. Bolo to už piate pomaturitné stretnutie a doteraz sa mu ešte nikdy nepodarilo priletieť včas. Aby nenarobil zmätok a nepokazil veselú zábavu, chcel si už vopred vyhladiť miesto, na ktoré sa posadí.

Vždy sedávali na prvom drôte hneď za dedinou, na úseku od stĺpu EV 274 po stĺp EV 277. Čin-čin nerád sedával na kraji, pretože tam sa mohol rozprávať len s jedným zo svojich susediacich (nakolko vrabc na kraji ich viac nemá). Ale na druhej strane, Čin-čin má rád okolo seba miesto. Preto by si teraz najradšej sadol medzi tých dvoch susedných vrabcov, ktorí sú od seba najviac vzdialení. Z tolkej diaľky však nedovídi na starých kamarátov. Našťastie blízko sedí straka Rapotajka, ktorá si iste túto nevšednú udalosť nenechala ujsť a má o všetkom prehľad. A naozaj. Rapotajka Čin-činovi vyrapotala, kedy ktorý vrabc prišiel a kam sa usadil.

ÚLOHA: Napíšte Čin-činovi program, ktorý načíta dĺžky drôtu  $A$ ,  $B$  od stĺpov EV 274 a EV 277 do dediny, ďalej načíta  $n$  doteraz usadených vrabcov a ich vzdialenosť po drôte od dediny. Program vypíše dvoch susedných vrabcov, ktorí sú spomedzi všetkých susedných vrabcov najviac od seba vzdialení. Riešenia lineárne od počtu vrabcov  $n$  sú veľmi vítané.

Například pre  $A = 5$ ,  $B = 16$ ,  $n = 5$  a polohy vrabcov: 9, 13, 7, 15, 10 sú najvzdialenejší vrabci s polohami 10 a 13.

#### 1424. O kiribatských náleziskách II

► 35

Na Kiribatských ostrovoch nedávno objavili pod zemou veľké zásoby rumu II. Po predchádzajúcich skúsenostiach inžinieri postavili jednu veľkú vrtnú vežu, problémom je však preprava rumu II. z tejto veže.

Po niekoľkých dúškoch inžinieri navrhli vybudovať veľkú rumárenskú sieť skladajúcu sa z rumovodov a z regulačných staníc. Na každej rúre je umiestnená tzv. čuchačka, cez ktorú môže poverený pracovník údržbovej skupiny očuchávať, či rum náhodou nevyteká mimo rumovod. Každou rúrou môže pretiecť hektoliter rumu za minútu. Regulačné stanice sú miesta, v ktorých sa stretáva jedna alebo viacero rúr, ktoré sú v tomto mieste prepojené. Platí zásada, že prítok do regulačnej stanice a výtok z nej sú v rovnováhe. Vplyv zamestnancov regulačnej stanice na porušenie tejto rovnováhy po vzore priateľov fyzikov zanedbáme. Celá sieť je v regulačnej stanici s priamo napojená na vrtnú vežu a rum je odčerpávaný zo siete v mestskej regulačnej stanici  $t$ .

Ráno inžinieri nad svojim návrhom triezvo pouvažovali a zistili, že stavba by to bola síce veľkolepá, ale rumu by sa cez ňu veľa do mesta neprepravilo.

ÚLOHA: Daný je počet regulačných staníc  $n$ , počet rumovodov  $m$  a ďalej  $m$  dvojíc čísel reprezentujúcich jednotlivé rumovody (od ktorej regulačnej stanice ku ktorej vedú), pričom stanica  $s$  má číslo 1 a stanica  $t$  má číslo  $n$ . Napíšte program, ktorý vypočíta, koľko rumu za minútu je takto navrhnutý rumovod schopný prepraviť.

Predpokladajte, že vyťaží možno toľko rumu, koľko z  $s$  odtiekne a že v  $t$  odčerpávajú bezo zbytku všetok rum, ktorý tam pritečie.

Například pre  $n = 4$ ,  $m = 5$  a rumovody: (1, 2), (2, 3), (1, 3), (3, 4), (1, 4) cez rumárenskú sieť pretečú najviac 2 hektolitre rumu za minútu.

#### 1425. O Števovej sieti II

30

Oprava zadania O Števovej sieti: Naše historicko-výskumné oddelenie zistilo, že Števo nemohol byť kaleráb, keďže kaleráby vôbec nemá rád. Správna zelenina na tomto mieste bola mrkva.

Števo dojedol svoju mrkvu, poobhliadal sa po okolí, čože to robia jeho kamaráti a zhrozil sa. V počítačovej sieti panoval úplný chaos! Števo začal liezť po stromoch, rozpájať, prepájať a zapájať nové káble a po niekoľkých minútach pospájal počítače pekne do kruhu. A spokojne na svoje dielo pozeral.

I Števo povedal, ak chcete riešiť akúkoľvek úlohu, musíte zvoliť šéfa! A študenti Matematicko-fyzikálnej fakulty sa zamysleli nad Števoými slovami a videli, že to čo vraví, je dobré.

ÚLOHA: Každý počítač je spojený s dvoma susednými počítačmi v kruhu, s každým osobitnou linkou. Linky majú čísla 1 (ľavý sused) a 2 (pravý sused). Každý počítač v sieti má jednoznačne priradené meno (reťazec max. 10 písmen, ktorý neobsahuje medzery – napríklad FERDO). Ak sú počítače  $A$  a  $B$  spojené linkou, môže napríklad počítač  $A$  poslať správu počítaču  $B$ , správa sa zaradí do mailboxu počítača  $B$ .

Napište program, ktorý (pokiaľ rozmýšľajú o Števoých slovách) spustia všetci Števovi kamaráti na svojich laptopoch, pričom po jeho skončení bude každý počítač poznať meno šéfa (keďže šéf môže byť len jeden, každý počítač musí zistiť to isté meno, je však úplne jedno, ktorý z počítačov bude šéfom). Program beží na každom počítači nezávisle.

Pre komunikáciu medzi počítačmi máte k dispozícii rovnaké procedúry ako v úlohe 1415.

Snažte sa, aby počítače skončili skoro, aby študenti príliš dlho nerozmýšľali nad Števoými slovami, mohlo by to mať nedorozumenia následky. Linky sú pomalé a preto sa snažte, aby ste toho neprenášali zbytočne veľa.

## 1431. O magickom kruhu

[35]

Stará sága vraví, že niekde na okraji Zemeplochy sa za dávnych časov nachádzal kontinent menom Ulantída. Avšak pred viac než mnohými rokmi ho pohltilo more. Mnoho cestovateľov zahynulo pri hľadaní tohto kontinentu. I čarodej Taratlach sa vydal Ulantídu hľadať. Keďže čoskoro cesta pokračovala po dne oceánu, musel Taratlach (i keď s nevôľou) prikočiť ku kúzleniu. Požiadať vodu aby sa rozostúpila, Nemôže byť žiadny problém – veď to už kedysi dávno zvládol akýsi Mojžiš. Text v jeho príručke hovoril, že treba na pláži vyznačiť kruh, postaviť sa do stredu a odriechnúť príslušnú magickú formulu. Potom neviditeľná ruka vyznačí na pláži 666 priamok a čo najrýchlejšie treba vykriknúť, koľkokrát sa priamky pretli vo vnútri kruhu. Taratlach v matematike nikdy nevyňikal a tak si spočítal, že by mu už len spočítanie tých priamok trvalo dlhšie, ako stanovený časový limit. Teraz len smutne sedí na pláži a hľadá do vln.

ÚLOHA: Napíšte program, ktorý načíta čísla  $n$  (počet priamok),  $r$  (polomer kruhu) a  $n$  priamok, pričom každá je daná súradnicami svojich dvoch bodov a spočíta, koľko dvojíc priamok sa pretne vnútri kruhu s polomerom  $r$  a stredom v počiatku súradnicovej sústavy.

Napríklad pre  $r = 3$ ,  $n = 4$  a priamky  $p$ :  $[-2, 0]$ ,  $[0, 2]$ ;  $q$ :  $[-5, 1]$ ,  $[5, 1]$ ;  $r$ :  $[-1, 0]$ ,  $[-1, -2]$  a  $s$ :  $[4, 7]$ ,  $[4, 6]$  je výsledok 3, lebo sa pretnú tri dvojice priamok ( $p$  a  $q$ ,  $p$  a  $r$ ,  $q$  a  $r$ ).

## 1432. O koláči III

V Nalomenej Trieske je nepísaným pravidlom, že nevesta, keď sa vydáva, musí upiecť svadobný koláč. Veľkosť koláča je úmerná spoločenskému postaveniu nevesty.

Prabantovka spomína: Jój, to za našich mladých čias, to boli iné svadby ako teraz. Ale najkrajšiu svadbu mala kováčova Anička, keď si brala richtárovie Ondra. Koľko prišlo vtedy hostí! A každému sa ušlo z Aničkinho koláča. Anička upiekla totiž koláč v tvare veľkého konvexného mnohouholníka – tyblka, a dokonca do každého z vrcholov malé modré nič pridala. Každý kúsok koláča mal trojuholníkový tvar, v každom vrchole kúsok malého modrého nič. Všetko prebiehalo hladko, až kým Ďuro, Aničkin ohrdnutý pytač, nezačal rozhlasovať, že Anička taký veľký koláč upiecť nemohla. Veď plech, na ktorom by sa taký koláč dal upiecť, hádam ani nejestvuje... Urazený kováč doniesol ohromný mnohouholníkový plech, pozbieral všetky kúsky koláča, zavolať učiteľa, notára a farára a spolu sa pustili ukladať kúsky koláča na plech. Po hodnej chvíli, keď zistili, že to nie je až také jednoduché, pozorne zmerali každý kúsok koláča, zakreslili tvar plechu, farár s notárom sa pod to slávnostne podpísali. Potom konečne rozdali koláče a svadobná hostina sa mohla začať...

ÚLOHA: Kováč vypísal odmenu tomu, kto pomôže celej Nalomenej Trieske ukázať, že Aničkin svadobný koláč sa skutočne celý piekol na tom obrovskom plechu. Napíšte program, ktorý načíta tvar plechu (vrcholy konvexného mnohouholníka vymenované proti smeru hodinových ručičiek), počet kúskov koláča a dĺžky strán jednotlivých kúskov (tiež proti smeru hodinových ručičiek) a vypíše, či sa kúsky dajú poukladať na plech tak, aby nič nezvyšilo, aby bol celý plech pokrytý a aby malé modré nič boli vo vrcholech plechu.

Napríklad na plech so súradnicami vrcholov  $(0.0; 0.0)$ ,  $(3.0; 0.0)$ ,  $(4.5; 2.0)$ ,  $(3.0; 4.0)$  a  $(0.0; 4.0)$  sa koláče  $(3.0; 4.0; 5.0)$ ,  $(3.0; 4.0; 5.0)$  a  $(2.5; 2.5; 4.0)$  dajú poukladať.

## 1433. O vrabcovi Čin-činovi I

[35]

Keď sa Čin-čin vrátil zo stretnutia domov, chvíľu veru rozmýšľal, či trafil správne. Veď áno, trochu aj pili, ale žeby až tak? Rozhodne niečo nebolo v poriadku. Ale čo? Čin-čin sa posadil na priedomie domu, o ktorom si ešte stále myslel, že je jeho vlastný a pustil sa do tuhého premýšľania. A po čase na to naozaj prišiel. Tie zbalené kufre boli tým, čo sa mu nepozdávalo. Aby sa priznal, boli mu nanajvýš podozrivé, len si žiaľ práve teraz nevedel spomenúť, na čo je taký kufor dobrý. Našťastie o chvíľu ku nemu pricupital malý vrabček a skutočne netrvalo dlho, kým v ňom Čin-čin spoznal svojho malého Pim-pima, synáčka, na ktorého bol právom hrdý. A vtedy sa spamätal. Veď dnes je presne ten deň, keď sa mali sťahovať! Ale v tých dvoch kufroch asi nebude zbalené celé dvojposchodové hniezdo

aj s nábytkom, uvažoval Čin-čin ďalej. A potom je tu pelikán a prázdne škatule... Čin-čin začínal tušiť, čo ho čaká. Zvyšné veci bude asi treba zbaliť do tých škatúl. Ale škatúľ nemôžem použiť veľa, veď sú dosť drahé – Čin-čin začínal prejavovať známky trpezlivého uvažovania.

ÚLOHA: Dané sú čísla  $s$  – objem jednej škatule (všetky škatule majú rovnaký objem),  $n$  – počet zvyšných vecí, ktoré treba pobaľiť, a potom ešte  $n$  čísel – objemov jednotlivých vecí. Všetky tieto údaje sú celé čísla. Čin-čin vie, že keby chcel zistiť, ako sa dajú jeho veci pobaľiť do najmenšieho možného počtu krabíc, odpovede by sa nemuseli dožiť ani Pim-pimove vnúčatá. Preto mu stačí také usporiadanie, pri ktorom sa nepoužije viac ako dvojnásobok najmenšieho možného počtu krabíc, ale výsledok potrebuje rýchlo, lebo pelikán čaká. Napíšte Čin-činovi program, ktorý nájde takéto usporiadanie. Výstupom je počet potrebných škatúl a zoznam obsahujúci pre každý predmet jeho objem a poradové číslo škatule, do ktorej má byť pobaľený. Na poradí predmetov vo výstupe nezáleží. Takisto krabice si môžete očíslovať v ľubovoľnom poradí. Ak súčet objemov niekoľkých predmetov neprevyšuje objem krabice, tieto predmety sa dajú do jednej krabice zabaľiť.

Riešenie musí obsahovať okrem popisu aj dôkaz správnosti, t.j. dôkaz, že vaše riešenie skutočne neprekročí dvojnásobok optimálneho počtu škatúl.

Tým, ktorým sa zdá byť tento príklad ľahký, odporúčame vyriešiť pozmenené zadanie: Nájsť čo najefektívnejší program, ktorý nájde počet škatúl a rozmiestnenie predmetov tak, aby ich nebolo viac ako tri polovice optimálneho počtu (pri nepárnych číslach zaokrúhľujeme podiel nahor). Samozrejme aj s dôkazom.

#### 1434. O Burundijských mapách

30

Kmene v Burundi sa konečne rozhodli žiť v pokoji a mieri. Kráľ Burundi sa tomuto ich rozhodnutiu najskôr veľmi začudoval, zdalo sa mu, akoby ani svoju vlastnú zem nespoznával. Ale on, učená hlava, našťastie vedel, čo robiť. Takmer všetci Burundijčania boli bojovníci, a keď z nich opadne vlna nadšenia z nastoleného mieru, všimnú si, že stratili prácu. Treba ich teda niečím zamestnať, aby si túto holú skutočnosť všimli čo najneskôr. „Ale čím?“ hŕtal kráľ. A vtedy na to prišiel. Treba nakresliť mapu Burundi. Potom si ju bude chcieť každý obkresliť a doma, ako hrdý Burundijčan, vyvesiť na stenu. To zaberie dosť času, lebo v Burundi majú len päť farbičiek, ktoré u nich zabudol nejaký stroskotanec. Lenže vychýrený a konkurzom vybraný kreslič máp, jediný, ktorý sa v Burundi našiel, vie, že mapa má každé územné teritórium zafarbené takou farbou, akou nie je zafarbené žiadne susedné teritórium. A v Burundi je len tých päť povestných farbičiek...

ÚLOHA: Napíšte kresličovi máp program, ktorý nájde zafarbenie mapy Burundi najviac piatimi rôznymi farbami. Vstupné hodnoty pre váš program sú:  $n$  – počet kmeňov v Burundi a pre každý kmeň zoznam susedných kmeňov a zoznam kmeňov, ktoré susedia so zvyškom sveta. Dva kmene (prípadne kmeň a zvyšok sveta) sú susedné, ak ich teritória majú spoločnú hranicu. Teritórium každého kmeňa je súvislý útvar. Výstupom je zafarbenie jednotlivých územných teritórií v Burundi (aj zvyšok sveta musí mať nejakú farbu, tá sa však samozrejme môže vyskytovať aj na nejakých s ním nesusedných teritóriách).

Príklad:

Vstup:

počet kmeňov  $n = 6$

susedia kmeňa 1: 2, 3, 4 a zvyšok sveta

susedia kmeňa 2: 1, 4, 6 a zvyšok sveta

susedia kmeňa 3: 1, 4, 5, 6 a zvyšok sveta

susedia kmeňa 4: 1, 2, 3 a 6

susedia kmeňa 5: 3, 6 a zvyšok sveta

susedia kmeňa 6: 2, 3, 4, 5 a zvyšok sveta

susedia zvyšku sveta: 1, 2, 3, 5 a 6

Výstup:

Zafarbenie teritórií:

kmeň 1: zelená

kmeň 2: červená

kmeň 3: červená

kmeň 4: žltá

kmeň 5: zelená

kmeň 6: biela

Zvyšok sveta: žltá

## 1435. O Števovej sieti III

17

Kamaráti si úspešne zvolili šéfa a s elánom sa pustili do písania prefikanej sieťovej aplikácie (PSA). Ale Števa nebavilo dlho na strome sedieť a do laptopa hľadiť a začal presviedčať svojich kamarátov nech sa idú ešte pozrieť, ako svet vyzerá z najbližšieho kopca. Ale kamaráti boli príliš zaujatí tvorbou PSA a vôbec sa im nechcelo plahočiť sa na nejaký kopec. A tak sa Števo rozhodol, že pôjde sám. Zbalil laptop aj obidve šnúry, ktoré ho spájali so susednými počítačmi a svižným krokom zmizol z dohľadu.

Kamaráti medzičasom dopísali PSA a prvýkrát ju spustili. V programe však bola chyba a tá mala katastrofálne následky – každému počítaču sa vymazali nielen všetky dôležité informácie o sieti a o šéfovi ale dokonca aj jeho vlastné meno. Úbohým kamarátom teraz nefunguje žiaden z ich algoritmov a sú z toho úplne zúfalí.

ÚLOHA: Všetky počítače sú pospájané do jedného radu, každý počítač okrem dvoch krajných je spojený so svojimi dvoma susedmi priamymi linkami, krajné počítače sú spojené prirodzene len s jedným susedom. Každý počítač má svoje linky očíslované číslami 1 a 2 (prípadne len 1). Ak sú počítače A a B spojené linkou, môže napríklad počítač A poslať správu počítaču B, správa sa zaradí do mailboxu počítača B.

Počítače však nemajú žiadne jednoznačné označenie. Úlohou je očíslovať ich číslami od 1 po  $p$ , kde  $p$  je počet počítačov v sieti. Číslo 1 môže mať ľubovoľný z krajných počítačov, jeho sused má číslo 2 a tak ďalej a číslo  $p$  má druhý krajný počítač. Napíšte program, ktorý spustia všetci Števovi kamaráti na svojich laptopoch, pričom po jeho skončení každý z počítačov pozná svoje číslo.

Pre komunikáciu medzi počítačmi máte k dispozícii rovnaké procedúry ako v príklade 1415, okrem procedúry *MyName*.

Snažte sa, aby počítače skončili skôr, ako študenti podľahnú beznádeji a rozhodnú sa zabiť celú počítačovú sieť a bežať za Števom (toho by aj tak nedobehli a ešte by mohli zabúdiť). Linky sú pomalé a preto sa snažte, aby ste toho neprenášali zbytočne veľa. Číslo  $p$  na začiatku behu programu nie je známe. Môžete však predpokladať, že je nepárne. Nepoužívajte generátor náhodných čísel.

## 1441. O prefikanom špiónovi

27

Agent 00 Ferdo pracuje už niekoľko desaťročí pre SIS (Slovenskú informatickú spoločnosť) na istej nemenovanej americkej univerzite. Cieľom jeho misie je získať čo najviac technických plánov najnovších algoritmov. Za tie roky sa vypracoval na vedúcu pozíciu inštitútu, takže všetky práce študentov prechádzajú najskôr práve jeho rukami. On sa ich pochopiteľne snaží čo najrýchlejšie poslať kolegom do SIS, načo používa špeciálnu vysoko-frekvenčnú vysielaciu.

Americká kontrarozviedka mu už je na stope, hlavne vďaka tzv. CIA (Capture Incoming Alfawave) spôsobu zachytávania prichádzajúcej správy. Zariadenia CIA majú rozmiestnené po celom kontinente, každé vie určiť približný smer k vysielacu prichádzajúcej správy. Agentovi A000H sa podarilo získať zoznam všetkých týchto zariadení, spolu s výsledkami posledných meraní. Zoznam začína číslom  $n$  (počet záznamov na zozname), pre každé  $i \in 1..n$  obsahuje pozíciu  $(x, y)$  i-teho prístroja CIA ako aj kruhovú výseč zaznamenávajúcu smer pravdepodobného výskytu Ferdovho vysielacza – pre výseč sú tam dve hodnoty: začiatkový uhol smeru (v stupňoch,  $0^\circ$  je smerom na východ, uhol stúpa proti smeru hodinových ručičiek) a kladný vnútorný uhol kruhovej výseče (najviac  $90^\circ$  stupňov).

ÚLOHA: Napíšte program, ktorý zistí, či američania vedia z daných hodnôt usúdiť približnú Ferdovu polohu, t.j. či existuje také miesto, pre ktoré by boli všetky informácie z prístrojov CIA správne (a navzájom nevylučujúce sa). Táto informácia je veľmi dôležitá, aby mohol byť Ferdo včas varovaný, preto sa sústreďte na rýchlosť vášho programu.

Američania ešte nezistili, že Zem je guľatá, preto používajú rovinné mapy kontinentu. Príklad:



Vstup:

 $n = 3$ 

x	y	zač. uhol	vnút. uhol
1	2	315	90
2	-0.4	90	45
2	3	180	90

Výstup:

Nemajú šancu zistiť jeho polohu.

**1442. O Pakovači**

[25]

V tomto čase na súostroví Kiribati vrcholía veľkolepé oslavy pri príležitosti zavedenia telefónnej siete. Niektorí triezvi domorodci však upozorňujú na fakt, že ostrovy nemajú dostatok dreva na vytlačenie telefónnych zoznamov. Bohatí Kiribatčania sa totiž vyznačujú dlhými menami. Vzhľadom na skutočnosť, že sú na svoje mená nesmierne hrdí a kvôli telefónu by si ich nenechali zmeniť, rozhodla sa telefónna spoločnosť mená v telefónnom zozname „pakovať“ a to nasledovne: každá  $n$ -krát sa opakujúca postupnosť znakov PZ sa môže zapísať aj v tvare  $n[PZ]$ . Napríklad **ababab** môžeme zapísať aj ako **3[ab]**. PZ môže byť už spakovaná postupnosť, takže konečný spakovaný zápis môže obsahovať ľubovoľný počet vnorených zátvoriek.

ÚLOHA: Napište program, ktorý na vstupe dostane postupnosť znakov **a...z** a ktorý pre túto postupnosť nájde najkratší možný zápis (ak takýchto zápisov existuje viac, stačí jeden z nich). Najkratší zápis je zápis skladajúci sa z najmenšieho počtu znakov, teda vrátane znakov **[, ]**, a číslíc.

Príklad:

Vstup:

baaaaaaaaaababababc

aaaaaaaaabcaaaaaabc

Výstup:

b12[a]4[ab]c

(dĺžka je 12 znakov)

a2[7[a]bc]

(dĺžka je 10 znakov)

**1443. O kľúčikoch**

[25]

V softvérovom družstve SoDr majú ďalšieho zákazníka. Tentokrát sa na nich obrátil nemenovaný výrobca kľúčikov. Kľúčiky sa vyrábajú z dopredu pripravených výliskov vybrusovaním. Pre každý typ výlisku je známe číslo  $n$  – šírka „aktívnej“ časti kľúčika, a číslo  $k$  – maximálna povolená hĺbka výbrusu.

Pri danom type výlisku možno kľúčik reprezentovať ako postupnosť celých čísel  $a_0, \dots, a_n$ , pričom  $a_i$  znamená hĺbku výbrusu vo vzdialenosti  $i$  od začiatku „aktívnej“ časti kľúčika. Medzi jednotlivými celočíselnými pozíciami je výbrus vedený po úsečke. Pre túto postupnosť ďalej platí:  $a_0 = a_n = k$ ,  $|a_i - a_{i+1}| = 1$  ( $0 \leq i < n$ ),  $0 \leq a_i \leq k$  ( $0 \leq i \leq n$ ).

Výrobca práve zahájil výrobu nového typu výliskov a rád by vedel, koľkých zákazníkov môže uspokojiť predajom zámkov a kľúčikov vyrobených na báze tohto výlisku (každý zákazník chce mať prirodzene iný kľúč).

ÚLOHA: Napište program, ktorý načíta typ výlisku (t.j. čísla  $n$  a  $k$ ) a určí, koľko rôznych kľúčikov možno vybrúsiť z takéhoto výlisku.

Napríklad pre  $n = 4$ ,  $k = 2$  možno vyrobiť 2 rôzne kľúčiky.

**1444. O veľkom suchu**

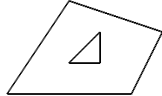
[20]

Ľudia sú už raz takí. Ak im niekto rozpráva, čo zlé ich postihlo, povedia: „To nič nie je, keby si vedel, čo sa stalo mne...“ Santo a Banto nie sú výnimky. Teraz sú obaja veľmi smutní. Prednedávnom sa rozhodli, že nebudú len v krčme vysedávať a posadili si vo svojich záhradkách zeleninu. Ale teraz už dva mesiace nepršalo a nakoniec im všetko vyschlo. Zem úplne stvrdla a popraskala. Najprv sa začali vytvárať len štrbiny, tie sa však postupne predlžovali, popretínali, no a teraz sú ich záhradky už samá kryha. A tak Santo s Bantom sedia znovu v krčme a hádajú sa, či záhrada je viac popraskaná. Keďže žiaden nedokázal toho druhého presvedčiť o svojej pravde, vybrali sa do svojich záhrad

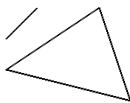
a začali kryhy počítať. Tak teraz smutne chodia a počítajú a počítajú. Pomôžte im ich spočítať, lebo na oblohe sa začali objavovať prvé obláčiky a ak začne pršať, svoj spor nikdy nevyriešia.

ÚLOHA: Na vstupe je zadané  $n$  – počet miest, v ktorých sa štrbiny popretínali a dvojice priesečníkov, medzi ktorými vedie štrbina. Viete, že štrbiny sa v ďalších miestach už nepretínajú. Spočítajte, na koľko kryh štrbiny rozdelili záhradu. Za kryhu sa samozrejme počíta aj plocha od okrajových štrbín po plot (čím viac, tým viac).

Príklad:



3 kryhy



2 kryhy

#### 1445. O Števovej sieti IV

[30]

Konečne sa kamarátom podarilo očíslovať sa a sú na to právom hrdí. Ale skutočnosť, že majú všetky počítače zapojené do jednej rady, ich netešila. Veď koho by aj, keď si stále ten na jednom konci s tým na druhom konci posielajú správy a všetkých ostatných to spomaľuje... Nenašli oni iné riešenie tohto problému ako sa porozpájať a náhodne potom pozapájať. I rozpájali oni a spájali sa a zrazu si tak uvedomili, že im chýba Števo. Ako mysleli, tak mysleli, no veruže nikto z nich si nemohol spomenúť, kam ten Števo zmizol. Húťali, húťali a vyhúťali, že on určite spadol zo stromu a v tej kope machu ho nie je vidno. Alebo tam dolu nie je mach? Nó, na tom predsa nezáleží. Rozhodne ho nie je vidno, jeho laptop tiež nie a to začína byť povážlivé. Čo ak osud úbohého Števa postretnie aj niekoho z jeho kamarátov? Potom by ich bolo zase o jedného menej, aj keď to nie je to najdôležitejšie. Oveľa horšie by bolo, keby sa im potom sieť rozpadla na viacero častí. A tu ich napadlo: Veď my musíme zistiť, či je medzi nami niekto taký dôležitý, že mu nemôžeme dovoliť spadnúť!

ÚLOHA: Počítače sú náhodne pospájané. Každý počítač má svoje meno (číslo, rôzne od ostatných počítačov) a linky na svojich susedov označené ich menami. Napríklad ak počítač s číslom 7 je spojený s počítačom č. 3, potom ich spoločnú linku má počítač č. 7 označenú číslom 3 a počítač č. 3 ju má označenú číslom 7. Dva počítače sú spojené nanajvýš jednou linkou. Môžete predpokladať, že jeden z počítačov má číslo 1.

Vašou úlohou je napísať program, ktorý spustia Števovi kamaráti na svojich laptopoch a po jeho skončení bude každý z nich vedieť, či sú pospájaní tak, že po odpojení hociktorého počítača zostane sieť súvislá.

Pre komunikáciu medzi počítačmi máte k dispozícii rovnaké procedúry ako v príklade 1415, okrem procedúry *MyName*, ktorá vracia číslo a nie reťazec. Okrem týchto procedúr môžete používať funkciu *KthLink*( $k$  : **integer**) : **boolean**, ktorá vráti, či je  $k$ -ta linka zapojená alebo nie.

#### z1411. Z univerzitetnej knižnice

[15]

„Oook!“ povedal knihovník, keď zistil, že sa stala chyba. Mrakoplaš zase poplietol abecedu. V univerzitetnej knižnici sa knihy triedia podľa začiatočného písmena a tie ktoré sa začínajú na rovnaké písmená sa triedia podľa počtu strán. Mrakoplaš si ako obyčajne vymenil dve písmená z abecedy a teraz treba úseky kníh začínajúcich na tieto dve písmená vymeniť späť tak, aby knihy v úsekoch zostali správne usporiadané. Je to však strašný problém, lebo v knižnici už nie je ani jedna voľná polica.

ÚLOHA: Dané sú čísla  $n, i, j, k, l$ , kde  $1 \leq i \leq j < k \leq l \leq n$  a pole  $A[1..n]$  celých čísel. Prvky si rozdelíme na päť úsekov:  $a_1 \dots a_{i-1}$ ,  $a_i \dots a_j$ ,  $a_{j+1} \dots a_{k-1}$ ,  $a_k \dots a_l$ ,  $a_{l+1} \dots a_n$  (niektoré úseky môžu mať aj nulovú dĺžku). Preusporiadajte prvky tak, že

druhý a štvrtý úsek budú navzájom vymenené, t.j. pole bude vyzeráť takto:  $a_1 \dots a_{i-1}, a_k \dots a_l, a_{j+1} \dots a_{k-1}, a_i \dots a_j, a_{l+1} \dots a_n$ . Okrem poľa  $A$  môžete použiť iba premenné typu **integer** (t.j. nepoužívajte iné polia, súbory a podobne).

Napríklad pre  $n = 7, i = 2, j = 3, k = 6, l = 6$   $A = (1, 2, 3, 4, 5, 6, 7)$  je výsledkom  $A = (1, 6, 4, 5, 2, 3, 7)$

#### z1412. Zorov závet

[15]

„Aargh...“ vysvetlil Zoro a zomrel. Zanedlho prišli dediči a začali sa domáhať vyplatenia životnej poisťky. V poisťovni im oznámili, že podľa zmluvy, keďže sa nejedná o vysokú čiastku, peniaze pozostalým vyplatia do sto dní. Dediči sú veľmi netrpezliví a chceli by vedieť, koľko dní už od Zorovej smrti uplynulo.

ÚLOHA: Napíšte program, ktorý načíta dátum Zorovej smrti, dnešný dátum a vypíše, koľko dní uplynulo od jeho smrti. Úradníci v poisťovni sú však veľmi leniví a neporiadni, preto sa môže stať, že niektoré prípady budú vybavovať aj niekoľko rokov. Rozdiel v zadaných dátumoch môže byť aj väčší ako sto dní. Môžete však predpokladať, že obidva dátumy sú v rokoch 1901–1999.

#### z1413. Záletný zemepán

[17]

Zemepán Zoltán je veľký záletník. V okolí jeho zámku žije  $n$  krásnych zemepání. Každá má zámok so spúšťou vežičiek a to sa zemepánovi Zoltánovi veľmi páči. V poslednom čase však na niektorých cestách striehnu žiarliví zbojníci a Zoltán sa bojí tadiaľ prechádzať. Trápia ho pochybnosti, či sa môže po bezpečných cestách dostať ku každej z krásnych zemepání.

ÚLOHA: Dané sú čísla  $n$  a  $k$ , kde  $n$  je počet krásnych zemepání a  $k$  je počet bezpečných ciest. Zámky sú očíslované číslami  $1 \dots n+1$ , zámok číslo  $n+1$  je zemepánovo sídlo. Ďalej je daný zoznam bezpečných ciest. Každá cesta vedie medzi dvoma zámkami a je zadaná ich číslami. Napíšte program, ktorý zistí, či sa zemepán môže dostať ku každej zemepanej, pričom môže cestovať aj cez viaceré zámky.

Napríklad pre  $n = 3, k = 2$  a cesty 4 3 a 1 3 sa Zoltán nemôže dostať k zemepani číslo 2, preto program vypíše NIE.

#### z1414. Zátvorkove prvočísla

[M35]

Zátvorka je učiteľom matematiky. Celkom dobrý učiteľ. Ale čo je veľa, to je veľa. Na poslednej hodine matematiky ho žiaci strážne nahnevali, dokonca na Ferovi, zvanom Zúrivec, bol nútený zlomiť obľúbené ukazovátka. Nuž im dal ťažkú, preťažkú domácu úlohu. Keď sa vrátil po náročnom dni domov, schladil sa jeho hnev a objavili sa pochybnosti. Čo ak ho žiaci tromfnú a niekto nájde lepšie riešenie ako on, učiteľ Zátvorka?

ÚLOHA: Pomôžte žiakom dokázať Zátvorkovi, že sú pod slnkom aj lepší matematici ako on. Nájdite (za výdatnej pomoci vášho programu) čo najdlhšiu aritmetickú postupnosť prvočísel. Aritmetická postupnosť je postupnosť čísel taká, že rozdiel každých dvoch po sebe idúcich čísel je rovnaký, dĺžkou postupnosti rozumieme počet jej členov. Hodnotiť budeme hlavne vami nájdenú postupnosť, ale aj postup, akým ste túto postupnosť našli. Preto priložte aj program a slovne popíšte váš postup hľadania.

Príkladom takejto postupnosti dĺžky tri je 3, 7, 11.

#### z1415. Zelená žabka

[12]

Zrazu sa okolo žabky Júlie zotmelo. Skutočne. Zlý Mórisko na ňu hodil svoj klobúk. Na tom by zatiaľ nebolo nič divné, ale to ešte neviete, že Móriskov klobúk má pôdorys tvaru obdĺžnika s rozmermi  $m \times n$ . Júlia začala skákať a volať o pomoc: „Kvááááá“. Skákala tak zbesilo, že si pri tom ani len nevšimla, ako naráža do klobúka a pod rovnakým uhlom odrazu ako bol uhol dopadu sa od jeho stien odráža. Popri tom s úžasom zistila, že klobúk leží na jahodovisku. Vždy keď doskočila na jahôdku, tak ju zjedla. Takže to napokon nebolo až také zlé.

ÚLOHA: Dané sú  $p$ ,  $n$ ,  $m$  a pole  $A[1..N, 1..M]$ . Na každom políčku poľa  $A$  bude udaný počet jahôdok, ktoré sa tam nachádzajú. Ďalej sú dané  $x$  a  $y$ , ktoré určujú pozíciu žabky, a  $dx$ ,  $dy$  určujúce smer jej prvého skoku (ak Júlia nenarazí do steny, tak po prvom skoku budú súradnice jej dopadu  $[x + dx, y + dy]$ ). Napíšte program, ktorý odsimuluje  $p$  skokov žabky. Ak na pozícii dopadu rastú ešte nejaké jahôdky, Júlia z nich jednu zje. Zobrazujte celú situáciu na obrazovke pomocou znakov po každom Júliinom skoku. Po skončení simulácie vypíšte počet jahôdok, ktoré Júlia zjedla.

Napríklad keď  $dx = -2$ ,  $dy = -4$ , potom žabka z pozície  $[6, 4]$  skočí na pozíciu  $[4, 1]$  (po odraze od steny) a po ďalšom skoku bude na políčku  $[2, 5]$ .

#### z1421. Záviš a jeho rozhlasový sen

22

Záviš je obchodníkom od narodenia. Raz ho napadlo, ako dobre by sa dalo zbohatnúť, keby si založil vlastné rádio. Mal by kopec peniažkov za reklamu, jedným slovom rozprávka.

I začal zisťovať, kto by mal záujem v Záviši rádiu odvysielať reklamu. Pre každý deň od dňa  $d$  (predpokladaného začiatku vysielania) si vypočítal, koľko bubáčikov by zarobil na reklame. Nadšený si ľahol spať. Keď sa ráno zobudil, napadla ho hrozná myšlienka: veď on nemá žiadny vysielateľ! Mohol by si nejaký prenajať, ale to stojí dosť bubáčikov. Navyše, vysielateľ sa dá prenajať iba na jedno súvislé obdobie. Chudák Záviš, schytil ceruzku a papier a začal počítat, odkedy dokedy je najvýhodnejšie prenajať si vysielateľ, aby čo najviac zarobil...

ÚLOHA: Napíšte program, ktorý načíta  $c$  – cenu za jeden deň prenajatia vysielateľa, počet dní  $d$ , pre každý deň  $b_i$  – koľko v ten deň môže Záviš zarobiť na reklame a vypočíta, odkedy dokedy je pre Záviša najvýhodnejšie prenajať si vysielateľ.

Napríklad pre 6 dní, pri cene za jeden deň vysielania 20 a zárobky v jednotlivé dni: 18, 35, 6, 80, 15 a 21 je najvýhodnejšie prenajať si vysielateľ od 2. do 4. dňa. Spolu zarobi  $35 + 6 + 80 - 3 \times 20 = 61$  bubáčikov.

#### z1422. Ziskuchtiví zbojníci a Kiribati

15

Súostrovia Kiribati má problémy. Od istého času ich pravidelne (každý prvý pondelok v mesiaci) prepadávajú Ziskuchtiví zbojníci. Nakoniec Kiribatčania povedali „dosť“ a rozhodli sa nakúpiť v miestnom obchode s domácimi zvieratami levy a umiestniť ich na pobreží. Najbližšie keď si Ziskuchtiví zbojníci prídu zobrať, čo im nepatrí, prídu mnohí o ruky, nohy alebo o hlavu (tí (ne)šťastnejší). Už treba len vyrátať dĺžku pobrežia (aby zistili, koľko levov treba) a Ziskuchtiví zbojníci, traste sa!

V kráľovskej knižnici si Kiribatčania našli mapu súostrovia, no nech robia, čo robia, dĺžku pobrežia nevedia zistiť. Zistili iba nasledujúce skutočnosti: Mapa je rozdelená na  $m \times n$  štvorčekov, každý štvorček predstavuje pevninu (je označený 1) alebo more (je označený 0), teda žiaden štvorček neobsahuje pevninu aj more zároveň. Štvorčeky dotýkajúce sa navzájom hranou (nie rohom) patria do toho istého ostrova. Žiadny ostrov sa nedotýka okraja mapy (inak by s nevedelo, či mimo mapy nepokračuje), teda na okraji mapy je more. Každý ostrov na mape patrí do súostrovia Kiribati a každý ostrov zo súostrovia Kiribati je na mape. Pobrežie tvoria hrany medzi štvorčekom pevniny a štvorčekom mora. V súostroví Kiribati sa môžu nachádzať (už bez ostrovov) aj lagúny, vnútorné jazerá a iné mláky, ktorých brehy sa do pobrežia nezarátavajú.

ÚLOHA: Napíšte program, ktorý vypočíta dĺžku Kiribatského pobrežia (tj. súčet dĺžok pobreží všetkých ostrovov na Kiribati), pričom sú dané rozmery mapy  $m$  a  $n$  a mapa súostrovia.

Príklad:

Vstup:  
 $m = 5, n = 6$   
 Mapa súostrovia:  
 000000  
 011100  
 010110  
 011110  
 000000

Výstup:  
 Dĺžka pobrežia je 14.

#### z1423. Zlato nad zlato

► [??]

V softférovom družstve SoDr majú novú úlohu. Tentoraz ide o program pre jedno zlatníctvo. Nanešťastie každý z  $n$  pracovníkov vytvoril vlastný program a medzi sebou sa nevedia dohodnúť, ktorý použiť. Programy (očíslované od 1 až  $n$  podľa osobného kódu pracovníka) sa líšia výsledným číslom udávajúcim počet karátov testovaného zlatého predmetu.

Po búrlivej polemike zavrhlí programy s najvyšším odhadom (aby zákazník nemusel veľa platiť), aj s najnižším odhadom (aby zákazník nemohol byť odsúdený za krádež). Nakoniec sa rozhodli pre zlatú strednú cestu, teda pre taký program, že keď rozdelíme zvyšné programy na dve skupiny podľa toho, či dávajú menší alebo väčší výsledok, tak ich bude buď rovnako, alebo tých s menším výsledkom bude o jeden menej.

Celé družstvo začalo búrlivo oslavovať, ako múdro to vyriešilo, až kým nezačali vyberať príslušný program. Naraz všetkým zamrzol úsmev na tvári.

ÚLOHA: Napíšte program, ktorý dostane na vstupe  $n$  rôznych celých čísel (výsledkov programov jednotlivých pracovníkov) a vypíše výsledok, aký dá program vybrať podľa kritérií uvedených vyššie.

Napríklad pre  $n = 7$  a výsledky programov 10, 4, 3, 15, 2, 6, 1 vybraný program dá výsledok 4.

#### z1424. Zátvorkove postupnosti

[20]

Zátvorka kdesi čítal o takejto postupnosti: Vezmime si ľubovoľné prirodzené číslo  $n > 1$ . Ak je párne, vydelíme ho dvomi, ak je nepárne, vynásobíme ho tromi a pripočítajme jednotku. Toto môžeme opakovať dovtedy, kým nie je  $n$  jedna.

Hm, povedal si Zátvorka, to by bolo niečo pre mojich neposlušných žiakov! Ak budú zase tak vyvádzať ako minule, dám im nejaké číslo, a nech si počítajú. Veď sa oni naučia poslúchať! Má to však jednu nevýhodu: ak to číslo bude veľmi veľké, bude sa ťažko pamätať. Nuž, Zátvorka teraz chudák sedí a rozmýšľa, aké číslo by sa najviac hodilo.

ÚLOHA: Pomôžte Zátvorkovi. Nájdite (za výdatnej pomoci vášho makačského programu) také číslo  $n$ , že  $p(n)/c(n)$  bude čo najväčšie, kde  $p(n)$  je počet členov Zátvorkovej postupnosti a  $c(n)$  je počet cifier čísla  $n$  (v desiatkovej sústave).

Napríklad pre číslo 7 vyzerá Zátvorkova postupnosť takto:

7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

Teda,  $p(n) = 17$ ;  $c(n) = 1$

#### z1425. Zrazené vláčiky

[10]

V Zmrzlinove to majú ťažké. Nielen, že je tam veľká zima ľuďom, ale dokonca aj vláčikom. A tak sa taký vláčik radšej stále pohybuje a nikomu nezastavuje, aby neprechladol alebo nezmrzol. Keď je vláčik malý, mama mu vždy hovorí, kade má chodiť, aby sa neustratil. A keď vláčik vyrastie a mama už naňho nemá čas, tak si už svoju trasu pamätá a chodí po nej stále dookola aj bez maminej pomoci. Bojí sa však toho, že doňho nejaký iný vláčik narazí. Vtedy je zle a treba ísť do opravy, čo je pre vláčiky asi také neprijemné ako je pre ľudí chodenie k zubárovi. Alebo možno ešte neprijemnejšie.

ÚLOHA: Zmrzlinovo je obdĺžniková krajina o rozmeroch  $80 \times 25$ . Napíšte program, ktorý najskôr načíta počet vláčikov  $n$  a postupne pre každý vláčik jeho dĺžku  $d$ , farbu  $f$  a jeho okružnú trasu. Okružná trasa je daná postupne súradnicami políčok v Zmrzlinove, ktoré musia susediť hranou (zároveň susedia hranou prvé a posledné zadané políčko). Potom váš program na prvé až  $d$ -te políčko trasy umiestni vláčik farby  $f$ , teda ho vykreslí na obrazovku. Po vykreslení všetkých vláčikov sa začnú naraz rovnakou rýchlosťou pohybovať po zadaných trasách stále dookola dovtedy, až kým sa nejaké vláčiky zrazia.

**z1431. Závišova záhradka**

10

Záviš zasa začal zarábať. Založil znova zaujímavú záhradku. Zárobok zo záhradky – žiadna zábava. Žeby Záhradkárske zápolenie zaujímavých záhradiek (známe značkou ZZZZ)? Znamená ZZZZ, že Záviš získa zlato, zlato, zlato, zlato? Žiadajú zvyčajné zábradlie záhradky (zoštvorcované). Zvíťaziť znamená, že žiadna známa záhradka zo Zeme zasiala záhony zrovna zvíťazeným zobrazením. Zvíťazí ziskuchtivý Záviš? Zohnal zoznam známych záhradiek. Žeby záhradky zo zoznamu zobrazovali Závišovu záhradku? Začal zrovnávať...

ÚLOHA: Napíšte program, ktorý načíta  $n$  (veľkosť záhradky) a dve matice typu  $n \times n$  pozostávajúce z núl a jedničiek (0 značí prázdne miesto, 1 záhon). Nato vypíše, či sa dané záhradky podobajú. Dve záhradky sa podobajú, ak možno dostať jednu z druhej pomocou niekoľkých operácií otáčania o  $90^\circ$  a preklápanie podľa osi  $x$  alebo  $y$  (osová súmernosť).

Príklad:

Vstup:

$n = 4$

1 0 0 0	0 0 1 0
0 0 1 1	1 0 1 0
0 1 0 0	0 1 0 0
0 0 1 0	0 0 0 1

Výstup:

Áno, podobajú sa (druhú možno dostať z prvej otočením o  $90^\circ$  doprava a preklopením podľa osi  $x$ ).

**z1432. Zahraničné spravodajstvo**

► 20

Z titulkov dnešných správ vyberáme: Rozpor medzi Kiribati a Burundi, Celosvetový štrajk pokračuje, Na severozápade sa premnožili gorily.

A teraz k jednotlivým správam podrobnejšie: Medzi Kiribati a Burundi sa strhla roztržka po tom, ako Kiribatská princezná odmietla Burundijského náčelníka. Podľa oficiálnych zdrojov princezná nemala k tomuto rozhodnutiu žiaden dôvod a náčelník je vraj šumný mládenec. Napokon sa obe strany dohodli na tom, že si princezná s náčelníkom zahrajú partičku tamojšej národnej hry a kto vyhrá, ten vyhrá. Médiá sa po dlhšom mlčaní rozhodli zverejniť aj dlho utajované pravidlá: Hrajú dvaja hráči – princezná a náčelník. Hráči roztrhnú nejaké dva princeznine náhrdelníky a guľôčky z nich položia na hrací plán s erbami oboch krajín tak, aby sa nepomiesali. Potom striedavo z plánu odoberajú určitý počet guľôčok, ktorý nesmie byť nulový. Pritom každý hráč môže na jeden ťah odobrať guľôčky iba jedného druhu (buď z jedného alebo z druhého náhrdelníka, nikdy však nie z oboch) a ich počet nesmie prevýšiť vopred stanovenú konštantu  $k$ . Prehráva ten, kto bude mať po vyprázdnení hracieho plánu buď nepárny počet guľôčok (alternatíva A) alebo ten, kto vezme poslednú guľôčku z plánu (alternatíva B) alebo ten, kto nevezme poslednú guľôčku z plánu (alternatíva C). Konkrétne alternatíva hry, ako aj konštanta  $k$  sa určia na mieste losovaním.

ÚLOHA: Napíšte program, ktorý určí hráča, pre ktorého existuje vyhrávajúca stratégia v danej hre. Váš program by mal teda rozhodnúť, ktorý z hráčov má šancu vyhrať bez ohľadu na to, ako bude hrať jeho súper. Súčasťou programu by mal byť aj poradca pre vyhrávajúceho hráča, ktorému by sme postupne zadávali ťahy protihráča a poradca by radil, ako má hrať vyhrávajúci hráč. Vstupnými hodnotami pre váš program budú čísla  $n_1$

a  $n_2$  udávajúce počet guľôčok na náhrdelníkoch, ďalej číslo  $k$  a alternatíva hry H. Princezná ťahá prvá.

V alternatíve A môžete predpokladať, že súčet  $n_1 + n_2$  je nepárny.

#### z1433. Zoltán tipuje

► [20]

Zemepán Zoltán je veľkým milovníkom koní. Pravidelne chodí na dostihy. Veľmi rád tipuje. S partiou podobne postihnutých priateľov sa každú nedeľu poobede stretávajú v hľadisku dostihovej dráhy a diskutujú o koňoch. Raz sa rozprúdila živá diskusia o tom, kto na čo stávil. „Ja som stávil dvesto zlatých na to, že Žltá Hviezda dobehne skôr ako Potmehúd,“ povedal gróf Šternberg zo Šternbergu. „To nič nie je, mojich päťsto zlatých a sud najlepšieho vína stoja proti prsteňu pána z Ostrej Lúky, že Artefax predbehne Halifaxa,“ odvetil barón dePrášil. Sám Zoltán stávil dvadsaťpäť zlatých v striebre, že jeho obľúbenec Železník porazí Drevenú Nohu.

Vysvitlo, že všetci Zoltánovi priatelia, vrátane Zoltána samého, uzatvorili stávky tipu „kôň 1 dobehne skôr ako kôň 2“. O chvíľu bolo všetko jasné. Kone dobehli, niektorých Zoltánových priateľov sa zmocnila nevýslovná radosť, iní statočne potláčali žiaľ. Zoltán po strate dvadsaťsedem zlatých (dva minul na kolu a hranolky) sa rozlúčil s priateľmi a pobral sa domov.

Doma sa ho zmocnil nepokoj: bolo by možné, aby všetci jeho priatelia v ten deň vyhrali?

ÚLOHA: Napište program, ktorý načíta počet koní  $m$ , počet stávk  $n$  a  $n$  stávk – dvojíc čísel  $a_i, b_i$ , ktoré predstavujú stávky „kôň  $a_i$  dobehne skôr ako kôň  $b_i$ “ a vypíše jedno poradie koní také, že všetci Zoltánovi priatelia vyhrajú, alebo vypíše, že také poradie nemožno nájsť.

Napríklad pre  $m = 7$  a  $n = 4$  a stávky 1, 2 a 2, 3 a 3, 4 a 3, 1 program vypíše **Nedá sa**.

#### z1434. ZGMYTDJB CHLBND

[25]

LYZBWKDB VGNDV CGAY JDAYC ZGCJBMB JYMYILBAGA ZGMYTDJU  
CHLBNU KYMGCNJYJGNYQG NETVBAU. JBJG CHLBND NCBW OGMB WNGMD  
MYHCDYAU UJBIFYVDU HGCMBVB TBCDRLGNBVB. VGNDVBLD CB  
VDYWGMWG QGZDV CVBTMD CHLBNU LGTMUCJDJ, BMY VYHGZBLDMG  
CB DA JG. HLYJG JYLB TULVY HGJLYOUFU NBCU HGAGK. HGAGTJY DA B  
LGTCDDLUFJY (TB NEZBJVYF HGAGKD NBCQG HGKDJKBK) JYPJ CHLBNE.  
ZGMYTDJY FY HGCMBJ VDYMYV LGTCDDLGNBVE JYPJ, BMY BF HGCJUH,  
BWEA CJY QG LGTCDDLGNBMD B HLGILBA, WJGLE CJY CD WNGMD JGAU  
VBHDCBMD (BF C HGHDCGA).

KHLQBZITJPOG LNPGSNFA

UBLVP ESMV OYFF BTCU OSJBGE WSBHC LNPGSNFQW XENFEEDPZ WC QB TQTYIFOLGJ  
LYMOBGMELGJ AZIPBPL PFPEMBAI FBEM X BEKGOGMPF N FTBMMNJV ZAIYEUY XNBPSXZ  
GELPZRKL CVG POPCTG NWAAIL BZITJXC VJRXQ LYMOBGMELR DOFAC UVIMUJN W  
MUYXKWNKPH HCAQSFUS RSNPGTN E BBCVKDVRKMV EL MRTUJH YTPQY  
MBYITBOSX W BWVBGRADU NWIBOFEMELGJ LEELJAEI AE PJROVPECEI HDGNVEEI FE  
WSBHC AICUJYE CA B TCUQUJNX RFEGGOG GQ KR Z VZPLVP XVCKVRCDU ZGMXC WTCIEI  
XSPFPRG ORQWTE MBYITBOC FPIEBBG TTFGSBF VGJ QESFVXGKB HD UUNGK QBOTZG  
TQAVEFBIOA POCXBGINPI XADUXQ LEELJA OVPEM MBYITBOC BCBDPVWY FVSEOF MI VBGS  
ULIINB FTBIE RPZSBF MEDFMTGDVX OJRV PB PINPZ WXFGI  
VMNGQWN EIFAXWSN QCZOI

#### z1435. Záhada sennej nádchy

[20]

I takto Mohamed riekol: „Keď hodiny šiestu hodinu odbijú, nech ste kdekoľvek na Zemi, bez otáľania klaknete si, tvárou k svätým miestam v Mekke pozerajúc, modliť sa

budete.“ I po krátke dramatickej pauze pokračoval: „A teraz chodte, poslovia moji, rozptýľte sa po celom svete, kam vás vaša vôľa a nohy budú viesť. Hapčíí...“ Po chvíli, keď sa vysmrkal, dokončil svoj prejav: „To som vám chcel povedať!“ Nedalo mu, a ešte nakoniec tichým hlasom do vetra zašepkal: „Bodaj by mi aspoň jeden z tých babrákov doniesol liek proti sennej nádche...“

A tak chodili po svete a robili, čo im bolo prikázané. I stalo sa, že o šiestej hodine boli na tom istom námestí viacerí, a vznikol malý problém: všetci si kľakli, ale keďže nikto presne nevedel, ktorým smerom je Mekka, každý pozorne pozrel pred seba a ak videl nejakého iného kľáčiaceho mohamedána, otočil sa jeho smerom vo viere, že on určite vie lepšie, kam má byť otočený (tí na krajoch, pokiaľ sa pozerajú z námestia von, alebo tí, ktorí sa pozerajú na niekoho, kto sa pozerá tým istým smerom, sa neotáčajú a nerušené sa modlia). Mohamedáni sa otáčajú vždy naraz každú sekundu (ako hodiny odbíjajú).

ÚLOHA: Dané je námestie  $m \times n$  políčok, pričom na každom políčku môže byť jeden z týchto 4 objektov: S J V Z (S je mohamedán pozerajúci na sever, ..., Z je mohamedán pozerajúci na západ).

- Napište program, ktorý simuluje stav na námestí každú sekundu počnúc šiestou hodinou. Ak sa už všetci mohamedáni pokojne modlia, program skončí.
- Môže sa stať, že niektorí z mohamedánov sa nikdy neprestanú otáčať? Ak áno, napíšte program, ktorý spočíta počet takýchto mohamedánov. Ak nie, dokážte.
- Popíšte usporiadanie mohamedánov na námestí po skončení programu z a).

Príklad:

JZJ	ZJV	ZVV
ZSV	ZVV	ZVV
ZJV	ZJV	ZJV

6:00:00    6:00:01    6:00:02

### 1511. O recidivistoch

Nepokoje v našich väzniciach narastajú do neúnosných medzí. A to nielen preto, že sú preplnené, ale aj preto, že recidivistom sú vždy pridelené nové čísla a nie tie ich, na ktoré si už zvykli. Vo väzniciach, kde sa všetci oslovujú číslami, je tento fakt veľmi nevítaný.

Všetko vyrieši nový projekt Alcatras II. Ak príde do tejto väznice ďalší väzeň, bachári zadajú do počítača jeho meno a priezvisko a program vypíše NOVY, ak tu ešte predtým väznený nebol, alebo RECIDIVISTA, ak neprišiel poprvý raz. Program tiež vypíše číslo väžňa. Novému väžňovi môže program priradiť ľubovoľné číslo, ktoré nemá iný vezeň, ale recidivistovi musí priradiť číslo z jeho predchádzajúceho pobytu. Možno predpokladať, že čísla sú priradované iba uvedeným programom.

ÚLOHA: Vytvorte horepopísaný program tak, aby fungoval čo najrýchlejšie a pre čo najväčšie množstvá väzňov. Program bude uvedený do prevádzky zároveň s otvorením väznice, takže väznica je na začiatku prázdna. Keďže Alcatras II nikdy nepadne, môže program fungovať v nekonečnej slučke. Mená väzňov sú zložené z veľkých písmen abecedy a z medzier.

Príklad vstupu a výstupu:

```
? LEX LUTHOR
NOVY 3560
? JAMES BOND
NOVY 007
? LEX LUTHOR
RECIDIVISTA 3560
? LUISE LANE
NOVY 806070
```



? JAMES BOND  
 RECIDIVISTA 007

### 1512. O hliadkach

Seržant Sergej je veliteľom hradnej stráže. Je zodpovedný za to, aby pri bráne vždy stála hliadka v počte  $k$  vojakov. Seržant je veľmi starostlivý človek. Aby nikomu zo svojich vojakov neukrivil tým, že bude musieť hliadkovať častejšie ako niekto iný, rozhodol sa viesť si presnú evidenciu. Zapisovať vždy mená všetkých, čo boli na stráží je veľmi zdĺhavé a navyše by všetky tie zápisy zaberali veľa miesta. Rozhodol sa teda, že bude skupinky svojich vojakov kódovať čo najúspornejšie – číslami.

ÚLOHA: Napište seržantovi procedúry pre kódovanie a dekódovanie skupín vojakov. Procedúra *Koduj* nech načíta  $n$  – počet seržantových vojakov a  $k$  – počet vojakov v hliadke ( $n > k$ ) a  $k$  (rôznych) čísel vojakov z intervalu  $1 \dots n$ . Jej výstupom bude prirodzené číslo z intervalu  $1 \dots \binom{n}{k}$ , jednoznačne kódujúce danú skupinu vojakov. Naopak, procedúra *Dekoduj* načíta čísla  $n$  a  $k$  a kód (číslo z intervalu  $1 \dots \binom{n}{k}$ ), vyprodukovaný procedúrou *Koduj* a vypíše čísla vojakov v danej hliadke.

### 1513. O leteckej spoločnosti

Novozaložená letecká spoločnosť Kiribati Air GmbH má svoj veľký cieľ: Vytlačiť z trhu všetku konkurenciu. O tom, že to myslí vskutku vážne, svedčí aj to, že najala množstvo programátorov, aby zabezpečila svojim zákazníkom čo najlepšie služby.

ÚLOHA: Napište program, ktorý zákazníkovi vypočíta minimálny čas cesty medzi dvoma mestami. Vzhľadom na to, že prevádzka lietadiel je finančne veľmi náročná, neexistuje zatiaľ priama linka medzi každými dvoma mestami – občas sa stane, že treba aj niekoľkokrát prestupovať. Dĺžka jedného letu nepresiahne 20 hodín, lietadlá na všetkých linkách premávajú každý deň podľa presne stanoveného časového harmonogramu. Lietadlá spoločnosti Kiribati Air GmbH nikdy nemeškajú, nezastaví ich žiadna búrka, hmla, či porucha.

Vstupom programu je letový poriadok (obsahuje počet liniek, pre každú linku mesto a čas odletu, mesto a čas priletu), nasledovaný niekoľkými otázkami typu „odkiaľ kam“. Výstup obsahuje pre každú dvojicu miest minimálny čas (v hodinách a minútach) potrebný na prepravu, alebo informáciu o tom, že danú cestu nemožno uskutočniť. Čas je absolútny. Meno mesta je jedno slovo.

Príklad:

Vstup:

3

Bratislava 15:00 Moskva 23:00

Moskva 05:00 BurundiDC 17:33

Moskva 22:00 Tokio 00:00

Bratislava Tokio

BurundiDc Moskva

Výstup:

Bratislava-Tokio: 33h 00m

BurundiDC-Moskva: Smola

### 1514. O tvrdohlavom učiteľovi

35

Učiteľ telocviku Sebe Vražda má tento rok veľmi hlúpych žiakov. Keďže učí telocvik vždy 2 triedy naraz a má rád poriadok, vyžaduje, aby sa žiaci na začiatku telocviku postavili do radu podľa triedy. Jeho žiakom to ale veľmi dlho trvá, tak sa rozhodol, že ich preusporiada sám. A to nie hocijako, ale tak, že vždy presunie 2 susedných žiakov naraz (myslel si, že to tak bude rýchlejšie). Teraz si láme hlavu, ako preusporiadať žiakov na najmenší počet výmen. Pomôžte Sebem, kým sa nenahnevá.

ÚLOHA: Napište program, ktorý dostane na vstup  $2n$  ( $n < 36$ ) znakov, z ktorých je  $n - 1$  znakov A,  $n - 1$  znakov B a dva susedné znaky  $m$  (ako medzera) a ak existuje, vypíše ľubovoľnú najkratšiu postupnosť výmen tak, aby na konci boli všetky písmená A naľavo od

písmen B (na umiestnení medzier na konci nezáleží). Výmena priebehu tak, že zoberieme 2 susedné písmená a presunieme ich do medzery (na ich miesto sa presunie medzera). Pri výmene nie je možné navzájom vymeniť ľavé a pravé presúvané písmeno. Ak takáto postupnosť neexistuje, program vypíše **Neexistuje**.

Príklady:

Vstup:

$n = 5$

ABBAmBAB

Iný vstup:

$n = 2$

BmBA

Výstup:

ABBAmBAB

ABBABAmmB

AmmABAABBB

AAAABmmBBB

Výstup:

Neexistuje

### 1515. O funkcionálnom programovaní I

[21]

Program vo funkcionálnom programovacom jazyku pozostáva z definícií niekoľkých funkcií. Každá funkcia má jednu alebo viacero vstupných premenných, pričom tieto vstupné premenné, rovnako ako výsledok funkcie, nadobúdajú hodnoty z množiny prirodzených čísel (pre účely tohoto príkladu budeme považovať 0 za prirodzené číslo). Tu je ukážka takéhoto programu:

$$\begin{aligned} m(x, y) &= 0 && \leftarrow && x = 0 \\ m(x, y) &= m(x-1, y) + y && \leftarrow && x > 0 \end{aligned}$$

**Klauzuly.** Definícia každej funkcie pozostáva z niekoľkých riadkov (tzv. klauzúl). Každá klauzula je tvaru:

$$\langle \text{funkcia} \rangle (\langle \text{argumenty} \rangle) = \langle \text{výraz} \rangle \quad \leftarrow \quad \langle \text{podmienka} \rangle,$$

kde *funkcia* je meno funkcie, *argumenty* je zoznam jej vstupných premenných oddelených čiarkami a *výraz* určuje hodnotu, ktorú funkcia nadobudne za predpokladu, že je splnená podmienka *podmienka*. Ak by mala byť podmienka vždy splnená, možno ju celkom vynechať.

Pozrime sa, ako sa vyhodnotí funkcia z nášho príkladu pre vstupné premenné  $x = 2$  a  $y = 3$ :

$$m(2, 3) \stackrel{1}{=} m(1, 3) + 3 \stackrel{2}{=} m(0, 3) + 3 + 3 \stackrel{3}{=} (0 + 3) + 3 = 6$$

Najprv (rovnosť 1) sa v definícii funkcie našla klauzula, ktorá zodpovedá vstupným premenným. Keďže  $x > 0$ , vybrala sa druhá klauzula z definície.  $m(2, 3)$  sa nahradilo pravou stranou rovnosti v klauzule. Hodnotu výrazu ešte stále nevieme priamo určiť, lebo sa v ňom vyskytuje volanie funkcie  $m$ . Musíme teda znovu použiť definíciu funkcie  $m$  (pre hodnoty  $x = 1$  a  $y = 3$ ). Opäť vyberieme druhú klauzulu (rovnosť 2). Vo výraze ešte stále vystupuje funkcia  $m$ , tentokrát však so vstupmi  $x = 0$  a  $y = 3$ , čiže použijeme prvú klauzulu. Dostaneme tak výraz, v ktorom sa vyskytuje už iba známa funkcia  $+$ , a ktorého hodnotu teda vieme spočítať. Ak si program dobre prezriete, zistíte, že tajomná funkcia  $m$  počíta súčin svojich dvoch vstupov.

**Výlučnosť klauzúl.** Všetky klauzuly pre jednu funkciu musia spĺňať tzv. podmienku výlučnosti klauzúl. To znamená, že pre ľubovoľné vstupné hodnoty musí byť podmienka na pravej strane splnená nanajvýš pre jednu klauzulu tejto funkcie (tým zaistíme, že funkcia má vždy jednoznačnú hodnotu). Ak nie je splnená podmienka žiadnej klauzuly, funkcia implicitne nadobúda hodnotu 0.

**Preddefinované funkcie.** V programe je možné používať preddefinované funkcie  $+$  a  $-$ . Funkcia  $+(a, b)$  vráti súčet čísel  $a, b$ . Funkcia  $-(a, b)$  vráti rozdiel čísel  $a, b$  ak  $a \geq b$  (a teda rozdiel je prirodzené číslo) alebo vráti 0, ak  $a < b$  (v tomto prípade by bol rozdiel

záporný a záporné čísla nemáme). Pre zvýšenie prehľadnosti budeme písať  $a + b$  namiesto  $+(a, b)$  a  $a - b$  namiesto  $-(a, b)$ .

**Podmienky a výrazy.** Podmienka môže obsahovať logické spojky  $\wedge$ , znamienka  $<$ ,  $>$ ,  $\leq$ ,  $\geq$ ,  $=$  a  $\neq$  výrazy. Výraz v klauzule môže obsahovať zátvorky, konštanty (ako napríklad 0, 1, 63 apod.), vstupné premenné, preddefinované funkcie a funkcie definované v programe, vrátane rekurzívnych volaní.

Ak je v podmienke niektorej klauzuly jednoznačne určená hodnota niektorej vstupnej premennej, možno túto premennú v celej klauzule nahradiť touto hodnotou. Napríklad klauzulu  $m(x, y) = 0 \leftarrow x = 0$  možno zapísať aj takto:  $m(0, y) = 0$ .

**Chvostová rekúzia.** Funkcia je napísaná chvostovou rekúziou, ak na pravej strane (vzhľadom k rovnítku) každej jej klauzuly sa vyskytuje najviac jedno rekurzívne volanie a to nie v podmienke, toto rekurzívne volanie nie je vstupnou hodnotou žiadnej inej funkcie a všetky ďalšie funkcie, ktoré sa v klauzule vyskytujú, sú napísané chvostovou rekúziou alebo sú preddefinované.

Význam chvostovej rekúzie spočíva v tom, že takúto rekúziu možno efektívnejšie realizovať (pri prepise do iných programovacích jazykov by bolo možné zapísať takúto funkciu pomocou cyklu). Náš príklad nie je napísaný chvostovou rekúziou, lebo v druhej klauzule je rekurzívne volanie  $m(x - 1, y)$  vstupnou hodnotou pre funkciu  $+$ .

Príklad: Zapíšeme teraz funkciu  $m$  pomocou chvostovej rekúzie. Pri definícii použijeme pomocnú funkciu  $m1$  s tromi vstupmi, ktorá je definovaná tiež chvostovou rekúziou.

$$\begin{aligned} m1(x, y, z) &= z \quad \leftarrow \quad x = 0 \\ m1(x, y, z) &= m1(x - 1, y, z + y) \quad \leftarrow \quad x > 0 \\ m(x, y) &= m1(x, y, 0) \end{aligned}$$

**ÚLOHA:** Napište definície dvoch dolu uvedených funkcií pomocou chvostovej rekúzie. Môžete používať aj definície iných funkcií, ale treba si ich naprogramovať (prirodzene, tiež chvostovo rekurzívne). Preddefinované sú len funkcie  $+$  a  $-$ . Nezabudnite popísať význam jednotlivých vstupných hodnôt a pomocných funkcií a odôvodniť správnosť vášho programu.

a)

$$\begin{aligned} f(0) &= 0 \\ f(1) &= 1 \\ f(n + 2) &= f(n) + f(n + 1), n \geq 0 \end{aligned}$$

b)  $g(n) = \lceil \sqrt{n} \rceil$ , kde  $\lceil x \rceil$  znamená hornú celú časť čísla  $x$ .

Príklad:

$$\begin{aligned} x &= 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ \dots \\ f(x) &= 0 \ 1 \ 1 \ 2 \ 3 \ 5 \ 8 \ 13 \ \dots \\ g(x) &= 0 \ 1 \ 2 \ 2 \ 2 \ 3 \ 3 \ \dots \end{aligned}$$

Snažte sa, aby Váš program pracoval čo najrýchlejšie (vzhľadom k počtu volaní funkcií), ale radšej program pomalý a dobrý ako program zlý.

## 1521. O prevrate

35

V istej nemenovanej krajine za veľa horami a aspoň tolkými dolami sa odohral štátny prevrat. Starý zlý náčelník Bubutu Sese Kosa bol zvrhnutý a na jeho miesto nastúpil mladý Tekila. Ako to po každom správnom prevrate chodí, nový náčelník nasľuboval zrušiť všetky staré (a teda zlé) zákony a nahradiť ich novými. I vyhlasoval Tekila nové zákony, rušil staré, až ich všetky zrušil. Ale zrušil naozaj všetky pozostatky po starom zlom Bubutuovi? Tekila schytil zbierku zákonov a začal kontrolovať všetky zákony, či pod každým je jeho znak. Zbierka zákonov je však veľmi rozsiahla. Ešte že nedávno vyšla na CD.

ÚLOHA: Na vstupe je jedna stránka zo zbierky nascanovaná vo vysokom rozlíšení – obdĺžnik  $m \times n$  núl a jednotiek. Takisto je daný Tekilov znak – obdĺžnik  $a \times b$ ,  $a \leq m$ ;  $b \leq n$ . Vašou úlohou je napísať program, ktorý zistí, či sa znak nachádza na stránke, t.j. či sa dá znak posunúť tak, aby sa presne zhodoval s nejakou oblasťou stránky.

Príklad:

Vstup:

$m = 5$ ;  $n = 4$ ;  $a = 3$ ;  $b = 3$

Znak:

111

010

010

Stránka zo zbierky:

11001

00111

10010

00010

Výstup:

Áno, na stránke je znak. (Ľavý horný roh má súradnice  $[3, 2]$ ).

## 1522. O komercializácii zlatokopectva

M37

Santo s Bantom sa rozhodli využiť roky strávené kopaním zlata prekvapujúcim spôsobom. Na staré kolená sa rozhodli stať sa sprievodcami po baniach, vykopaných mnohými zlatokopmi na Vyšnej Klondike. Taká baňa sa skladá zo siení, kde sa kedysi kopalo zlato, pospájaných chodbami. Zlatokopi sú leniví chodiť do kopca, preto všetky siene a chodby sú v rovnakej výške. Žiadne dve chodby sa nekrižujú (lebo by dochádzalo k zrážkam vozíkov naložených zlatom).

Niekoľko siení (takzvané vonkajšie siene) je pospájaných chodbami do vonkajšieho okruhu. Každá vonkajšia sieň je spojená s práve dvoma inými vonkajšími sieňami chodbou. Všetky ostatné siene (takzvané vnútorné siene) sú vnútri vonkajšieho okruhu. Do bane sa vchádza cez jedinú sieň na vonkajšom okruhu. Z každej siene vedú práve tri chodby do troch rôznych iných siení (t.j. z každej vonkajšej chodby vedie práve jedna chodba do vnútornej siene). Medzi každými dvoma sieňami v bani sa dá prejsť práve jedným spôsobom tak, že nepoužijeme žiadnu chodbu z vonkajšieho okruhu a každou sieňou prejdeme najviac raz.

Santo s Bantom by chceli pre návštevníkov pripraviť okružnú cestu, ktorou by návštevníci prešli cez každú sieň práve raz. Keďže jedna trasa sa rýchlo okuká, chceli by ich mať pripravených niekoľko. Na to by radi vedeli, koľko takých okružných ciest vlastne existuje. Teraz smutne sedia nad mapou bane a snažia sa ich spočítať.

ÚLOHA: Napíšte program, ktorý načíta  $n$  – počet siení v bani,  $k$  – počet vonkajších siení ( $3 \leq k < n \leq 1000$ ) a zoznam  $3n/2$  chodieb medzi sieňami a zistí počet okružných ciest po bani. Pre jednoduchosť predpokladajte, že siene sú číslované  $1 \dots n$ , siene 1 až  $k$  sú vonkajšie a sieň číslo jedna je vstupná miestnosť. Každá okružná cesta začína aj končí v sieni číslo 1. Dve trasy, ktoré sa líšia len smerom prehladky, považujeme za rôzne.

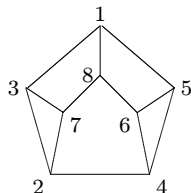
Pre baňu na plánu existuje 6 okružných trás:

1, 5, 4, 6, 8, 7, 2, 3 (a naspäť do 1),

1, 8, 6, 5, 4, 2, 7, 3,

1, 5, 6, 4, 2, 3, 7, 8

plus každá ešte v opačnom smere.



## 1523. O autíčkach

Ako vždy na prednáške, aj dnes Brutus a Frutus nemali čo robiť. Keďže nemali žiaden dobrý časopis, ani knihu, a piškvorky sa im už hrať nechcelo, pustili sa do novej hry. Autíčka sa hrajú na štvorcovom papieri, na ktorom je vyznačená pretekárska dráha. Každý hráč má na začiatku na štartovacom bode svoje autíčko. Cieľom je dopraviť toto

autíčko do cieľového bodu na čo najmenej ťahov. V jednom ťahu sa môže autíčko pohnúť o svoj vektor rýchlosti, ktorý môže hráč pred začiatkom ťahu zmeniť o  $-1, 0$ , alebo  $1$  v oboch smeroch ( $x$  aj  $y$ ). Autíčko nevie prechádzať stenou, t.j. vektor rýchlosti nesmie križovať miesto mimo dráhy. Na začiatku autíčko stojí, v cieľi môže mať akúkoľvek rýchlosť. Po tom, čo Frutus tretíkrát za sebou vyhral, rozhodol sa Brutus požiadať vás o pomoc (aby nemusel Frutovi vybiť ďalší zub).

ÚLOHA: Napíšte program, ktorý pre danú trať vypíše minimálny počet ťahov, za ktoré sa dá prejsť zo štartu do cieľa. Trať je zadávaná nasledovne:  $m, n$  - veľkosť poľa,  $n$  riadkov po  $m$  znakoch samotná trať ( $0$  - trať,  $X$  - mimo trate), a nakoniec súradnice štartu a cieľa.

Príklad:

Vstup:

$m = 14, n = 14$

XXXXXXXXXXXXX

X00000XXXXXXX

X00000XXXXXXX

X00000000XXXXX

X000X0000000XXX

X000X000000000

X0000X00000000

XX000XXXXXXX

XX000XXXXXXX

XX000XXXXXXX

XX000XXXXXXX

XX000XXXXXXX

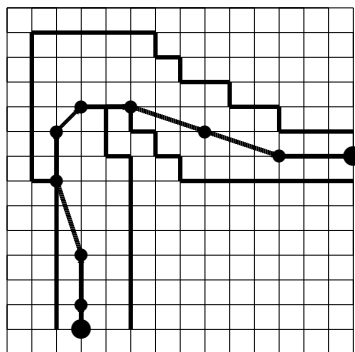
XX000XXXXXXX

XXXXXXXXXXXXX

Štart: (3, 13) Cieľ: (14, 6)

Výstup:

Trať sa dá prejsť na 9 ťahov.



#### 1524. O vrtoch v Legolande

37

Na Legolandskej náhornej plošine (alebo skôr pod ňou) objavili ropu. Majetkuchtívi Lewingovci spravili hneď niekoľko vrtov. Hneď ako zistili, že na vyčerpanie celého ropného poľa im viac netreba, rozhodli sa, že si pozemok ohradia. Legolandská náhorná plošina je celá rozdelená na rovnako veľké štvorcové políčka. Každé políčko má význačný bod – jeho stred. V Legolande cenu pozemku určuje počet význačných bodov vnútri pozemku. Lewingovci by boli radi, keby platili čo najmenej – len za políčka s vrtmi. Stĺpy, medzi ktoré sa natiahne plot, samozrejme stoja vo význačných bodoch nejakých iných políčok.

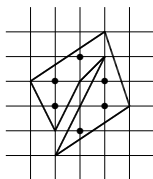
ÚLOHA: Napíšte program, ktorý načíta číslo  $n$  a súradnice  $n$  mrežových bodov a zistí, či existuje taký mnohoúhelník s vrcholmi v mrežových bodoch, ktorý obsahuje vo vnútri všetky mrežové body zo zadanej množiny a žiadne iné mrežové body. Ak existuje, vypíšte zoznam vrcholov ľubovoľného vyhovujúceho – za radom po obvodu, každý zadaný svojimi súradnicami; obvod nesmie prechádzať dvakrát tým istým bodom. Ak neexistuje, vypíšte o tom správu. Mrežové body sú body s celočíselnými súradnicami.

Napríklad pre  $n = 6$  a mrežové body:

(1, 2), (1, 3), (2, 4), (2, 1), (3, 2), (3, 3) sú sú-

radnice hľadaného mnohoúhelníka: (1, 0),

(4, 2), (3, 5), (0, 3), (1, 1), (2, 3), (3, 4).



## 1525. O funkcionálnom programovaní II

[21]

V príklade budeme pracovať s funkcionálnym jazykom opísaným v zadaniach prvého kola. Tento jazyk používal iba prirodzené čísla. Keďže neoddeliteľnú súčasť každého programovacieho jazyka tvoria dátové štruktúry, obohatíme ho ešte o možnosť kódovania dátových štruktúr do prirodzených čísel.

Preddefinované funkcie. Okrem funkcií  $+$  a  $-$  jazyk obsahuje aj ďalšiu preddefinovanú funkciu  $p : \mathcal{N} \times \mathcal{N} \rightarrow \mathcal{N}$  s nasledujúcimi vlastnosťami:

1.  $p(x, y) = p(v, w) \rightarrow x = v \wedge y = w$ , pre každé  $x, y, v, w \in \mathcal{N}$ ,
2.  $0 < p(x, y) \wedge x < p(x, y) \wedge y < p(x, y)$ , pre každé  $x, y \in \mathcal{N}$ ,
3.  $x > 0 \rightarrow (\exists v, w \in \mathcal{N})(x = p(v, w))$ , pre každé  $x \in \mathcal{N}$ .

Pomocou funkcie  $p$  je teda možné každú dvojicu prirodzených čísel jednoznačne zakódovať do jedného prirodzeného čísla a naopak pre každé prirodzené číslo  $x > 0$  existuje jediná dvojica, ktorej je  $x$  kódom. Číslo  $0$  nie je kódom žiadnej dvojice.

Dohoda: výraz  $p(x, y)$  budeme zapisovať ako  $x, y$ , pričom  $u, v, w$  znamená  $u, (v, w)$ , teda zátvorkuje sa doprava. Operátor  $,$  má nižšiu prioritu ako  $+$  a  $-$ , teda  $u, v + w$  je to isté ako  $u, (v + w)$ .

Funkcie spĺňajúce uvedené tri vlastnosti sa nazývajú *párovacie funkcie* a existuje ich nekonečne veľa. Jedna z nich je napríklad Cantorova<sup>17</sup> párovacia funkcia  $c(x, y) = \frac{(x+y) \cdot (x+y+1)}{2} + x + 1$ . Usporiadajme všetky usporiadané dvojice prirodzených čísel  $(x, y)$  podľa súčtu  $x + y$ , pričom dvojice s rovnakým súčtom usporiadame podľa prvého člena. Dostávame tak postupnosť  $(0, 0), (0, 1), (1, 0), (0, 2), (1, 1), (2, 0), (0, 3), \dots$ . Hodnota  $c(x, y)$  udáva poradie dvojice  $(x, y)$  v tejto postupnosti. Ľahko možno overiť, že funkcia  $c$  spĺňa všetky uvedené vlastnosti.

Zoznamy. Zoznam čísel  $a_1, a_2, \dots, a_n$  zakódujeme ako  $a1, a2, \dots, an, 0 = p(a1, p(a2, p(\dots p(an, 0) \dots)))$ . Každé prirodzené číslo je kódom práve jedného zoznamu, pričom  $0$  kóduje prázdny zoznam a číslo  $x = u, v$  kóduje zoznam, ktorého prvým prvkom je  $u$  a  $v$  je kódom zoznamu ostatných prvkov.

Pomocné premenné. V klauzule je možné okrem vstupných premenných používať aj pomocné premenné. V priebehu vyhodnocovania klauzuly sa jednotlivým pomocným premenným priradiť hodnoty. Pomocné premenné, ktoré už majú priradenú hodnotu, nazývame ohodnotené. Ohodnotené premenné je možné použiť v ľubovoľnom výraze v klauzule, rovnako ako vstupné premenné.

Vyhodnocovanie klauzúl. Klauzula sa vyhodnocuje nasledujúcim spôsobom: Najprv sa priradia hodnoty vstupným premenným. Potom sa postupne zľava doprava vyhodnocuje podmienka v pravej časti klauzuly, ktorá má tvar  $A1 \wedge A2 \wedge \dots \wedge An$ .  $Ai$  môže mať jeden z nasledujúcich tvarov:

- $Ai$  má tvar  $s \neq t, s < t, s \leq t, s > t, s \geq t$  a všetky premenné vo výrazoch  $s$  a  $t$  sú ohodnotené. Preto je možné určiť hodnotu oboch výrazov a porovnať ich.
- $Ai$  má tvar  $s = t$ , pričom všetky premenné v  $s$  aj  $t$  sú ohodnotené. Postupuje sa rovnako ako v predchádzajúcom prípade.
- $Ai$  má tvar  $s = t$ , pričom všetky premenné v  $s$  sú ohodnotené a  $t$  obsahuje iba volania párovacej funkcie, neohodnotené pomocné premenné a konštanty. V tom prípade sa pomocným premenným priradia také hodnoty, pri ktorých by platila rovnosť  $s = t$ . Ak také hodnoty neexistujú (napríklad  $0 = u, v$ ), podmienka  $Ai$  sa považuje za nesplnenú. Z vlastností párovacej funkcie vyplýva, že ak také hodnoty existujú, dajú sa hodnoty premenným priradiť jednoznačne.

Ak nebolo možné ohodnotiť vstupné premenné klauzuly, alebo niektorá časť podmienky nie je splnená, preruší sa vyhodnocovanie klauzuly a použije sa iná klauzula funkcie.

<sup>17</sup> Cantor, Georg, 1845–1918, nemecký matematik a zakladateľ teórie množín

Ak je celá podmienka splnená, vyhodnotí sa výraz naľavo od šípky a funkcia nadobudne jeho hodnotu.

Príklad: Funkcia *Length* pre zadaný zoznam vráti ako výsledok jeho dĺžku.

$$\begin{aligned} \text{Length}(x) &= 0 \quad \leftarrow \quad x = 0 \\ \text{Length}(x) &= 1 + \text{Length}(v) \quad \leftarrow \quad x = u, v \end{aligned}$$

Vysvetlime si, ako prebehne vyhodnotenie funkcie *Length* pre vstup  $x = 7, 0, 0$ :

$$\text{Length}(7, 0, 0) \stackrel{1}{=} 1 + \text{Length}(0, 0) \stackrel{2}{=} 1 + (1 + \text{Length}(0)) \stackrel{3}{=} 1 + (1 + 0) = 2$$

Funkcia bola zavolaná s parametrom  $x = 7, 0, 0$ , teda v definícii vyhovuje druhá klauzula, pričom  $u = 7$  a  $v = 0, 0$  (rovnosť 1). Podobné vyhodnotenie prebehlo v rovnosti 2, len  $u = v = 0$ . V rovnosti 3 je parameter funkcie  $x = 0$ , preto bola použitá prvá klauzula. Teraz sú už známe všetky argumenty funkcie  $+$  a preto sa môže vyhodnotiť konečný výsledok. Dĺžka zoznamu  $7, 0, 0$  je teda 2.

ÚLOHA: Vo funkcionálnom jazyku napíšte

1. funkciu *Rev*, ktorá zo vstupného zoznamu  $x$  vypočíta zoznam  $y$ , ktorého prvky budú usporiadané presne v opačnom poradí. Napríklad  $\text{Rev}(1, 3, 2, 4, 7, 0, 0) = 0, 7, 4, 2, 3, 1, 0$ .
2. funkciu *Append*, ktorá zo vstupných zoznamov  $x$  a  $y$  vypočíta zoznam  $z$ , ktorý bude obsahovať postupne všetky prvky zoznamu  $x$  a potom všetky prvky zoznamu  $y$ , v pôvodnom poradí. Napríklad  $\text{Append}((1, 2, 3, 0), 4, 5, 0) = 1, 2, 3, 4, 5, 0$ . Pozor –  $\text{Append}((1, 2, 3, 0), 4, 5, 0) \neq (1, 2, 3), 4, 5, 0$ !
3. funkciu *Order*, ktorá vzostupne usporiada zoznam čísel. Napríklad  $\text{Order}(1, 3, 2, 4, 7, 0, 0) = 0, 1, 2, 3, 4, 7, 0$ .

Snažte sa písať programy pomocou chvostovej rekurzie a tak, aby boli čo najefektívnejšie (vzhľadom k počtu volaní funkcií). Najdôležitejšie však je, aby bol program správny.

### 1531. O zlej kráľovnej

[27]

„Zrkadielko, zrkadielko, povedz že mi, ktorá zo žien najkrajšia je tu na Zemi?“ pýtala sa zrkadielka navoňaná, naondulovaná a našminkovaná kráľovná, ošperkovaná od hlavy po päty.

„Tu na hrade vôkol teba nenájde sa krajšia deva, no za horami, dolami, krajšia Snehuľka medzi trpaslíkmi,“ odpovedalo zrkadielko. A nazlostila sa kráľovná, to zrkadielko ju určite klame. Veď Snehuľku sama zahrúsila. A v hneve kráľovná zrkadielko rozbila. Čoskoro však začala trpko ľutovať svoj čin – tej nehanebnici sa už predsa podarilo uniknúť, čo ak zase...

A rýchlo si nechala doniesť nové zrkadlo, to však bolo voľajaké hlúpe. Radcovia radili, radili a poradili, že ho treba naprogramovať. Pomôžte kráľovnej a naprogramujte zrkadielko.

ÚLOHA: Zrkadielko naprogramujte tak, aby stále prijímalo správy a odpovedalo na otázky (až do skonania sveta).

Na vstup prichádzajú od sudičiek správy o narodení, z márníc správy o smrti jednotlivých žien, správa o skončení roka a otázky, ktorá zo žien je najkrajšia. Správy majú nasledujúci tvar:

**NARODENIE**  $\langle \text{meno} \rangle \langle \text{index} \rangle \langle \text{rozkvet} \rangle$  kde  $\langle \text{meno} \rangle$  je meno novonarodenej,  $\langle \text{index} \rangle$  je počiatkový index krásy (reálne číslo), tento sa každý rok na Silvestra zvýši o 1, až kým nedosiahne vrchol – počas roka, keď žena dosiahne  $\langle \text{rozkvet} \rangle$  rokov, potom sa bude každý rok o 1 znižovať

**EXITUS**  $\langle \text{meno} \rangle$ , kde  $\langle \text{meno} \rangle$  je meno zosnulej

SILVESTER

ZRKADIELKO, ZRKADIELKO POVEDZ ŽE MI, KTORÁ ZO ŽIEN NAJKRAJŠIA JE NA ZEMI.

Ak najkrajšia zo žien je KRÁLOVNÁ, tak zrkadielko odpovie ČO BYS' PREŠLA ŠÍRY SVET, KRAJŠEJ DEVY NIKDE NIET. Ak kráľovná nie je najkrajšia, ale najkrajšia je *(kráska)*, vypíše TU NA HRADE VOKOL TEBA NENÁJDE SA KRAJŠIA DEVA, NO ZA HORAMI, DOLAMI KRAJŠIA JE *(kráska)*. Môžete predpokladať, že stále je práve jedna zo žien najkrajšia.

Ak nie je žiadna žena v zozname (teda ani na Zemi), vypíše VŠETKY ŽENY POMRELI.

Keď správa pre zrkadielko nebude mať žiaden z predchádzajúcich tvarov, zrkadielko odpovie TVOJA REČ MI ZNIE AKO BZUKOT MÚCH.

**1532. O eskimákoch a iglu**

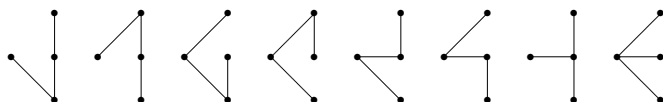
M30

Ako iste všetci vieme, aj eskimáci majú svoje osady, kde si pokojne žijú, stavajú iglu, chytajú ryby a celkovo sa majú fajn. Ešte donedávna si eskimáci z jednej osady budovali všetky iglu do radu a cestička spájala len susedné domky. Bolo to vyhovujúce, lebo sa dalo z každého iglu dostať do každého iného iglu a keď si niekto postavil nový príbytok na koniec zástavby, stačilo vyšľapať len krátku vzdialenosť k nemu.

Ale teraz, po nástupe Veľkého Eskimáka, sa mnohé zmenilo. Veľký Eskimák si nechal postaviť Veľké Iglu v každej osade a vyšľapať k sebe cestu od všetkých ostatných príbytkov z osady. Keď však zistil, že vyšľapané cestičky rýchlo znovu zafúka a ich stála údržba je aj pre neho pomerne drahá, rozhodol sa presne stanoviť, ktoré cesty sa budú v ktorej osade udržiavať. V každej osade z už vyšľapaných ciest vybral také, aby sa dalo z každého iglu dostať do každého, ale aby nebola žiadna cesta nadbytočná (teda aby sa nedalo z jedného iglu do druhého prejsť dvoma rôznymi spôsobmi). Navyše sa snažil svoj výber ciest urobiť tak, aby pohľad z lietadla na dve rôzne osady vyzeral rôzne. Začala ho ale trápiť myšlienka, či sa to vôbec dá...

ÚLOHA: Napište program, ktorý pre zadaný počet iglu  $n$  v osade (napriek názoru Veľkého Eskimáka, že jeho Iglu je za tri, budeme ho počítať iba raz) vypočíta, koľkými spôsobmi sa dajú navrhnuť udržiavané cesty.

Napríklad pre  $n = 4$  je to 8 spôsobov.

**1533. O veľkom smäde**

37

Veľká piesočná krajina je pomerne riedko obývaná. Všetky osady domorodcov ležia pri Malej piesočnej rieke – jedinej rieke v krajine. Časté pieskové búrky veľmi sťažujú pestovanie rastliny butu, známej svojimi pozoruhodne veľkými plodmi, z ktorých miestni obyvatelia tradične pripravujú opojný nápoj lavórovicu. Počet plodov butu, ktoré prežijú do najbližšieho zberu úrody, sa stáva predmetom ostrých sporov. To využívajú rôzni pokútni veľtci a predpovedači budúcnosti. Za menší či väčší poplatok sú ochotní nechať sa ponúknuť lavórovickou a potom v tranze vykrikovať veštby typu „Prvé štyri osady poniže Jamy v piesku dole prúdom Piesočnej rieky budú mať úrodu minimálne osemdesiattri plodov butu“ alebo „Čo by si splavil Piesočnú riekou od Piesočného vrchu po Kopu piesku, viac než päťdesiatsedem plodov butu nenájdeš“.

Jeden takýto veľtec bol domorodým obyvateľom veľmi podozrivý. Na to, aby ho mohli natrieť kuracou krvou, vyváľal v perí a zakopať po krk do piesku, musia mu najprv dokázať, že aspoň raz klamal. I spísali si všetky jeho veštby na papier a teraz nad nimi sedia a nevedia, čo ďalej.



ÚLOHA: Očíslujme osady dole prúdom Piesočnej rieky  $1 \dots n$ . Všetky veštby sú tvaru „v osadách  $k, k+1, \dots, l$  sa spolu urodí aspoň  $m$  plodov butu“ alebo „v osadách  $k, k+1, \dots, l$  sa spolu urodí najviac  $m$  plodov butu“. Zodpovedajúce vstupné údaje majú tvar  $k \ l \text{ aspon } m$  a  $k \ l \text{ najviac } m$ . Napište program, ktorý načíta pre každú veštbu trojicu čísel  $k, l, m$  ( $k \leq l$ ) a typ veštby a zistí, či je možné, aby sa všetky splnili.

Príklad:

Vstup:

3 veštby:

1 2 aspon 26

3 3 aspon 15

1 3 najviac 40

Výstup:

Všetky veštby sa nemôžu splniť.

### 1534. O vianočných darčekom

[30]

Frutus a Brutus sa i tento rok zišli cez vianočné sviatky doma. Vianočná pohoda panovala všade navôkol, pod stromček dostali obaja navlas rovnaké darčeky, aby sa nemohli hádať a handrkovať o to, koho darček je hodnotnejší. Ale keď to všetko porozbalovali, nastal problém, ktorý by nikto neočakával. Obom zostali po darčekom identické kopy škatúl. Vtedy prišlo Frutovi na um, ako dokázať, že je šikovnejší. Začal z týchto škatúl stavať vežu. Brutus sa nedal a onedlho mal aj on škatule poukladané na sebe. Pravidlá boli jasné: poukladať jednotlivé škatule na seba tak, aby veža bola najvyššia možná. Iba tak sa dalo vyhrať. Vždy, keď sa jednému podarilo získať určitú výšku, druhý prišiel s konkurenčným rozostavením škatúl, tvoriacim vyššiu vežu. A pravdepodobne kombinujú až dodnes, lebo nevedia zistiť, aká najvyššia veža sa vôbec dá postaviť.

ÚLOHA: Napište program, ktorý pre zadané  $n$  – počet škatúl a rozmery  $-(x_1, y_1, z_1), \dots, (x_n, y_n, z_n)$  vypíše najväčšiu možnú výšku veže, ktorá sa dá z daných škatúl postaviť. Škatule sa dajú ľubovoľne otáčať. Ak je škatuľa B položená na škatuli A, pričom horná stena A má rozmery  $a_1 \times a_2$  ( $a_1 \leq a_2$ ) a dolná stena B má rozmery  $b_1 \times b_2$  ( $b_1 \leq b_2$ ), potom musí platiť  $a_1 \leq b_1$  a  $a_2 \leq b_2$ .

Napríklad pre  $n = 3$  a škatule rozmerov  $(2, 2, 2)$ ,  $(3, 2, 1)$ ,  $(1, 2, 2)$  je maximálna výška 7.

### 1535. O funkcionálnom programovaní III

[22]

V príklade budeme pracovať s funkcionálnym jazykom opísaným v zadaniach prvého a druhého kola. Tam sme sa naučili pracovať s klauzami nad prirodzenými číslami a kódovať do prirodzených čísel dátovú štruktúru zoznam. Veľmi prirodzeným spôsobom možno pomocou zoznamov reprezentovať štruktúru stromu. V tomto príklade budeme uvažovať tzv. *binárne stromy*. Prázdny binárny strom neobsahuje žiaden vrchol. Neprázdny binárny strom obsahuje práve jeden vrchol, do ktorého nevchádza žiadna hrana – *koreň*. Z koreňa vychádzajú dve hrany, ktoré ho spájajú s ľavým a pravým podstromom (v prípade, že tento podstrom nie je prázdny, hrana vedie do jeho koreňa). V každom vrchole nášho binárneho stromu je uložené prirodzené číslo.

Binárny strom. Prázdny binárny strom reprezentujeme prázdny zoznamom  $\emptyset$ . Neprázdny binárny strom  $T$  s hodnotou  $k$  v koreni, s ľavým podstromom  $T_L$  a pravým podstromom  $T_P$  reprezentujeme ako zoznam  $zT = (k, zTL, zTP)$ , kde  $zTL$  je zoznam reprezentujúci strom  $T_L$  a  $zTP$  je zoznam reprezentujúci  $T_P$ .

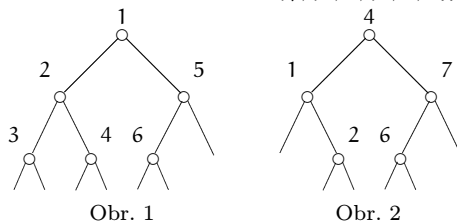
Hĺbka stromu. Hĺbku stromu definujeme vo funkcionálnom jazyku pomocou funkcie *Depth*.

$$\text{Depth}(zT) = 0 \quad \leftarrow \quad zT = \emptyset$$

$$\text{Depth}(zT) = \text{Depth}(zTL) + 1 \quad \leftarrow \quad zT = k, zTL, zTP \wedge \text{Depth}(zTL) > \text{Depth}(zTP)$$

$$\text{Depth}(zT) = \text{Depth}(zTP) + 1 \quad \leftarrow \quad zT = k, zTL, zTP \wedge \text{Depth}(zTL) \leq \text{Depth}(zTP)$$

Príklad. Strom na obrázku 1 je reprezentovaný zoznamom  $(1, (2, (3, 0, 0), (4, 0, 0)), (5, (6, 0, 0), 0))$ , strom na obrázku 2 zoznamom  $(4, (1, 0, (2, 0, 0)), (7, (6, 0, 0), 0))$ .



Obr. 1

Obr. 2

Vyhľadávacie stromy. Binárny strom sa nazýva *vyhľadávací*, ak pre každý jeho podstrom  $T$  s koreňom vo vrchole  $v$  platí, že hodnota vo vrchole  $v$  je väčšia ako najväčšia hodnota z ľavého podstromu stromu  $T$  a zároveň je menšia ako najmenšia hodnota z pravého podstromu stromu  $T$ .

Napríklad strom na obr. 1 nie je vyhľadávací, pretože každý ľavý syn má hodnotu väčšiu ako otec. Naopak strom na obr. 2 vyhľadávací je.

Vo vyhľadávacích stromoch možno ľahko zistiť, či je v ňom zaradený daný prvok – na základe hodnoty koreňa možno totiž jednoznačne určiť, v ktorom podstrome sa hľadaný prvok môže nachádzať. Istým zlepšením sú *vyvážené vyhľadávacie stromy* – u nich podstromy každého vrchola majú približne rovnakú veľkosť, a teda každým porovnaním vylúčime polovicu možností.

*AVL-stromy.* Binárny vyhľadávací strom nazveme AVL-stromom, ak sa hĺbky ľavého a pravého podstromu každého vrchola líšia najviac o 1.

Strom na obr. 2 je AVL-stromom, pretože sa hĺbky oboch podstromov pre každý vrchol okrem vrcholov s číslami 1 a 7 rovnajú. Ľavý podstrom vrchola s číslom 1 má hĺbku 0 a pravý 1, teda ich rozdiel je rovný práve jednej. Podobne rozdiel hĺbok ľavého a pravého podstromu vrchola s číslom 7 je jedna.

ÚLOHA: Vo funkcionálnom jazyku napíšte

- funkciu  $Member(T, x)$ , ktorá vráti 0, ak sa v AVL-strome  $T$  nenachádza vrchol s hodnotou  $x$ , alebo 1, ak sa tam vrchol s uvedenou hodnotou nachádza.
- funkciu  $Insert(T, x)$ , ktorá zo vstupného AVL-stromu  $T$  a prvku  $x$  vypočíta AVL-strom  $T'$ , ktorý bude obsahovať vrcholy s rovnakými číslami ako strom  $T$  a navyše vrchol s číslom  $x$ . Môžete predpokladať, že strom  $T$  vrchol s číslom  $x$  neobsahuje. Napríklad  $Insert((1, 0, (2, 0, 0)), 3) = 2, (1, 0, 0), (3, 0, 0)$ .
- funkciu  $Delete(T, x)$ , ktorá zo vstupného AVL-stromu  $T$  a prvku  $x$  vypočíta vyhľadávací AVL-strom  $T'$ , ktorý bude obsahovať vrcholy s rovnakými číslami ako strom  $T$  okrem vrchola s číslom  $x$ . Môžete predpokladať, že strom  $T$  vrchol s číslom  $x$  obsahuje. Napríklad  $Delete((4, (1, 0, (2, 0, 0)), (5, 0, 0)), 5) = 2, (1, 0, 0), (4, 0, 0)$ .

Snažte sa písať svoje programy tak, aby boli čo najefektívnejšie (vzhľadom k počtu volaní funkcií). Najdôležitejšie však je, aby bol program správny.

#### 1541. O životnom prostredí

30

Bolo to už strašne dávno čo si mohol človek vyjsť voľne do prírody, kochať sa jej krásou, dýchať čerstvý a svieži vzduch, piť vodu z potoka... Ľudská hlúposť zničila všetkú túto krásu a dnešný človek sa dennodenne stretáva s následkami nepredvídateľného konania našich otcov a starých otcov. Aj u nás je čoraz väčší problém napríklad s vodou, pretože voda v potokoch je väčšinou znečistená kadejakými chemikáliami, a dnešné fabriky a iný znečisťovatelia badateľne znižujú aj kvalitu podzemnej vody.

Minulý rok sa preto Ministerstvo životného prostredia rozhodlo vytvoriť centrálnu databanku všetkých znečisťovateľov. Zhromaždili si o každom z nich tie najpodstatnej-

šie údaje ako sú poloha a polomer znečisťovaného územia. Súčasná kapacita dostupných vodných zdrojov nestačí, preto je potrebné kopať nové studne. Pri kopaní studne treba vedieť odhadnúť približné znečistenie. Ministerstvo životného prostredia sa rozhodlo zdarma poskytovať informáciu o tom, koľko znečisťovateľov prispieva k znečisteniu toho ktorého miesta. Žiadosti o túto novú službu je veľa, preto ministerstvo potrebuje software, ktorý by ich dokázal čo najefektívnejšie spracovať.

ÚLOHA: Napíšte program, ktorý na vstupe dostane počet znečisťovateľov  $n$ , o každom znečisťovateľovi dostane tri čísla  $x$ ,  $y$  a  $r$ , ktoré určujú, že tento znečisťovateľ znečisťuje kruh so stredom  $(x, y)$  a polomerom  $r$ . Ďalej je na vstupe počet otázok  $p$  a  $p$  súradníc miest  $(x, y)$ . Váš program by mal pre každé takéto mesto  $v$  čo najrýchlejšom čase určiť počet znečisťovateľov, ktorý miesto znečisťujú.

Najdôležitejšie je, aby váš program vedel vyhodnotiť znečistenie nejakého miesta  $v$  čo najkratšom čase, aj na úkor použitej pamäti. Môžete predpokladať, že číslo  $p$  je ďaleko väčšie ako  $n$ .

Príklad:

Vstup:

$n = 2$

$(0, 0)$ ,  $r = 10$

$(3, 3)$ ,  $r = 1$

$p = 4$

$(1, 1)$ ,  $(10, 0)$ ,  $(11, 0)$ ,  $(2.5, 2.5)$

Výstup:

1 znečisťovateľ

1 znečisťovateľ

0 znečisťovateľov

2 znečisťovateľia

#### 1542. O malom Tadeášovi

Nedávno mal malý Tadeáš prvé narodeniny. Keď sa to verní rodinní priatelia Frutus a Brutus dozvedeli, neváhali a okamžite zabočili do najbližšieho hračkárstva, aby mu kúpili nejaké prekvapenie. Výsledkom ich úporného snaženia vystihnúť Tadeáškov vkus bolo  $n^3$  kociek, asi polovica s obrázkami zvieratiek (to sa snažil Frutus) a zvyšok s autíčkami (to zase Brutus). Od predavačky vymámili ešte zvyšok baliaceho papiera, požičali si nožnice, sadli na zem a snažili sa čo najrýchlejšie kocky zabaliť. Dohodli sa, že bude najlepšie, keď bude balík vyzeráť ako jedna veľikánska kocka. Frutus poukladal kocky na papier, Brutus okolo nich ten papier obstrihal, nožnice vrátili a teraz dumajú, či vystrihnutým útvarom možno kocky obaliť.

ÚLOHA: Pomôžte Frutovi a Brutovi – napíšte im program, ktorý ich problém vyrieši. Pre dané  $n$ ,  $k$ ,  $l$  a maticu núl a jednotiek s rozmermi  $k \times l$  (jednotka znamená baliaci papier, nula nič) vypíšte, či je možné plášťom zadaným maticou obaliť kocku s hranami dĺžky  $n$  (malé kocky majú hrany dĺžky 1). Môžete predpokladať, že matica má práve  $6n^2$  jednotiek a oblasť určená jednotkami je súvislá. Zadaný plášť nemožno nastrihávať.

Príklady:

Vstup:

$n = 2$ ,  $k = 5$ ,  $l = 9$

000011000

011010000

011111011

111111110

101010100

Výstup:

Áno, týmto plášťom sa dajú kocky obaliť.

Vstup:

$n = 2$ ,  $k = 4$ ,  $l = 7$

1111100

1111111

1111111

1111100

Výstup:

Nie, týmto plášťom sa nedajú kocky obaliť.

## 1543. O prekliatom meste

35

„Utekajte, utekajte vo všetky strany sveta. Bo kto v tomto meste ostane, svrab, lepra a zimnica ho postretnú a po celom tele mu navrú pluzgiere veľké ako hrach. Celý sa bude zvíjať v kĺkoch a do dvoch dní ochrnie, takže nakoniec na slnečnej pálave zhynie. Cesta, ktorou pôjdete naskutku bodliacím, ostružinou a jedovatými kaktusmi zarastie, aby žiaden smrteľník už na toto miesto nikdy nevkočil.“

Toto a ešte mnoho horších vecí veštila čarodejnica Nica každému, kto sa práve vtedy nachádzal v Čiernom meste, kde ona väznila princeznú Maju. A rytieri, hoci medzi nimi boli aj udatní bojovníci, sa všetci rozbehli po uliciach mesta, aby z neho ušli. A za každým bojovníkom cesta, po ktorej prešiel, zarastala všetkou tou pichľavou flórou, ako Nica hovorila. Keď sa nakoniec dostali von z mesta, zistili, že ich je ledva polovica. I začali si klásť otázky: „Nemohlo sa nás dostať z mesta viac? Možno, ak by sme boli šli inými cestami, aj ďalší mohli uniknúť hrozným mukám, ktoré Nica prorokovala...“ a upadli do hlbokéj depresie. Jediným liekom na ich depresiu by bolo zistenie, že viacerí uniknúť nemohli.

ÚLOHA: Zistite, koľko najviac rytierov mohlo ujsť z mesta. Vieme, že Čierne mesto má tvar obdĺžnika a sú v ňom dva typy ulíc – severojužné a západovýchodné. Pretínajú sa v križovatkách. Severojužné ulice značíme zo západu na východ celými číslami  $1, \dots, m$ , západovýchodné zo severu na juh číslami  $1, \dots, n$ . Na vstupe je zadané  $m, n$ , počet rytierov  $p$  a  $p$  usporiadaných dvojíc  $(x_1, y_1), \dots, (x_p, y_p)$ , ktoré predstavujú počiatočné polohy rytierov ( $x_i$  je číslo severojužnej a  $y_i$  číslo príslušnej západovýchodnej ulice). Rytieri môžu utekať iba po uliciach a na každej križovatke môžu zmeniť smer. Ak však nejaký rytier vkročil na nejakú križovátku, táto zarastie nepreniknuteľnou húštinou, takže žiaden iný (a ani ten istý) rytier na ňu už viac nemôže prísť (predpokladajte, že dvaja rytieri nikdy nevojdú na žiadnu križovátku súčasne). Keď rytier dorazí na niektorú z okrajových križovatiek, podarilo sa mu uniknúť z mesta.

Príklad:

Vstup:

 $m = 3, n = 3, p = 3$ Rytieri:  $(1, 1), (2, 2), (3, 2)$ 

Výstup:

Môžu uniknúť 3 rytieri.

Vstup:

 $m = 5, n = 5, p = 5$ Rytieri:  $(2, 3), (3, 2), (3, 3), (3, 4), (4, 3)$ 

Výstup:

Môžu uniknúť 4 rytieri.

## 1544. O Rasťových otlakoch

35

Na okraji horskej lúky sedí na snehu Rasto a plače. Jeho nezbedný kamarát Dano mu pred túrou nasypal do topánky kamienky. Rasťovi od nich na päte vznikli parádne otlaky, po pár kilometroch už chudák od bolesti hádam ani nevedel, ako sa volá. So smutným výrazom si sadol, vyzul topánku, vysypal kamienky a z batohu vytiahol prenosnú lekárničku. Chvilu sa snažil zalepiť vzniknuté otlaky, no čoskoro to vzdal a teraz už iba tíško vzlyká. Pomôžte mu, kým neprechladne. Je to otázka (Danovho) života a smrti.

Rasťova päta sa dá predstaviť ako matica  $a \times b$  núl a jednotiek, pričom nuly znamenajú zdravú pokožku, jednotky otlak. Každý otlak treba prelepiť náplastou  $3 \times 1$ , pričom otlak musí zakrývať strednú, mäkkú časť náplasti. Náplasť sa dá nalepiť buď zvislo, alebo vodorovne, to znamená, že náplasť na otlaku so súradnicami  $(x, y)$  pokryje buď políčka  $(x-1, y), (x, y)$  a  $(x+1, y)$  alebo políčka  $(x, y-1), (x, y)$  a  $(x, y+1)$ . Žiadne dve náplasti sa nesmú prekryvať (pokryvať to isté políčko), pretože príliš hrubá vrstva náplastí by mohla úbohého Rasťa tlačiť.

ÚLOHA: Napíšte program, ktorý načíta  $n$  – počet Rasťových otlakov, súradnice jednotlivých otlakov a zistí, či je možné prelepiť všetky Rasťove otlaky neprekývajúcimi sa

náplastami. V prípade, že to možné je, vypíšte aj jeden možný spôsob prelepenia, ak to nie je možné, vypíšte správu Chudák Rastó.

Príklad:

Vstup:

Počet otlakov: 3

Súradnice otlakov:

(1,0), (2,1), (2,3)

Vstup:

Počet otlakov: 5

Súradnice otlakov:

(3,2), (4,1), (5,2), (2,3), (3,4)

Výstup:

Otlak (1,0) prelepiť zvislo

Otlak (2,1) prelepiť zvislo

Otlak (2,3) prelepiť vodorovne

Výstup:

Chudák Rastó

#### 1545. O funkcionálnom programovaní IV

Opäť sme tu s našim funkcionálnym jazykom, ktorý už tak dôverne poznáte z prvých troch kôl. Tentokrát si zadefinujeme dátovú štruktúru reprezentujúcu jednoduché aritmetické výrazy. Budeme uvažovať výrazy, ktoré obsahujú konštanty (z oboru prirodzených čísel), premenné  $x_0, x_1, x_2, \dots$  a operácie sčítania, odčítania a násobenia. Formálne výraz môžeme popísať nasledujúcou definíciou:

1. Ak  $k \in \mathcal{N}$ , tak  $k$  je výraz (konštanta).
2. Ak  $k \in \mathcal{N}$ , tak  $x_k$  je výraz (premenná).
3. Ak  $t_1$  a  $t_2$  sú výrazy, tak  $(t_1 + t_2)$  je výraz.
4. Ak  $t_1$  a  $t_2$  sú výrazy, tak  $(t_1 - t_2)$  je výraz.
5. Ak  $t_1$  a  $t_2$  sú výrazy, tak  $(t_1 \cdot t_2)$  je výraz.
6. Nič iné nie je výraz.

Ukážeme si teraz jeden z mnohých možných spôsobov, ako pomocou párovacej funkcie kódovať výrazy do prirodzených čísel, aby sme s nimi mohli pracovať v našom funkcionálnom jazyku.

1. Kódom konštanty  $k$  je číslo  $1, k$ .
2. Kódom premennej  $x_k$  je číslo  $2, k$ .
3. Nech kódom výrazu  $t_1$  je číslo  $u_1$  a kódom výrazu  $t_2$  je číslo  $u_2$ . Potom kódom výrazu  $(t_1 + t_2)$  je číslo  $3, u, v$ .
4. Nech kódom výrazu  $t_1$  je číslo  $u_1$  a kódom výrazu  $t_2$  je číslo  $u_2$ . Potom kódom výrazu  $(t_1 - t_2)$  je číslo  $4, u, v$ .
5. Nech kódom výrazu  $t_1$  je číslo  $u_1$  a kódom výrazu  $t_2$  je číslo  $u_2$ . Potom kódom výrazu  $(t_1 \cdot t_2)$  je číslo  $5, u, v$ .

Príklad: Kódom výrazu  $((x_1 + x_3) \cdot 4)$  je číslo  $5, (3, (2, 1), (2, 2)), (1, 4)$ .

Ekvivalencia výrazov. Ak všetkým premenným nachádzajúcim sa vo výraze  $t$  priradíme nejaké hodnoty, môžeme vypočítať hodnotu výrazu  $t$ . Dva výrazy  $t_1$  a  $t_2$  nazveme ekvivalentné, ak pre ľubovoľné ohodnotenie premenných budú ich hodnoty rovnaké.

Príklad: Pre ohodnotenie premenných  $x_0 = 2, x_2 = 1, x_3 = 3$  bude hodnotou výrazu  $(x_0 + x_3)$  číslo 5 a hodnotou výrazu  $(x_0 + x_2)$  bude číslo 3. Tieto výrazy teda nie sú ekvivalentné. Výrazy  $((x_0 + x_1) - (0 \cdot x_2))$  a  $(x_1 + x_0)$  ekvivalentné sú.

ÚLOHA: Vo funkcionálnom jazyku napíšte funkciu *Ekviv*( $x1, x2$ ), ktorá dostane dva výrazy zakódované do prirodzených čísel a zistí, či sú tieto dva výrazy ekvivalentné. Funkcia vráti číslo 1, ak výrazy sú ekvivalentné, inak vráti číslo 0.

#### z1511. Zorov znak

17

Zoro pomstiteľ dával veľmi charakteristickým spôsobom najavo svoju prítomnosť. Dnes by sme povedali, že si potrpí na image. Po každom jeho (vý)čine bolo podľa písmena **Z**, načrtnutého tromi rýchlymi ťahmi špičkou kordu na stenu alebo iný podklad (tváre jeho nepriateľov nevynímajúc) vidieť, s kým má človek tú česť. Historický ústav po veľkom

úsilí nazhromaždil množstvo fotografií stien budov, v ktorých predpokladali, že by sa Zoro mohol vyskytovať. Fotografie je však strašne veľa, ich prezeranie by trvalo strašne dlho. Pomôžte pracovníkom Historického ústavu a napíšte program, ktorý zistí, koľko znakov **Z** je na fotografii.

ÚLOHA: Fotografia je obdĺžnikové pole ( $m \times n$ ) núl a jednotiek, **Z** sa skladá z dvoch vodorovných a jednej šikmej úsečky, zvierajúcej s oboma osami uhol  $45^\circ$ . Každá z úsečiek je dĺžky aspoň 3. Všetky tri úsečky môžu byť ľubovoľne dlhé, avšak šikmá úsečka musí pretínať obe vodorovné. Predpokladajte, že žiadne **Z** nesusedí so žiadnym iným tmavým bodom. Napíšte program, ktorý zistí, koľko takýchto písmen **Z** sa nachádza v tomto poli.

Príklad:

Vstup:

```
000000000001000000010
11111111111100000100
000000000100011111110
111001001001000010000
010000010001000100000
000111111100011111000
```

Výstup:

Na stene sú dva znaky **Z**.

### z1512. Zeofina spomína

Korytnačka Zeofina to nemala ľahké. Jej sestra Žofka sa stala slávnou pre svoje matematické vedomosti. Zeofinu však ešte v detstve uniesli z rodných Galapág vedci, aby preskúmali jej zvláštne schopnosti. Po strastiplných rokoch sa jej konečne podarilo utiecť z laboratória v Káhire. Utekala, utekala (ako to len korytnačky dokážu) a ani nevedela ako sa dostala na púšť, kde ju zastihla piesočná búrka.

Keď sa búrka utišila, uvidela okolo seba holú pláň. Urobila tri kroky a zistila, že za sebou zanecháva v piesku stopu. Takto si začala do piesku kresliť obrázky. I začnelo sa jej za domovom, a tak sa rozhodla, že nakreslí stromy, aké rastú na Galapágach.

Stromy na Galapágach predovšetkým nemajú listy a sú súmerné podľa osi kmeňa. Pre každý druh je špecifická dĺžka kmeňa  $d$  a uhol rozvetvenia konárov  $u$ . Čím je strom starší, tým je košatejší. Jednoročný strom tvorí vlastne iba jeho kmeň. Ak má strom  $n$  rokov ( $n > 1$ ), tak na konci jeho kmeňa vybiehajú dva konáre. Jeden vľavo pod uhlom  $u/2$  a druhý vpravo pod uhlom  $u/2$ . Každý z týchto konárov sa môže ďalej košatiť a vyzerajú presne ako stromy s vekom  $n - 1$ , polovičnou dĺžkou kmeňa a rovnakým uhlom rozvetvenia konárov.

ÚLOHA: Napíšte program, ktorý povie korytnačke Zeofine, ako má nakresliť strom. Váš program načíta kladné celé čísla  $n$ ,  $u$ ,  $d$  a napíše sériu pokynov pre korytnačku Zeofinu. Korytnačka vie reagovať na tieto pokyny:

**Dopredu**  $a$  – posunie sa v smere svojho natočenia o dĺžku  $a$ ,

**Vľavo**  $u$  – otočí sa o  $u$  stupňov vľavo,

**Vpravo**  $u$  – otočí sa o  $u$  stupňov vpravo.

Korytnačka môže pri kreslení cez jednu čiaru prejsť aj viackrát.

Na ladenie svojho programu môžete napríklad v Turbo Pascale použiť knižnicu *Graph3*, konkrétne jej funkcie *GraphMode*, *Forwd*, *TurnLeft* a *TurnRight*.

### z1513. Z Písmenkovej ulice

Písmenková ulica má veľmi zvláštnych obyvateľov. Hlavne všetkým veľmi záleží na tom (ako to už býva), čo o ňom povedia susedia. Napríklad keď si človek večer zasvieti, ale vonku nie je ešte taká tma, susedia si budú o ňom na druhý deň povrávať: „Všimnite si, on vôbec nevie šetriť elektrinou!“ No musíte uznať, že to nie je veľmi príjemné. Ale ťažko povedať, kedy si už možno bez obáv zasvietiť. Preto večer každý nenápadne pozoruje, či už niekto zo susedov (teda niekto z ľudí bývajúcich nad, pod alebo vedľa) nezasvietil. Ak

áno, potom už môže aj on. Našťastie vždy sa nájde niekto, kto to už psychicky nevydrží a napriek vedomiu, že ho na druhý deň bude ohovárať celé susedstvo, zasvieti. Potom to už postupuje jedna radosť, až nakoniec svietia všetky okná bytov, kde je niekto doma (môžete predpokladať, že ak sa na začiatku rozsvieti jediné okno, na konci budú všetky okná, kde je niekto doma rozsvietené).

ÚLOHA: Na vstupe je daný počet poschodí a počet okien na jednom poschodí domu na Písmenkovej ulici. Ďalej je daný počet okien, kde nikto nie je doma a zoznam týchto okien. Každé okno je dané dvojicou  $a, b$ , kde  $a$  je poschodie a  $b$  číslo okna na poschodí zľava. Ďalej je dané okno, kde to nevydržali a zasvietili. Odsimulujte na obrazovke jeden zo spôsobov, ako sa môžu rozsvetovať ostatné okná.

#### z1514. Zlo volejbalových majstrovstiev

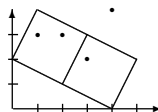
V Nalomenej Trieske sa každoročne pri príležitosti osláv založenia obce konajú Majstrovstvá Nalomenej Triesky vo volejbale. Je to veľká udalosť a každý obyvateľ považuje za česť zúčastniť sa na majstrovstvách ako hráč. Každý rok sa preto vybrali najlepší hráči, rozdelili sa do družstiev a hry sa mohli začať. Až pred pár rokmi sa začalo ozývať čoraz viac protestných hlasov, že vybratí hráči vôbec nie sú najlepší a prečo by sa aj ostatní nemohli hier zúčastniť a ... A vôbec!

Nakoniec starosta Nalomenej Triesky prišiel na šalamúnske riešenie. V deň majstrovstiev prikázal všetkým záujemcom o hru rovnomerne sa rozmiestniť na lúke. Potom sa náhodne určila poloha ihriska. Všetci, ktorí sa nachádzali na ihrisku, sa mohli majstrovstiev zúčastniť ako hráči, ostatní ako diváci. Tento spôsob sa hneď všetkým viac páčil. A tak sa odvtedy hráči určujú takto a v Nalomenej Trieske znova panuje zhoda a pokojná atmosféra.

ÚLOHA: Na vstupe sú dané súradnice rohov ihriska (ihrisko je obdĺžnik), sieť je rovnobežná s kratšou stranou ihriska a delí ho na dva rovnaké obdĺžniky. Ďalej je daný počet ľudí, ktorí sa rozhodli zúčastniť sa hry, a súradnice každého z nich. Zistite, ktorí budú hráčmi prvého družstva, ktorí druhého a ktorí z nich budú diváci.

Človek nachádzajúci sa presne na okraji ihriska, sa tiež stáva hráčom jedného z družstiev. Ten, kto sa nachádza na úrovni siete, je do družstva pridelený náhodne.

Napríklad pre ihrisko  $(0, 2)$ ,  $(1, 4)$ ,  $(5, 2)$ ,  $(4, 0)$  a 4 ľudí so súradnicami  $(1, 2, 3)$ ,  $(4, 4)$ ,  $(2, 3)$ ,  $(3, 2)$  sú hráči prvého družstva:  $(1, 2, 3)$ ,  $(2, 3)$  a hráč druhého družstva:  $(3, 2)$ . Divák je  $(4, 4)$ .



#### z1515. Zachráňte nás!

KRÁL SA ZBLÁZNIL stop ŠETRÍŤ CHCE stop PENIAZE ZRUŠIL stop DAL TLAČÍŤ NOVÉ stop IBA DVA DRUHY stop CHCEME MU UKÁZAŤ ŽE JE TO HLÚPOŠŤ stop ŽE SA NEDÁ VŠETKO ZAPLATIŤ stop ZAKÁZAL VYDÁVAŤ PENIAZE SPÄŤ stop POMÔŽTE NÁM stop RÝCHLO stop

ÚLOHA: Napíšte program, ktorý dostane na vstupe dve prirodzené čísla  $p$  a  $q$  (hodnoty mincí) a vypíše najväčšiu hodnotu, ktorá sa pomocou mincí len týchto dvoch hodnôt a bez vydávania peňazí späť, zaplatiť nedá. Ak taká hodnota neexistuje, program o tom podá správu. Príklad:

Vstup:	Výstup:
$p = 3, q = 5$	7
$p = 1, q = 10$	dá sa všetko
$p = 6, q = 8$	nedá sa nekonečne veľa hodnôt

#### z1521. Závišove slimáky

Záviš dostal ďalší nápad, ako zbohatnúť: založil si farmu na slimáky. Slimáky sa prechádzali po jeho záhradke a utešene rástli. Záviš si všimol, že každý mesiac slimákovi

dorastie jeden závit ulity. Aby si získal zákazníkov, rozhodol sa, že vydá ponukový katalóg s obrázkami rôzne starých slimákov. Keďže nemá fotoaparát, chcel by tlačiť obrázky na počítači.

ÚLOHA: Napíšte program, ktorý načíta vek slimáka (v mesiacoch) a smer natočenia (vľavo alebo vpravo) a vytlačí na tlačiarňu príslušného slimáka. Presný tvar slimáka iste ľahko vypočítate z príkladu.

Tlačiareň vie vypisovať znaky iba postupne zľava doprava a po riadkoch zhora nadol. Ak pracujete v Turbo Pascale, môžete na výstup na tlačiareň použiť unit *Printer* (pri použití tohto unitu vypisujete na tlačiareň príkazmi *write(lst,...)*, resp. *writeln(lst,...)*).

Príklady:

vek: 1 mesiac,

smer: vľavo

H \*

H\*\*

vek: 2 mesiace,

smer: vpravo

\*\*\*\*

\* \*

\* \*\*

\* H

\*\*\*\*\*H

vek: 3 mesiace, smer: vľavo

\*\*\*\*\*

\* \*

\* \*\*\*\* \*

\* \* \* \*

\* \*\* \* \*

\* \* \* \*

\*\*\*\*\* \*

H \*

H\*\*\*\*\*

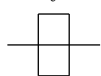
### z1522. Zeofinina nočná mora

Kým Zeofína nakreslila všetky galapágske stromy, na ktoré si spomenula, padol súmrak. A Zeofína nebolo viac treba, neváhala a zaspala ako nemluvňa. Nebol to však spánok, pri ktorom si človek (alebo skôr korytnačka) oddýchne. Iba sa prehadzovala z boka na bok. Zase tá nočná mora. Máva ju od čias, keď ju v Káhire nútili preliezať bludiská. Chceli, aby prešla každou chodbičkou bludiska, ale žiadnou nesmela ísť dvakrát. V jej sne najprv celé bludisko pozostávalo iba z jednej chodbičky. Už-už sa chystala, že ňou prejde, keď sa zrazu chodbička po oboch stranách vyliačila a okrem pôvodnej tu boli aj dve bočné chodbičky. Vchod do nich a východ z nich delili pôvodnú chodbičku na tretiny. Každá z týchto nových chodbičiek sa skladala z troch rovnako dlhých úsekov (s dĺžkou jednej tretiny pôvodnej chodbičky), ktoré zvierali pravý uhol. Kým si však Zeofína rozmyslela, ako pôjde teraz, opäť sa každý priamy úsek (medzi dvoma križovatkami, zákrutami, vchodom, východom) po oboch stranách vyliačil a takto sa to opakovalo, kým sa Zeofína prepotená nezobudila. Zeofína by sa veľmi uľavilo, ak by vedela, ako treba bludisko z jej nočnej mory vlastne prejsť.

Bludisko stupňa 0 má iba jednu priamu chodbu (možno ho teda nakresliť ako úsečku). Bludisko stupňa  $n + 1$  vznikne z bludiska stupňa  $n$  tak, že cez prostrednú tretinu každého priameho úseku dĺžky  $l$  preložíme obdĺžnik so stranami  $\frac{2}{3}l$  (tie delia úsek na tretiny a zároveň ich úsek delí na polovice) a  $\frac{1}{3}l$ . Dĺžka najkratšieho priameho úseku chodby je  $d$ .

ÚLOHA: Napíšte program, ktorý načíta čísla  $n$  a  $d$  a poradí korytnačke Zeofíne, ako má prejsť bludiskom stupňa  $n$  tak, aby prešla každou chodbičkou práve raz. Ako výstup vypíšete sériu pokynov pre korytnačku Zeofínu. Pokyny, na ktoré vie korytnačka reagovať sú uvedené v príklade 1515z.

Príklady bludísk:



$n = 1$



$n = 2$

### z1523. Zanzibarský slon

V ďalekom Zanzibare sa genetickým inžinierom podaril malý zázrak, vypestovali špe-



ciálny druh slona, ktorý má oproti klasickému slonovi veľkú zvláštnosť, nielen jeho kly, ale všetky jeho kosti sú zo vzácnnej slonoviny. Keďže Zanzibar je chudobný štát, rád by na svojom unikátnom slonovi zarobil. Rada starších sa teda rozhodla, že predá niektoré kosti svojho slona a takto získané peniaze venuje na rozvoj kultúry. Problém je však v tom, že nechcú svojmu slonovi ublížiť.

Zanzibarský vedci podrobne preskúmali slona a prišli na to, že ich slon má v tele takzvané významné uzly a jeho kosti spájajú vždy dva z týchto uzlov. Ďalej zistili, že keď slonovi odoberú niektoré kosti, slon ostane zdravý, ak bude jeho kostra naďalej súvislá, čo znamená, že každé dva významné uzly budú nejakým počtom kostí (aj cez iné uzly) navzájom prepojené. Teraz už ostáva zistiť ktoré kosti treba odobrať, aby Zanzibar svojmu slonovi neublížil a pritom kosti, ktoré po operácii slonovi ostanú mali čo najmenšiu cenu.

ÚLOHA: Napište program, ktorý načíta popis kostry slona a vypíše popis kostry po operácii, ktorá slonovi neublíži (kostra ostane súvislá) a zároveň kosti, ktoré v slonovi ostanú, budú mať najmenšiu možnú celkovú cenu. Popis kostry (aj pri vstupe aj pri výstupe) začína riadkom obsahujúcim počet významných uzlov slona  $n$  a počet kostí v kostre  $M$ . Potom nasleduje  $M$  riadkov, pričom každý z nich obsahuje informáciu o jednej slonej kosti a to tri čísla  $a$ ,  $b$  a  $c$ , ktoré hovoria, že kosť spája významné uzly  $a$  a  $b$  a jej cena na čiernom trhu je  $c$ .

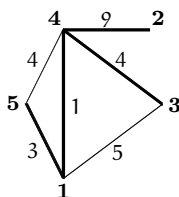
Príklad:

Vstup:

```
5 6
1 3 5
5 1 3
2 4 9
4 3 4
1 4 1
4 5 4
```

Výstup:

```
5 4
1 5 3
2 4 9
4 3 4
1 4 1
```



#### z1524. Zvedavý Jonatán

Keď sa Jonatán ráno zobudil, ledva-ledva sa zmohol na cestu do kuchyne. Unavene sledoval mamu, ako mu pripravuje raňajky. Najskôr položila na stôl chlieb, potom nakrájala syr, uhorky, papriku a iné dobroty na tenké plátky a začala to ukladať na Jonatánov krajec. Potom Jonatán zobral vidličku a bez rozmyslu na ňu napichol chlieb. Ale beda! Na vidličke zostalo len trochu zeleniny a zo dva plátky syra, chlieb aj s ostatnou oblohou zostal na stole. Taká nepríjemnosť hneď zrána Jonatána do nového dňa veľmi nepovzbudila.

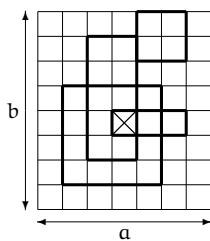
ÚLOHA: Predpokladajme, že chlieb je obdĺžnik s rozmerom  $a \times b$  (kde  $a$  a  $b$  sú celé kladné čísla menšie ako 100) a všetky druhy oblohy na chlebe sú tiež obdĺžnikové s rovnakou hrúbkou 1. Navyše jednotlivé kúsky oblohy sú poukladané tak, že ich strany sú rovnobežné so stranami obdĺžnika tvoriaceho chlieb. Žiadna časť oblohy neprečnieva cez krajec. Ľavý dolný roh krajca má súradnice  $[0,0]$ . Vidlička má zuby dĺžky  $k$  a teda chlieb ňou možno napichnúť iba na tých miestach, kde je na chlebe menej ako  $k$  plátok oblohy.

Napište program, ktorý načíta rozmery a súradnice umiestnenia každého kúsku oblohy a dĺžku zubov vidličky a vypíše, či existuje na chlebe miesto, kde sa vidlička nedostane až po chlieb (t.j. či existuje bod, ktorý leží vo vnútri aspoň  $k$  obdĺžnikov určujúcich jednotlivé plátky oblohy).

Príklad:

Vstup:  
 $a = 7, b = 8, k = 3$   
 Rozmery      Ľavý dolný roh  
 $4 \times 4$        $[1, 1]$   
 $2 \times 2$        $[4, 6]$   
 $3 \times 1$        $[3, 3]$   
 $2 \times 5$        $[2, 2]$

Výstup:  
 Na chlebe existuje miesto, kde sa vidlička zabodne iba do oblohy!



### z1525. Záclony čarodejnice Kirky

Čarodejnica Kirka si kúpila nové záclony. Nie že by sa jej nepáčili pavučinky jej domácich miláčikov, jednoducho sa jej po dlhých rokoch zažiadalo čosi zmeniť. Nové záclony musia byť samozrejme vzorne rovnomerne zavesené, aby vynikol ich vzor. Okrem toho vzdialenosť dvoch susedných štipcov môže byť najviac  $d$ , lebo inak by záclony nepekne ovisali.

Kirka si na vešanie vymyslela efektný spôsob. Využíva pri tom svoj čertovsky dobrý odhad, pomocou ktorého vie presne určiť stred medzi dvoma bodmi, v ktorých je záclona upevnená. Najprv upevní záclonu na oboch koncoch. Pomocou svojho čertovsky dobrého odhadu upevní záclonu presne v jej strede. Potom postupuje tak, že si vždy vyberie nejaký úsek medzi dvoma štipcami, ktorý je ešte dlhší ako  $d$  a presne v strede tohto úseku pripevní záclonu ďalším štipcom. Tak postupuje, kým všetky úseky nemajú dĺžku nanajvýš  $d$ . Je zrejmé, že nakoniec bude záclona zafixovaná v pravidelných intervaloch. Problém je v tom, že Kirka už nie je žiadna storočná mladica a jej staré kĺby si žiadajú čo najmenej pohybu. Preto chce jednotlivé úseky rozdeľovať v takom poradí, aby jej staručká ruka prešla pri vešaní záclony najkratšiu možnú vzdialenosť.

ÚLOHA: Napíšte program, ktorý načíta dĺžku záclony  $l$  a maximálnu vzdialenosť susedných štipcov  $d$  a vypíše, v akom poradí treba vešať jednotlivé štipce, aby sa Kirka čo najmenej narobila. Jednotlivé štipce vo výstupe zadávajúte ich vzdialenosťou od ľavého okraja záclony.

Dôležitou súčasťou riešenia je tiež dôkaz, že postupnosti, ktoré vypisuje váš program, sú skutočne tie najlepšie.

Napríklad pre  $l = 8$  a  $d = 1$  musí Kirkina ruka prejsť vzdialenosť 23. Poradie štipcov môže byť 0, 8, 4, 2, 1, 3, 6, 7, 5 (existujú aj iné rovnako vhodné poradie).

### z1531. Zmagorený marsochod

Písal sa rok 2020, výskumy povrchu Marsu boli v plnom prúde, keď do ústredia KSP (Kontrola Skúmania Planét) prišla správa o zvláštnom správaní istého marsochodu Z-10. Z-10 sa pri návrate podarilo dostať na náhornú plošinu, na ktorej je základňa. Na tejto plošine sa mu však pokazili niektoré motorické obvody, a preto sa dokáže hýbať iba smerom na juh a východ o násobky dĺžky 100m (nedokáže sa totiž otáčať a opačný náhon kolies bol zablokovaný). Našťastie sa základňa nachádza juhovýchodne od Z-10. Medzi základňou a Z-10 sú však rozličné prekážky (vírivé magnetické polia, iónové mračná, výmole, priepasti), cez ktoré nemožno prejsť. Takisto sa od Z-10 chce, aby cestou na základňu vykonal čo najviac práce, ktorá bola preň plánovaná.

ÚLOHA: Na vstupe je číslo  $n$  – rozmer mapy ( $n \leq 100$ ) a matica  $n \times n$  celých čísel, ktoré predstavujú mapu náhornej plošiny. Jedno políčko matice reprezentuje štvorec terénu s rozmerom  $100 \times 100$  metrov. Políčka so zápornou hodnotou predstavujú prekážky, políčka s kladnou hodnotou sú tie, na ktorých treba niečo spraviť. Ostatné políčka majú hodnotu 0. Ďalej sú na vstupe súradnice  $(r, s)$  štvorčeka mapy, na ktorom stojí Z-10 ( $r$  je riadok,  $s$  je

stĺpec). Základňa sa nachádza na pravom dolnom políčku mapy (toto políčko má súradnice  $(n, n)$ ).

Nájdite takú cestu, po ktorej môže Z-10 prejsť (t.j. neprechádza cez žiadnu prekážku) a na ktorej leží najväčší možný počet miest, kde treba niečo vykonať. Cesta je postupnosť políčok mapy začínajúca v  $(x, y)$ , končiac v  $(n, n)$ , pričom každé políčko je buď priamo pod predchádzajúcim alebo napravo od neho. Ak existuje takýchto ciest viac, vypíšte ľubovoľnú z nich, ak neexistuje žiadna, vypíšte príslušnú správu.

Príklad:

Vstup:

$n = 3, r = 1, s = 1$

Mapa:

```
0 1 1
0 -1 -1
0 3 0
```

Výstup:

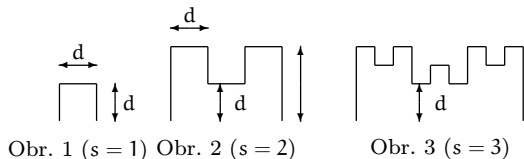
Najlepšia cesta:

$(1, 1), (2, 1), (3, 1), (3, 2), (3, 3)$

### z1532. Zeofinine veže

Po mnohých nástrahách sa Zeofine podarilo takmer nemožné – vrátila sa na rodné Galapágy. Predtým však prešla hodný kus sveta. A všetky užovky, volavky, ba aj korytnačie družky sa začali Zeofiny vypytovať, ako v tom veľkom svete bolo. Zaujímali sa najmä o Európu. Tu Zeofinu obzvlášť zaujali hrady a zámky. Chcela im nejaké nakresliť. Zistila však, že ak chce, aby pochopili, ako hrady vyzerajú, musí ich najskôr oboznámiť s niečím jednoduchším – napríklad s vežami.

Veža stupňa 1, resp. 2 a veľkosti  $d$  je na obr. 1, resp. obr. 2. Vežu stupňa  $s + 1$ ,  $s > 1$  a veľkosti  $d$  dostaneme z veží nižších stupňov nasledujúcim spôsobom. Najskôr nakreslíme zvislú čiaru dĺžky  $d$ , na ňu nadväzuje veža stupňa  $s$  a veľkosti  $\frac{d}{2}$ . Za touto podvežou nasleduje skupinka vežíčiek, ktoré by sme dostali, ak by sme vežu stupňa  $s$  prevrátili horeznačky, za touto skupinkou je opäť veža stupňa  $s$ , za ktorou sa ešte dokreslí zvislá čiaru s dĺžkou  $d$ .



Obr. 1 ( $s = 1$ ) Obr. 2 ( $s = 2$ )

Obr. 3 ( $s = 3$ )

ÚLOHA: Napište program, ktorý pre zadané  $s$  a  $d$  vypíše postupnosť príkazov pre Zeofinu tak, aby nakreslila vežu stupňa  $s$  a veľkosti  $d$ . Zeofína pozná príkazy **DOPREDU** a (posunutie v smere aktuálneho natočenia o dĺžku  $a$ ; zanecháva pri tom za sebou čiaru), **VLAVO**  $u$  a **VPRAVO**  $u$  (zmena aktuálneho natočenia o uhol  $u$ ).

### z1533. Znak krásy

Kde bolo, tam bolo, za siedmimi horami a siedmimi dolami žil raz jeden kráľ a ten kráľ mal jednu-jedinkú dcéru. A tá bola taká pekná, aká bola hlúpa. A veru, väčšiu špatu nepoznalo ani Zrkadielko zo susedného Snehulienkinho kráľovstva. Ani hlúpy Jano by si ju nevzal za ženu, hoc aj s pol kráľovstvom. Ale všetci si ju vážili pre jej múdrosť a zdalo sa, že princeznej krása nechýba. Avšak len do dňa, keď sa jej kráľovský otec rozhodol spraviť z nej svoju nástupkyňu, keď už mu Boh nepožehnal syna a zať bol v nedohľadne. Princezná bola síce bystrá a múdra, ale ako môže svojím vzhľadom reprezentovať vlastnú krajinu v zahraničí? A tak princezná putovala od jedného plastického chirurga k druhému, z jedného salóna krásy do druhého, ale výsledok nebol omnoho lepší. Jediná nádej bola v starej dobrej čarodejnici Kirke. Kirka usadila princeznú do zaprášeného kresla a čosi si mumlala popod nos, zatiaľ čo v kotlíku bublala tajomná tekutina. A čuduj sa svete, kúzlo sa podarilo! Z voňavých pár vystupujúcich z kotlíka sa vyformoval akýsi obrazec

a ľahko ako pavučinka pristál na nových Kirkiných záclonách. Bol to univerzálny znak krásy. Skôr než odletí, princezná si ho musí odkresliť na papier. Nesmie však zdvihnúť pero z papiera skôr, ako bude celý obrazec nakreslený. Princezná to samozrejme zvládne, ale skúste presvedčiť Kirku, že v tom nie sú žiadne čary, a možno vám aj uverí.

ÚLOHA: Vstupom vášho programu bude popis znaku. Znak krásy sa skladá z niekoľkých uzlov, pričom medzi niektorými z nich vedú čiary. Prvý riadok vstupu obsahuje čísla  $n$  a  $m$ , kde  $n$  je počet uzlov a  $m$  je počet čiar. Uzly sú očíslované číslami  $1, 2, \dots, n$ . Potom nasleduje  $m$  dvojíc čísel uzlov, každá dvojica obsahuje koncové body jednej čiary. Váš program má vypísať postupnosť čísel uzlov udávajúcu, ako sa bude daný obrazec kresliť. Každá čiara musí byť vykreslená práve raz.

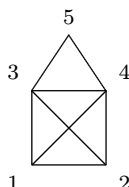
Príklad:

Vstup:

5 8  
1 2  
1 4  
1 3  
3 5  
5 4  
2 3  
2 4  
3 4

Výstup:

1 3 5 4 2 3 4 1 2



### z1534. Zúfalý Santo

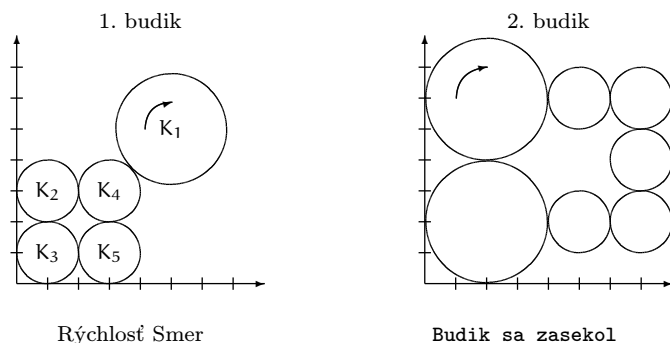
Santo sa otočil a uvidel, že sa za ním objavil stôl plný jedla. Podišiel k nemu, načiahol sa a ČRRRRRRRRRRRN. Santova ruka vyletela a zmietla budík zo stola. Keď sa konečne vyhrabal spod perín, zbadal na zemi rozkotúľané kolieska. Vzdychol, ale keďže Santo je šikovný chlapík, nakoniec sa mu podarilo budík opraviť. No keď ho natiahol, zistil, že sa ručičky točia opačným smerom. Pokúšal sa ho znovu opraviť, ale úplne sa poplietol, ktorým smerom sa má ktoré koliesko točiť. Teraz skúša otáčať rôznymi kolieskami a pozerá, ktoré sa kam točí.

ÚLOHA: Je daný počet koliesok  $n$ , súradnice stredov a polomery všetkých koliesok a smer otáčania prvého kolieska zo vstupu. Žiadne dve kolieska sa v rovine nepretínajú, niektoré dvojice sa však môžu dotýkať. Ak sa dve kolieska dotýkajú, musia sa otáčať navzájom rôznymi smermi. Majme dve dotýkajúce sa kolieska  $K_1$  a  $K_2$ . Nech koliesko  $K_1$  má polomer  $r_1$  a koliesko  $K_2$  polomer  $r_2$ . Ich rýchlosti otáčania (fyzik by povedal uhlové rýchlosti)  $\omega_1$  resp.  $\omega_2$  musia byť v opačnom pomere ako polomery, t.j.  $\frac{\omega_1}{\omega_2} = \frac{r_2}{r_1}$ . Rýchlosť otáčania prvého kolieska je 1 otáčka za minútu. Ak nie je možné tieto podmienky splniť, budík sa zasekne. Naopak, môže sa stať, že nejaká skupinka koliesok nemá žiadne spojenie s kolieskom 1 a preto stojí.

Zistite, ktorým smerom a akou rýchlosťou sa otáča každé z koliesok. Ak sa budík zasekne, vypíšte o tom správu **Budík sa zasekol**. Kolieska, ktoré nie sú v spojení s prvým kolieskom majú nulovú rýchlosť otáčania. Ak takéto kolieska existujú, vypíšte o tom správu **Budík ma nepohanane kolieska**.

Polomery a stredy sú zadávané ako reálne čísla. Počítajte s tým, že presnosť vstupov aj výpočtov bude ovplyvnená zaokrúhľovacími chybami (pozor na porovnanie).

Príklad:



### z1535. Zakrývanie koberca

Na Kiribati vládne zhon. Čoskoro do ich krajiny príde na návštevu náčelník susedného súostrovia a Kiribatčania chystajú slávnostné privítanie. Voľakde zohnali dokonca aj dlhočizný červený koberec a rozhodli sa rozprestrieť ho od letiska až do stredu ich hlavnej osady. Ako ho tak rozprestierajú, zrazu zistili, že na mnohých miestach je koberec prehryzený od molí!

Aby v hanbe nezostali, musia s tým niečo spraviť. V mnohých kiribatských domácnostiach používajú malé červené koberčeky, ktoré sú síce rovnako široké, ako dlhočizný koberec, ale dlhé sú iba jeden meter. Niekoľko napadlo, že by sa týmito koberčekom dali zakryť úseky koberca, ktoré sú prehryzené od molí. Chceli by však použiť čo najmenej koberčekov, aby v domácnostiach zbytočne nechýbali.

ÚLOHA: Daná je dĺžka koberca  $d$ , počet poškodených úsekov koberca  $n$  a pre každý poškodený úsek sú dané vzdialenosti jeho začiatku a konca od letiska (v metroch). Vypíšte najmenší počet koberčekov, aký stačí na zakrytie, ak každý poškodený úsek musí byť celý zakrytý koberčekom, koberčeky nemožno strihať (po odchode návštevy ich treba vrátiť pôvodným majiteľom), ale môžu sa navzájom prekrývať. Ďalej vypíšte, ako treba jednotlivé koberčeky poukladať, aby sa zakryli všetky poškodené úseky.

Dôležitou súčasťou riešenia je dôkaz, že váš algoritmus je správny, t.j. že vždy pozakrýva diery pomocou najmenšieho možného počtu koberčekov.

Príklad:

Vstup:

$d = 10$ ,  $n = 3$

Poškodené úseky:

Začiatok	Koniec
1.2	1.3
5.4	6.5
1.95	2.1

Výstup:

Stačia 3 koberčeky.

Rozmiestnenie koberčekov:

Začiatok	Koniec
1.15	2.15
5.4	6.4
5.5	6.5

### 1611. O kubomíre

Kubomír je nanajvýš jednotvárne miesto. Pozostáva z  $n$  štvorcových políček usporiadaných do radu vedľa seba. V Kubomíre žije jediný druh živočícha – kubár, a jedna nadzmyslová neuchopiteľná bytosť bez tvaru, farby, chuti a zápachu – Kuboh.

Kubár má tvar kocky – vznikne, kedy a kde si Kuboh zmyslí. Kubár potom proste JE, EXISTUJE, ale inak nič – ani sa nepohne. Potom si zrazu Kuboh zmyslí, že kubár

zanikne, a kubár naozaj zanikne. Kuboh si zavše zmyslí niekoľko nových kubárov vedľa seba, napríklad od políčka a až po políčko b. Hneď nato pribudne po jednom kubárovi na políčkach  $a, a+1, a+2, \dots, b-1, b$ . Takisto si môže Kuboh zmyslieť, že zanikne po jednom kubárovi na políčkach c až d. Kuboh je neomylný, teda niečo také si zmyslí iba vtedy, keď na týchto pozíciách je po aspoň jednom kubárovi. A takto stĺpce kubárov na jednotlivých políčkach Kubomíru stúpajú a klesajú.

Z času na čas sa Kuboh filozoficky zamyslí: clqq A koľkože to máme kubárov na r-tom políčku? Alebo: „A na koľkýchže políčkach máme aspoň jedného kubára?“ Z dlhej chvíle sa naučil programovať a teraz trávi dlhé chvíle dumaním nad tým, ako by to naprogramoval. Ukážte, že ste aspoň takí dobrí ako Kuboh.

ÚLOHA: Je daný počet políčok Kubomíru  $n$ . Na začiatku je Kubomír prázdny. Na vstup prichádzajú správy, ktoré vášmu programu oznamujú všetky zmeny v Kubomíre, alebo požadujú nejaké hodnoty o aktuálnom stave. Úlohou vášho programu je pomocou vhodnej reprezentácie stavu Kubomíru správne a rýchlo reagovať na všetky správy a poskytovať požadované údaje. Každá správa je jedného z nasledujúcich typov:

VZNIK  $a$   $b$  – na políčkach  $a$  až  $b$  vzniklo po jednom kubárovi ( $1 \leq a \leq b \leq n$ ).

ZÁNIK  $c$   $d$  – na políčkach  $c$  až  $d$  zaniklo po jednom kubárovi ( $1 \leq c \leq d \leq n$ ; môžete predpokladať, že na políčkach  $c, \dots, d$  predtým bolo aspoň po jednom kubárovi).

VÝŠKA  $r$  – vypíšte, koľko kubárov je na  $r$ -tom políčku tajničky ( $1 \leq r \leq n$ ).

ASPOŇ JEDEŇ – vypíšte, na koľkých políčkach sa nachádza aspoň jeden kubár.

Príklad keď  $n = 6$

> VZNIK 2 6

> VZNIK 4 4

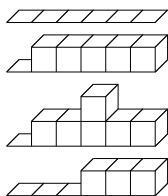
> VÝŠKA 5

1

> ZÁNIK 2 4

> ASPOŇ JEDEŇ

3



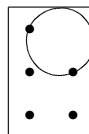
## 1612. O pažravej Lívii

„Mééééééééé, mééééééééé,“ začula teta Anastázia zo svojho domčeka. Koza Lívia sa znovu priblížila až k domčeku na koniec záhrady. A tam má teta Anastázia posadených najviac kvetov. Dnes ju už aspoň osemkrát vyhnala pást sa na iný koniec, kde tých kvetov nie je až tolko, ale márnosť nad márnosť! Tak sa nakoniec teta Anastázia s ťažkým srdcom rozhodla Líviu priviazať. Začala v záhrade hľadať čo najväčšiu plochu takú, aby tam neboli žiadne kvetiny a aby Lívia mala dosť veľký priestor na pasenie sa.

ÚLOHA: Teta chce priviazať kozu Líviu špagátom o kolík. Potrebuje nájsť také miesto na zapichnutie kolíka, aby sa Lívia mohla pásť na čo najväčšej ploche, ale pritom musí zvoliť takú dĺžku špagátu, aby Lívia nemala v dosahu žiadnu kvetinu a nemohla vyjsť za hranicu záhrady. Záhrada má tvar obdĺžnika  $a \times b$ .

Napište program, ktorý načíta rozmery záhradky  $a \times b$ , počet kvetov  $n$  a súradnice jednotlivých kvetín  $(x_1, y_1)$  až  $(x_n, y_n)$  a nájde stred a polomer kružnice s najväčšou plochou, ktorá leží celá v záhradke a neobsahuje žiadnu kvetinu.

Napríklad pre záhradku s rozmermi  $a = 3.0$ ,  $b = 2.0$ . Päť kvetov ( $n = 5$ ) so súradnicami  $(0.5, 0.5)$ ,  $(0.5, 1.5)$ ,  $(1.5, 0.5)$ ,  $(1.5, 1.5)$ ,  $(2.5, 0.5)$  je výsledkom kružnica so stredom  $(2.225, 1.225)$  a polomerom  $0.775$ .



## 1613. O predvolebných mítingoch

Kráľ Burundi sa díval na klesajúci graf svojich preferencií. Nie že by to bol dôvod na znepokojenie, ale bolo mu z toho smutno. I dal si zavolať hlavného radcu a spýtal sa ho, čo

s tým. Radca poradil: „Ludia majú radi zmenu a vaše Veličenstvo sedí na tróne už desať rokov. Ponúknite im ju, usporiadajte voľby!“ Kráľ sa zhrozil, keď mu však jeho poradca vysvetlil, že bude len jeden kandidát, nadšene súhlasil.

Onedlho si už kráľ prezeral vypracovaný harmonogram predvolebných mítingov. Na zozname vždy bolo mesto a deň, kedy sa tam bude konať míting. Keď sa kráľ prizrel lepšie, s hrôzou zistil, že mítingov je príliš veľa a že sa nestihne všetkých mítingov zúčastniť. Burundi je totiž veľká krajina so zlými cestami, preprava z jedného mesta do iného môže trvať aj niekoľko dní. Kráľ neváhal a sľúbil okamžite 50 percent akcií Západoburundijských Železniarní tomu, kto mu nájde takú trasu predvolebného výjazdu, aby stihol čo najviac mítingov.

ÚLOHA: Burundi má  $n$  miest očíslovaných 1 až  $n$ . Máte k dispozícii zoznam burundijských ciest (všetky sú obojsmerné), pre každú cestu viete, ako dlho trvá jej prejedenie autom. Ďalej pre každý míting viete, v ktorom meste a ktorý deň sa koná. Kráľ je v deň 0 v hlavnom meste (mesto č. 1). Kráľ sa na mítingoch zdrží iba zanedbateľný čas, jeho predvolebná cesta nemusí končiť v hlavnom meste. Napíšte program, ktorý načíta zoznam burundijských ciest a zoznam mítingov a zistí, koľko najviac mítingov môže kráľ stihnúť.

Príklad

Vstup:

4 mestá

cesty:

z 1 do 2 za 4 dni

z 1 do 3 za 3 dni

z 2 do 3 za 4 dni

z 2 do 4 za 5 dní

z 3 do 4 za 2 dni

mítingy:

mesto 1 deň 3

mesto 2 deň 4

mesto 3 deň 9

mesto 1 deň 12

mesto 4 deň 17

Výstup:

Dajú sa stihnúť najviac

4 mítingy.

#### 1614. O výskumníčke Janke

Janka je výskumníčka a pracuje v laboratóriu na výskum inteligencie živočíšnych druhov. Inak je to malá biela myška, ktorá má rada syr. Po niekoľkých týždňoch trpezlivej práce sa jej podarilo vycvičiť si svojho človeka. Vždy, keď Janka vybehne po rebríku, dostane kúsok chutného syra.

Teraz však Janka nie je hladná. Okolo seba mala porozhadzovaných niekoľko kociek s písmenami. Väčšinu upratala kdesi do kúta, ale zopár ich tu stále ostalo. Len tak na skúšku ich usporiadala tak, aby písmená na nich tvorili palindróm, t.j. slovo, ktoré je rovnaké, či ho čítate spredu alebo odzadu. Aké bolo jej prekvapenie, keď o chvíľu prišiel človek s kusom ementálu a hľbou ďalších kociek. Poukladal ich do radu a nenápadne sa vzdialil. Janke bolo hneď jasné, čo od nej človek chce. Medzi poukladané kocky má povesúvať niekoľko ďalších tak, aby vznikol palindróm. Ihneď začala premýšľať, koľko najmenej kociek bude treba do radu vsunúť (kocky sú ťažké a Janka sa nechce príliš namáhať).

ÚLOHA: Napíšte program, ktorý načíta vstupné slovo  $w = a_1a_2 \dots a_n$  a zistí, koľko najmenej písmen musíme pridať, aby z neho vznikol palindróm. Nové písmená môžeme vsúvať medzi ľubovoľnú dvojicu susedných písmen, na začiatok a na koniec. Jeden takto vzniknutý palindróm aj vypíšte.

Vstup:

abab

obyamamalybk

Výstup:

babab (pridaných písmen: 1)

kobylamamalybok (pridaných písmen: 3)

#### 1615. O profesorovi Indigovi a víle Amálke I

Iste ste si všimli, že dobrý koniec mnohých rozprávok majú na svedomí rôzne dobré víly a im podobné živly, ktoré keď je najhoršie, dajú hlavnému hrdinovi dobrú radu. Všimol si to aj profesor Indigo a hneď vymyslel, ako by sa dala táto ich dobrá vlastnosť využiť. Indigo si uvedomil, že aj keď počítače vedia riešiť mnoho úloh, niekedy ani ony

nevedia, kam z konopí. Občas by sa aj počítaču hodila nejaká dobrá rada. Profesor sa rozhodol skonštruovať prefikáný počítač (*PP*), ktorý bude ťažiť zo spolupráce s dobrou vílou. Dohodol sa s vílou Amálkou, že bude pomáhať prefikánu počítaču s výpočtami. Tak vznikol počítač *PP<sub>Amálka inside</sub>*.

Prefikáný počítač vie riešiť úlohy, na ktoré je odpoveď áno/nie. Profesor preňho spravil aj kompilátor Pascalu (resp. C/C++), ktorý zatiaľ nepodporuje žiadne príkazy na prácu so vstupno-výstupnými zariadeniami (ako obrazovka, klávesnica alebo disk). Na odovzdávanie vstupných údajov programu sa používajú parametre odovzdávané pomocou rezervovaného slova **program**. Aby bolo možné ako parametre odovzdávať aj zložené typy, povolil deklarácie typov a konštánt pred slovom **program**. Napríklad

```
const MAX = 10;
type bod = record x, y : integer; end;
program MojPrg(B : bod; p : array[1..MAX] of byte);
```

je hlavička programu, ktorý na vstupe dostane jeden údaj typu bod a pole 10 bajtov. Na ukončenie programu slúžia dva špeciálne príkazy *fail* a *accept*. Príkaz *accept* znamená, že program dá odpoveď áno. Príkaz *fail* znamená čosi ako „nepodarilo sa mi dať odpoveď áno“ (presnejší popis ďalej).

Poslednou novou konštrukciou je príkaz **choose**, ktorý má rovnakú štruktúru ako príkaz **case**, až na to, že neobsahuje žiadny výraz, podľa ktorého sa program vetví. Napríklad

```
choose
  1 : Príkaz1 ;
  2 : begin Príkaz2; Príkaz3; end;
end;
```

Vždy keď *PP<sub>Amálka inside</sub>* narazí pri vykonávaní programu na príkaz **choose**, víla Amálka rozhodne, ktorá vetva sa vykoná.

V jazyku C/C++ sa vstupné údaje odovzdávajú ako parametre funkcie *main*. Príkazy *fail* a *accept* fungujú podobne ako **return**, až na to, že nemajú žiadne parametre a ukončujú program. Konštrukcia **choose** je analógiou konštrukcie **switch** bez riadiaceho výrazu – Amálka rozhodne, na ktoré návěstie *case* treba skočiť. Pre iné programovacie jazyky si navrhnete obdobné konštrukcie sami.

Amálka riadi beh programu tak, aby vždy, keď to je možné, bola odpoveď áno. To znamená, že vždy, keď pre daný vstup existuje taká postupnosť vetvení **choose**, ktorá dovedie program k príkazu *accept*, *PP<sub>Amálka inside</sub>* dá odpoveď áno. Ak takáto postupnosť neexistuje, *PP<sub>Amálka inside</sub>* dá odpoveď nie (Amálka ako správna víla vie predpovedať, že žiadna postupnosť vetvení nevedie k príkazu *accept* a zastaví výpočet). *PP<sub>Amálka inside</sub>* dá teda odpoveď nie, ak každá z postupností vetvení **choose** vedie buď k príkazu *fail*, k inému spôsobu skončenia programu, alebo výpočet pri tejto postupnosti beží donekonečna.

Profesor Indigo napísal prvý program pre prefikáný počítač. Tu je:

```
const MAX = 100;
type pole = array[0..MAX - 1, 0..MAX - 1] of integer;
program PrvyProgram(N : integer; p : pole);
var x, y : integer;
```



```

begin
  x := 0; y := 0;
  repeat
    choose
      vľavo : x := x - 1;
      vpravo : x := x + 1;
      hore : y := y - 1;
      dole : y := y + 1;
    end;
    if (x < 0) or (x ≥ N) or (y < 0) or (y ≥ N) then fail;
    if p[x, y] ≠ 0 then fail;
    if (x = N - 1) and (y = N - 1) then accept;
  until false;
end.

```

Profesor si však nie je celkom istý, či všetko pracuje tak, ako má. Pre kontrolu by potreboval program, ktorý robí presne to isté, ale nevyužíva schopnosti víly Amálky.

ÚLOHA: Napište program, ktorý robí presne to isté ako profesorov (t.j. na rovnaké vstupy dávajú oba programy rovnaké odpovede), ale nepoužíva konštrukciu *choose* a vždy skončí príkazom *accept* alebo *fail* (váš program teda nepotrebuje vílu Amálku).

### 1621. O magickom symbole

Jedného dňa sa miestny kúznik Kin Lezuk vážne zamyslel. Potreboval totiž zistiť, koľko najmenej dní mu potrvá, kým aktivuje celý magický symbol okolo dediny na ochranu pred temnými silami. Magický symbol je zložený z  $n$  magických kameňov, ktoré ležia na kružnici, ktorej stredom je dedina. Aktivácia prebieha nasledovne. Kin sa ráno zobudí a vydá sa k niektorému kameňu. Potom postupuje po kružnici celý deň rovnakým smerom a zaklína všetky magické kamene po ceste. Po zakliatí niekoľkých kameňov sa vráti domov (čiže ide od posledného zakliateho kameňa do dediny). Kin však môže putovať najviac  $h$  hodín denne, aby bol pred zotmením doma. Aktivácia je ukončená, ak zakliat každý kameň aspoň raz (na poradi zaklínania jednotlivých kameňov nezáleží).

ÚLOHA: Napište program, ktorý vypočíta, koľko dní Kin potrebuje na aktiváciu kúzla. Vstupom programu bude počet hodín  $h$  (maximálna doba trvania Kinovej cesty), počet magických kruhov  $n$  a ďalej pre každý kameň čas, koľko trvá cesta z dediny ku nemu, a koľko trvá cesta od tohoto kameňa do dediny. Taktiež pre každé dva susedné kamene na kružnici dostane program čas cesty medzi nimi (jedným aj druhým smerom). Čas potrebný na zakliatie kameňa je zanedbateľný.

Napríklad pre  $h = 3$ ,  $n = 3$  a časy v tabuľke mu to bude trvať 2 dni.

	Tam Späť			Tam Späť	
dedina – kameň 1	1	3	kameň 1 – kameň 2	1	2
dedina – kameň 2	2	1	kameň 2 – kameň 3	1	2
dedina – kameň 3	2	1	kameň 3 – kameň 1	4	1

### 1622. Ako Janko s Marienkou bytovú otázku riešili

Iste si všetci živo pamätáte na rozprávku o perníkovej chalúpke. Áno, tak ako vy ste boli deti, aj Janko a Marienka boli deti a medzičasom vyrástli. Napriek tomu, že sa doteraz poctivo zúčastňujú každého zbierania jahôd a hádzania ježibaby do pece, nemajú si za čo kúpiť byt. A tak raz, keď už za ježibabou zatvárali na peci vrátka, pošepla Marienka Jankovi: „Janičko, veď je to celkom dobrá chalúpka, čo keby sme sa tu zabývali?“ Nelenili teda a začali opravovať svoj nový domček, aby im nezatekalo (viete predsa, že predtým z chalúčky odlomili a zjedli niekoľko perníkov). Marienka vyvaľkala cestu a Janko vykrajoval rôzne tvary. Už mali takmer všetko hotové a z cesta na doske zostali len zvyšky, keď si

všimli, že Janko zabudol vykrojiť perník na vetrák v tvare kruhu. Teraz sedia a dumajú, či sa vôbec kruh z tých zvyškov vykrojiť dá.

Napište program, ktorého vstupom je tvar zvyšku cesta –  $n$ -uholník zadaný súradnicami svojich vrcholov zadaných postupne po obvode proti smeru hodinových ručičiek, a rozmer vetráku – kruhu s polomerom  $r$ . Výstupom bude odpoveď **ano**, ak sa kruhový perník dá zo zvyšku cesta vykrojiť, alebo **nie**, ak sa nedá.

Napríklad pre cestu tvaru  $n = 6$ -uholníka s vrcholmi  $(0,0)$ ,  $(3,1)$ ,  $(5,0)$ ,  $(3,2)$ ,  $(3,4)$ ,  $(1,3)$  a vetráku rozmeru  $r = 1$  je odpoveď **ano**. Pre cestu tvaru  $n = 3$ -uholníka s vrcholmi  $(0,2)$ ,  $(8,0)$ ,  $(2,4)$  vetráku rozmeru  $r = 2$  je odpoveď **nie**.

### 1623. O bále

Kde bolo tam bolo, za siedmimi horami a siedmimi dolami, kde sa voda sypala a piesok sa lial, bolo raz jedno kráľovstvo. . .

V tomto kráľovstve sa každoročne na kráľovskom zámku konal veľký ples. Pozvaných bolo mnoho hostí a nemálo dvojíc si zahovorilo tanec. Prastará veštbá však varuje, že posledný bál nesmie mať viac tanečných kôl, ako je zlatých jabĺčok na kráľovskej jabloni. S každým tanečným kolom odpadne jedno jablčko a ak by sa tancovalo aj po spadnutí posledného jablčka, krajinu by postihlo nešťastie.

Každoročne sa stáva predmetom špekulácií, či sa podarí splniť želania všetkých tanečníkov – teda, či sa každej dvojici, ktorá chce spolu tancovať, podarí tancovať aspoň jedno kolo. V tom istom kole môže tancovať ľubovoľne veľa dvojíc za predpokladu, že každý pán tancuje najviac s jednou dámou a každá dáma najviac s jedným pánom.

ÚLOHA: Je daný počet zlatých jabĺčok na kráľovskej jabloni  $p$ , ďalej  $k$  – počet dvojíc, ktoré si dohodli tanec a jednotlivé dvojice – vždy najskôr meno pána, potom meno dámy. Zistite, či je možné uspokojiť želania všetkých tancujúcich a v prípade, že to je možné, vypíšte ľubovoľný vyhovujúci rozvrh tancov. Môžete predpokladať, že mená sú jednoslovné.

Príklady:

Vstup:

$p = 3$ ,  $k = 7$

Janiček Zlatovláska

Bajaja Zlatovláska

Bajaja Marienka

Popolvár Zlatovláska

Bajaja Snehulienka

Janiček Marienka

Popolvár Snehulienka

Vstup:

$p = 2$ ,  $k = 5$

Janiček Zlatovláska

Bajaja Zlatovláska

Bajaja Marienka

Janiček Marienka

Popolvár Zlatovláska

Výstup:

ANO

1: Janíček Zlatovláska, Bajaja Marienka,

Popolvár Snehulienka

2: Janíček Marienka, Bajaja Zlatovláska

3: Popolvár Zlatovláska, Bajaja Snehulienka

Výstup:

NIE

### 1624. Populárna prednáška

Profesor Indigo sa vďaka svojmu novému počítaču stal slávnym a jeho prednášku na univerzite si zapísalo veľmi veľa študentov. Profesor od svojich študentov vyžadoval, aby chodili na všetky prednášky a pri príchode sa zapísali do zoznamu prítomných. Ako správny informatik sa neobťažoval s nejakými menami, ale hneď na začiatku roka očísloval študentov číslami  $1, 2, \dots, n$  a študenti do zoznamu zapisovali svoje čísla v dvojkovej sústave. Všetko išlo ako po masle, až raz na jednej prednáške jeden študent chýbal. Profesor si to hneď všimol a rozhodol sa, že musí zistiť, ktorý to bol. Rozhodol sa napísať si program a

keďže víla Amálka mala ten deň práve dovolenku, musel použiť normálny počítač. Zoznam čísel študentov bol však veľmi dlhý, preto sa mu ho nechcelo do počítača prepisovať celý.

ÚLOHA: Napište program, ktorý načíta počet všetkých študentov  $n$  a potom bude klásť profesorovi otázky typu "Zadajte  $i$ -ty bit (sprava) z  $j$ -teho čísla v zozname" ( $1 \leq j < n$ ) a na základe odpovedí vypíše číslo študenta, ktorý nebol na prednáške. Môžete predpokladať, že v zozname je binárny zápis každého z čísel od 1 po  $n$  okrem jedného. Ak má  $j$ -te číslo menej ako  $i$  bitov, profesor zadá 0. Snažte sa, aby váš program nekládol profesorovi príliš veľa otázok.

Predpokladajme, že  $n = 3$  a v zozname sú čísla 11 a 1, ktoré v desiatkovej sústave zodpovedajú 3 a 1. Chýba teda číslo 10 (2). Práca programu môže vyzeráť napríklad takto:

```
Zadajte N
> 3
Zadajte 1. bit (sprava) z 1. čísla v zozname
> 1
Zadajte 1. bit (sprava) z 2. čísla v zozname
> 1
Vysledok je 10
```

### 1625. O profesorovi Indigovi a víle Amálke II

Profesor Indigo zostavil svoj nový počítač  $PP_{\text{Amálka inside}}$  (popísaný v zadaniach prvého kola) hlavne preto, lebo sa mu zdalo, že to zjednoduší a skráti mnohé programy. Zistil však, že tento počítač má aj ďalšiu výhodu – mnohé výpočty trvali oveľa kratšie ako na bežnom počítači. Aj to umožňujú to obdivuhodné schopnosti víly Amálky:

V prípade, že pre daný vstup neexistuje postupnosť vetvení **choose**, ktorá vedie k príkazu *idaccept*, víla to zistí hneď na začiatku, takže samotný počítač vôbec nemusí prevádzať žiadne výpočty. Ak je teda odpoveď nie, čas výpočtu je 0. V prípade, že odpoveď je áno, výpočet prebehne a víla zvolí v každom príkaze **choose** takú vetvu, že dovedie výpočet k príkazu *accept*. Postupnosť vetvení, ktoré vedú k príkazom *accept* však môže byť viacero a môžu sa veľmi líšiť dĺžkou zodpovedajúceho výpočtu. Víla Amálka sa však vďaka svojej vynikajúcej intuícii vždy rozhodne tak, aby bol výpočet najkratší možný.

Uvažujme napríklad program, ktorý bol uvedený v zadaniach prvého kola. Ak pre daný vstup neexistovala cesta z políčka  $[0, 0]$  na políčko  $[n - 1, n - 1]$  idúca iba po nulách, tak výpočet trval nulový čas. V opačnom prípade bol čas výpočtu úmerný dĺžke najkratšej cesty po nulách medzi týmito dvoma políčkami. Pre niektoré rozmiestenia nenulových prvkov sa však môže stať, že dĺžka najkratšej cesty bude približne  $n^2/2$ , časová zložitosť programu je  $O(n^2)$  aj na počítači  $PP_{\text{Amálka inside}}$ .

Dobrý priateľ profesora Indiga, doktor Jones, je archeológ. K jeho najnovším úspechom patrí nález vzácneho papyrusového zvitku, ktorý patril hlavnému správcovi ciest Nacestejamasovi. Tento papyrus obsahuje zoznam všetkých ciest v krajine. Každá cesta v zozname spájala dve mestá. Tento zoznam je o to cennejší, že je to prvá písomná zmienka o mnohých mestách. Bohužiaľ, všetky názvy miest v papyruse boli doktorovi Jonesovi úplne neznáme. Rozhodol sa, že zoznam porovná s najstaršou dostupnou mapou krajiny. Dúfa, že sa mu takto podarí určiť, akým mestám zodpovedali názvy na papyruse. Doktor Jones vychádzal z týchto predpokladov:

- Každé mesto, ktoré je spomenuté v Nacestejamasovom papyruse, sa nachádza aj na mape, na mape však môžu byť aj ďalšie mestá, ktoré vznikli neskôr ako papyrus.
- Nové mestá mohli vzniknúť buď na už existujúcej ceste, v tom prípade vznikajúce mesto rozdelilo cestu, na ktorej vzniklo, na dve cesty; alebo mimo cesty. Cesty mohli vzniknúť medzi ľubovoľnou dvojicou miest.
- Každá cesta, uvedená v papyruse je aj na mape; mohla byť však rozdelená pri vzniku miest na niekoľko častí.

- Medzi žiadnou dvojicou miest nevedie nikdy viac ako jedna cesta.

Porovnávanie mapy so zoznamom ciest je veľmi zdĺhavá práca. Doktorovi sa doteraz ani nepodarilo zistiť, či sa každému názvu mesta na papyruse dá priradiť nejaké mesto na mape v súlade s predpokladmi. Profesor Indigo teda ponúkol doktorovi Jonesovi, že napíše program pre prefikálny počítač  $PP_{\text{Amálka inside}}$ , ktorý slávnomu archeológovi pomôže. Indigo však večne nemá čas, a tak táto úloha ostáva vám...

ÚLOHA: Očíslujme mestá na mape  $1, \dots, M_m$  a názvy miest v papyruse očíslujme  $1, \dots, M_p$ . Napíšte program pre  $PP_{\text{Amálka inside}}$ , ktorý na vstup dostane počet miest na mape  $M_m$  a počet ciest na mape  $C_m$ , počet miest na papyruse  $M_p$  a počet ciest na papyruse  $C_p$  ako aj zoznamy ciest na mape a na papyruse (každá cesta je určená počiatočným a koncovým mestom). Program má zistiť, či je možné každému číslu mesta z papyrusu priradiť číslo mesta na mape, tak aby toto priradenie spĺňalo Jonesove predpoklady. Presnejšie, váš program má mať hlavičku

```
const MAX = 1000;
type cesta = record z, k : integer; end;
zoznam_ciest = array[1..MAX] of cesta;
program Jonesova_mapa(Mm, Cm, Mp, Cp : integer; ZCm, ZCp : zoznam_ciest);
```

Váš program má teda dať odpoveď áno (za výdatnej pomoci víly Amálky) práve vtedy, ak existuje zodpovedajúce priradenie miest na mape miestam na papyruse. Ak takéto priradenie neexistuje, program má dať odpoveď nie. Skúste tiež odhadnúť časovú zložitosť vášho programu na počítači  $PP_{\text{Amálka inside}}$ .

Napríklad pre  $Mm = 5$ ,  $Cm = 5$ ,  $Mp = 3$ ,  $Cp = 3$ , a  $ZCm = ((1,2),(2,3),(3,4),(4,1),(2,5))$ ,  $ZCp = ((1,2),(2,3),(1,3))$  má dať  $PP_{\text{Amálka inside}}$  výstup *áno*. V tomto prípade existuje viacero možných priradení, napríklad mestu 1 z papyrusu priradíme mesto 3 z mapy, mestu 2 z papyrusu priradíme mesto 4 a mestu 3 z papyrusu priradíme mesto 1.

### 1631. O svokre

Medzi Bonifácove vášne patrí na poprednom mieste cyklistika. Stalo sa však, že ho raz prišla navštíviť svokra. Keď už u neho bývala vyše mesiaca a terorizovala ho neustálym vysávaním, upratovaním a umývaním riadu, rozhodol sa zúfalý Bonifác konečne konať. Navrhol svokre bicyklový výlet. Svokra ihneď súhlasila, netušiac, čo činí. Bonifác totiž sedí nad mapami v snahe nájsť taký cieľ ich bicyklovej vychádzky, ktorý by netrénovaná svokra nezvládla.

ÚLOHA: Mapa je reprezentovaná poľom  $m \times n$ , pre každé políčko je daná jeho nadmorská výška  $v_{ij}$  v metroch. Ďalej je dané číslo  $k$ , ktoré určuje, koľko metrov zvládne svokra dokopy vystúpať počas jednej bicyklovej vychádzky. Bonifác so svokrou začínajú cyklotúru na políčku so súradnicami  $(r, s)$ .

Počas výletu prechádzajú iba medzi políčkami susediacimi hranou a medzi dvoma susediacimi políčkami cesta buď rovnomerne stúpa, alebo rovnomerne klesá, alebo vedie vodorovne. Vodorovné a klesajúce úseky pre svokru nepredstavujú žiadnu námahu, do celkového stúpania sa teda započítavajú iba stúpajúce úseky. Napíšte program, ktorý nájde taký cieľ cesty, do ktorého sa svokra nedostane. Predpokladajte, že svokra si pre daný štart a cieľ vždy nájde cestu s minimálnym celkovým stúpaním.

Vstup:

 $m = n = 4, k = 11$ 

Výšková mapa:

(ľavý dolný roh je (4, 1))

8.0	6.0	8.0	12.0
1.0	7.5	4.0	1.0
2.0	2.0	5.5	1.2
13.0	2.0	1.0	6.2

Štart: (3,2)

Výstup:

Koniec: (1,4)

Prevýšenie: 11.5

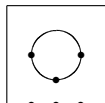
**1632. Ako si Janko s Marienkou živnosť založili**

„Hotovo“ – povedal Janíčko, keď sa mu podarilo nainštalovať posledný kúsok perníka na vetrák. Netrvalo dlho a po celom okolí sa roznieslo, akýže to majú Janko s Marienkou domček. Detváky z dediny chodili obdivovať všetky tie dobroty. I napadlo ich, čo tak si živnosť zriadiť a detvákom z dediny perníky i sladké medovníčky predávať, aby im z šibalstva z novej chalúčky nebodaj neodjedli. Nuž oprášila Marienka sladké recepty starej matere a zamiesila plné koryto cesta s hrozienkami. Celý deň obaja vákali, vykrajovali, cesto do pece strkali, upečené koláče z pece vyťahovali. Večer im ostal iba jeden kus cesta tvaru obdĺžnika  $a \times b$ . Janko schytil starú známu kruhovú formičku, čo ňou kedysi vetrák vykrajoval a podľho vykrojiť si taký kus, ktorý by čo najviac hrozienok obsahoval. Prikladal Janko formičku tak i onak, ale stále sa mu nejako videlo, že by on aj viac hrozienok mať mohol.

ÚLOHA: Napíšte program, ktorého vstupom sú rozmery zvyšku cesta  $a \times b$ , polomer formičky  $r$ , počet hrozienok  $n$  a súradnice jednotlivých hrozienok  $(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)$  a ktorý nájde stred kruhu s polomerom  $r$ , ktorý celý leží v ceste a obsahuje maximálny možný počet hrozienok (hrozienka na hranici kruhu patria dovnútra kruhu).

Napríklad pre  $a = 4.0, b = 4.0, r = 1.0$

a hrozienka na súradniciach (1.0, 0.0), (2.0, 0.0), (3.0, 0.0), (2.0, 1.0), (1.0, 2.0), (3.0, 2.0) má hľadaný kruh stred v bode (2.0, 2.0) a obsahuje 3 hrozienka

**1633. Santo, Banto a parcela**

Banto sa nedávno opäť vybral do krčmy vo Vyšnej Klondike. Aké bolo Santovo prekvapenie, keď sa ani nie o dve hodiny vrátil triezvy, ale o to smutnejší. V krčme sa totiž dozvedel, že vyšlo nové nariadenie, ktoré by mohlo ohroziť ich parcelu.

Santo s Bantom vlastníu parcelu tvaru konvexného  $n$ -uholníka. V každom jeho vrchole je zatĺčený kolík. Nie je to bohvieaká parcela, ale za tie roky, čo na nej strávili ryžovaním zlata, im prirástla k srdcu. Nové nariadenie hovorí toto:

0. Ruší sa staré vymedzenie parciel.

1. Každá nová parcela musí mať tvar trojuholníka.

2. Hranice parcely sa vyznačujú špagátom. Špagát môže viesť medzi ľubovoľnou dvojicou kolíkov.

3. Nie je dovolené zatĺkať nové kolíky za účelom vyznačenia hranice parcely.

4. Špagáty vyznačujúce hranice parciel sa nesmú nikde križovať.

5. Koncom týždňa budú všetky parcely, ktoré nebudú mať tvar trojuholníka, prevedené pod správu šerifského úradu.

Santa napadlo, že by mohli kúpiť špagát a pokúsiť sa ho natiahnuť medzi kolíky tak, aby rozdelili parcelu na trojuholníky. Tak by sa mohli vyhnúť tomu, aby bola ich parcela zhabaná v prospech šerifského úradu. Skôr ako pošle Banta do obchodu by rád vedieť, ako vlastne má špagáty natahovať, aby minul čo najmenej špagátu a koľko špagátu vlastne

potrebuje. A tak, kým Banto smutne sedí na pelasti a čumí do steny, Santo sedí nad nákresem parcely a snaží sa zistiť, ako najvýhodnejšie natiahnuť špagáty.

ÚLOHA: Na vstupe je dané číslo  $n$  a súradnice  $n$  vrcholov  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$  parcely vymenovaných proti smeru hodinových ručičiek. Napíšte program, ktorý poradí Santovi, medzi ktorými dvojicami kolíkov má viesť špagát, aby ho minul najmenej, ako aj celkovú dĺžku potrebného špagátu (špagát možno aj nastrihať).

Napríklad pre  $n = 5$  a vrcholy so súradnicami  $(0.0, 2.0), (2.0, 0.0), (6.0, 2.0), (4.0, 5.0), (3.0, 5.0)$  treba špagát natiahnuť medzi kolíkmi  $(2,5), (3,5), (1,2), (2,3), (3,4), (4,5)$  a  $(5,1)$ . Dĺžka špagátu je spolu 25.490.

### 1634. O reforme

Za čias panovania kráľa Klohodrabata VII. vládla na Kiribati anarchia. Po jeho smrti sa jeho syn Kiripraluk III., následník trónu, rozhodol, že kráľovstvo pozdvihne k rozkvetu. Zvolal mudrcov z celej krajiny, aby spolu vypracovali plán obnovy kráľovstva. Po dôkladnej analýze súčasného stavu dospeli mudrci k záveru, že najpv je treba pozdvihnúť úroveň vzdelania (čo iné už od mudrcov môžete čakať).

Tak Kiripraluk navštívil kráľovskú knižnicu. To čo uvidel ho veru nepotešilo. Bola tam jedna polica zaprášených kníh a podriemkávajúci knihovník v kúte. „Kde sú ostatné knihy?!“ – čudoval sa kráľ. Dozvedel sa, že žiadne ďalšie nie sú, a aj z týchto je iba po jednom rukopise. Tak sa rozhodol dať rozmnožiť aspoň tie knihy, ktoré mali. Najal všetkých pisárov, ktorí v krajine ešte zostali, a prikázal im mať o týždeň z každej knihy ďalšiu kópiu.

ÚLOHA: Je daný počet kníh  $n$ , počet pisárov  $k$  ( $k \leq n$ ) a počty strán  $p_1, \dots, p_n$  jednotlivých kníh v poradí, v akom sú knihy na polici uložené zľava doprava. Každému pisárovi pridelieme niekoľko kníh uložených na polici vedľa seba. Každú knihu celú prepisuje jeden pisár, každý pisár musí prepísať aspoň jednu knihu. Určite, ako rozdeliť knihy medzi pisárov tak, aby celkový čas prepisovania bol minimálny.

Čas prepisovania jedného pisára je úmerný celkovému počtu strán, ktoré má prepísať. Celkový čas prepisovania je čas, ktorý dosiahne pisár, ktorý má najviac práce.

Príklad:

Vstup	Výstup
$n = 9, k = 3$	Prvý pisár: 100 200 300 400 500
Počty strán:	Druhý pisár: 600 700
100 200 300 400 500 600 700 800 900	Tretí pisár: 800 900
	(celkový čas: 1700 strán)

### 1635. O profesorovi Indigovi a víle Amálke III

Profesor Indigo si listoval v zbierke príkladov KSP a keď si čítal príklady prvého kola deviateho ročníka, narazil na takýto príklad:

#### O koláči.

Na obdĺžnikovom plechu upiekli (mimochodom nie veľmi dobrý) koláč rozmerov  $m \times n$  (rozмеры koláča a plechu boli rovnaké) a rozrezali ho na  $k$  kusov. Všetky kusy boli obdĺžnikového tvaru, ale ich rozмеры sa navzájom značne líšili. Hostia sadli ku stolu s koláčom, ale keďže sa veľmi nemali k jedeniu, začali si krátiť dlhú chvíľu skladaním nakrájaných kúskov koláča do pôvodného tvaru, teda do tvaru, aký mal na plechu. Pritom im najväčšie problémy robilo zistenie rozmerov plechu, na ktorom sa koláč piekol. Po dlhej a namáhavej práci sa dohodli na tom, že táto úloha nie je súca pre nich, ale pre účastníkov KSP.

ÚLOHA: Je daný počet  $k$  nakrájaných kúskov koláča obdĺžnikového tvaru rozmerov  $x_i, y_i$ ,  $1 \leq i \leq k$ . Napíšte program, ktorý zistí, aké mohli byť rozмеры plechu  $m \times n$ , na ktorom bol daný koláč upečený (stačí jedno riešenie). Všetky rozмеры sú celočíselné.

Výsledkom programu sú len rozmery  $m$  a  $n$ , nežiada sa nájsť rozloženie daných kúskov na plechu! Pri pečení je celý plech pokrytý cestom.

Profesor sa rozhodol napísať program pre  $PP_{\text{Amálka inside}}$ , ktorý by pre dané rozmery koláča  $m$  a  $n$  a zoznam rozmerov jednotlivých kúskov koláča zistil, či sa kúsky koláča dajú naukladať na plech. Ako na potvoru má práve zlomený malíček a nemôže programovať...

ÚLOHA: Napíšte program pre  $PP_{\text{Amálka inside}}$ , ktorý za pomoci víly Amálky zistí, či sa kúsky koláča dajú na plech naukladať. Váš program by (v Pascali) mal mať hlavičku:

```
const MAX = 1000;
type obdlznik = record a, b : integer; end;
    zoznam_kuskov = array[1..MAX] of obdlznik;
program O_kolaci(M, N, K : integer; Zk : zoznam_kuskov);
```

#### 1641. O Kiribatskej mape

Kiribatský kráľ KIRI-KIRI investoval obrovské peniaze do vesmírneho programu, a to mu vytýkajú všetci jeho odporcovia. Aby utišil hlas nespokojného ľudu, KIRI-KIRI sa rozhodol, že vyrieši legendárnu otázku o počte Kiribatských ostrovov. A tak si dal urobiť satelitný snímok s obrovským rozlíšením na ktorom sa dá rozoznať, kde je more a kde je súš. Napíšte pre KIRI-KIRIho program, ktorý mu spočíta koľko ostrovov má Kiribatské súostrovie. Vzhľadom na veľký rozmer mapy si nemôžete pamätať celú mapu, ale iba niekoľko riadkov z nej.

ÚLOHA: Váš program má k dispozícii vstupný súbor. Tento súbor na prvom riadku obsahuje dve celé čísla  $m$ ,  $n$ , kde  $m$  je počet riadkov a  $n$  počet stĺpcov mapy. Nasleduje  $m$  riadkov, každý z nich obsahuje  $n$  znakov 0 a 1, kde 0 označuje more a 1 súš.

Počet riadkov nie je nijako zhora obmedzený ( $m \cdot n$  môže byť omnoho viac ako veľkosť dostupnej pamäti), ale môžete predpokladať, že niekoľko riadkov sa do pamäti zmestí. Výstupom vášho programu má byť počet ostrovov v súostroví. Ostrovy sa môžu dotýkať rohmi.

Príklad:

Vstup: 5 5  
01000  
11100  
01000  
00111

Výstup:

Kiribatské súostrovie má 2 ostrovy.

#### 1642. O tajnej službe

Nemenovaná tajná služba dostala príkaz monitorovať pohyb podozrivých osôb. Sledované osoby by samozrejme nemali tušiť, že ich niekto sleduje. Na tento účel vymyslel riaditeľ onej tanej služby systém SPS – satelitný pozorovací systém.

Každý satelit systému SPS má schopnosť sledovať obdĺžnik územia s rozmermi  $a \times b$ . Kvôli stabilite satelitu musí byť tento obdĺžnik otočený stranou  $a$  vo východozápadnom a stranou  $b$  v severojužnom smere. Pre každú sledovanú osobu agenti tajnej služby zistili polohu dôležitých objektov, v ktorých sa obvykle zdržiava. Je v štátnom záujme, aby všetky objekty, v ktorých sa môžu nachádzať podozrivé osoby, boli ostro sledované. Satelity SPS sú však pomerne drahé, preto sa riaditeľ rozhodol rozmestniť ich tak, aby ich potreboval čo najmenej. Čoskoro bude musieť predložiť návrh rozpočtu na budúci rok, veľmi rýchlo by teda potreboval vedieť, koľko tých satelitov vlastne potrebuje.

ÚLOHA: Napíšte program, ktorý načíta rozmery územia, pozorovaného jedným satelitom, počet sledovaných objektov  $n$  a súradnice všetkých objektov  $(x_1, y_1), \dots, (x_n, y_n)$  a vypíše, koľko najmenej satelitov je treba na to, aby každý objekt bol pozorovaný aspoň jedným satelitom. Satelit sleduje obdĺžnik aj s okrajom.

Například pre  $a = 2$  a  $b = 1.5$  a objekty so súradnicami  $(0.0, 1.0)$ ,  $(-1.0, 0.0)$ ,  $(1.0, 0.0)$ ,  $(0.0, -1.0)$  treba minimálne 2 satelity.

#### 1643. O Danovej kostre

Dano je študent – zoznámte sa – pred rokom odišiel študovať do Ameriky. Chudák, vtedy ešte netušil, ako je tam draho. Štipendium, ktoré mal rozplánované až do leta, minul už teraz. Za posledné dva doláre si kúpil plechovku piva a sadol si na lavičku do parku. Tam si ho všimol „podnikateľ“ Johnny, ktorý hneď pochopil v akej situácii sa Dano ocitol. Johnny ponúkol Danovi, že od neho odkúpi kosti, jeho vlastné kosti, za \$30 kus (tí chlapíci dnes už obchodujú so všetkým).

Dano hneď na druhý deň zabehol na röntgen. Tam zistili, že Dano sa skladá z  $n$  uzlových bodov očíslovaných od 1 po  $n$ , ktoré sú navzájom poprepájané kosťami. Každá kosť spája dva uzlové body. Ďalej je každá kosť charakterizovaná svojou hrúbkou. Keď Dano videl ten obrovský kapitál, ktorý v sebe nosí, zaleskli sa mu oči šťastím a rozhodol sa, že niektoré kosti predsa len predá. Stanovil si však dve podmienky:

1. Každé dva uzlové body jeho tela musia ostať prepojené, teda musí existovať cesta (po kostiach) od každého uzlového bodu do každého iného uzlového bodu.
2. Najtenšia kostička, ktorá Danovi ostane bude najhrubšia možná (vzhľadom na prvú podmienku).

Teda Dano chce, aby ostal súvislý a aby najkrehšia časť jeho kostry bola čo najpevnejšia – logické, nie?

ÚLOHA: Dano si podľa röntgenu vyhotovil zoznam svojich kostí. Tento zoznam udáva pre každú kosť, ktoré dva uzlové body spája a akú má hrúbku. Pomôžte chudákovi Danovi, ktorý na také množstvo údajov nestačí. Poradte mu, ktoré kosti má predať, tak aby zarobil čo najviac a pritom dodržal všetky podmienky, ktoré si stanovil.

Ešte niečo – Dano je skvelý programátor, neprijme hocaký algoritmus! Pokúste sa vyriešiť úlohu čo najefektívnejšie – najlepšie v časovej zložitosti porovnateľnej s počtom kostí.

Například pre  $n = 5$  a kosti vedúce z 1 do 2 hrúbky 10, z 2 do 3 hrúbky 6, z 2 do 4 hrúbky 3, z 3 do 5 hrúbky 7, z 4 do 5 hrúbky 7, z 3 do 4 hrúbky 8 je najvýhodnejšie predať kosti vedúce z 2 do 4 a z 3 do 4, najtenšia má hrúbku 6.

#### 1644. Lalčo lyžiarom

Lalčo všetko bol. Raz bol dokonca lyžiarom. Ako tak schádzal snehové pláne, vhupla mu do hlavy myšlienka: „Možno keby som sa vyviezol ešte dvoma vlekmi a až potom zišiel dolu druhou stranou kopca, bolo by toho lyžovania viac ako stáť v radoch na vleky. Ale keby som sa vyviezol radšej hentými troma vlekmi...“

A tak tam stál na Alpskom grúni a hútal, ktorými vlekmi sa má vyvieť a po ktorých zjazdovkách potom zísť dolu, aby dosiahol čo najlepší pomer času na zjazdovkách a času na vlekoch. Nakoniec to aj vyhútal. Podarí sa to aj vám?

ÚLOHA: V Alpách je  $n$  lyžiarskych stanovišť,  $m$  zjazdoviek a  $k$  vlekov. Zjazdovky aj vleky vždy vedú medzi nejakými lyžiarskymi stanovišťami, zjazdovky vždy z vyššie položeného do nižšie položeného a vleky z nižšie položeného do vyššie položeného. Na vstupe sú zadané 3 celé čísla  $n, m, k$ . Nasleduje  $m + k$  trojíc celých čísel. Prvých  $m$  trojíc popisuje zjazdovky (z ktorého do ktorého stanovišta vedú a trvanie zjazdu), ďalších  $k$  trojíc popisuje vleky (z ktorého do ktorého stanovišta vedú a ako dlho trvá čakanie a vývoz vlekmi). Správna trasa sa skladá z dvoch fáz: v prvej fáze sa Lalčo z nejakého stanovišta vyvezie niekoľkými vlekmi, v druhej fáze sa niekoľkými zjazdovkami spustí naspäť do stanovišta, z ktorého vyrazil. Úlohou vášho programu je vypísať trasu, ktorá má najvyšší pomer medzi časom stráveným na zjazdovkách a časom stráveným na vlekoch.

Například: pre  $n = 5$ ,  $m = 4$ ,  $k = 3$  a zjazdovky vedúce z 1 do 3 (12min), z 2 do 3 (6min), z 3 do 4 (9min) a z 5 do 4 (9min) a vleky vedúce zo 4 do 5 (12min), z 5 do 1



(12min) a zo 4 do 2 (18min) je hľadaná trasa: 4, 5, 1, 3, 4 (pomer 0.875).

#### 1645. O profesorovi Indigovi a víle Amálke IV

Profesor Indigo robil na svojom počítači  $PP_{\text{Amálka inside}}$  pomerne rozsiahle úpravy. Konkrétne sa mu podarilo k počítaču pripojiť rozhranie pre čiernu skrinku. Čierna skrinka je zariadenie, ktoré vie počítať nejakú funkciu. Napríklad čierna skrinka pre funkciu **function**  $cudo(x, y) : \text{integer}$  počítajúcu  $x + y^2$  pre zadaný vstup  $x = 2, y = 3$  vráti hodnotu 11. Profesora ale zaujímajú prefikanejšie čierne skrinky, také, ktoré reprezentujú orientované grafy. Pripojením takejto čiernej skrinky získa počítač prístup k dvom funkciám:

```
function pocet_vrcholov : integer;  
function hrana(a, b : integer) : boolean;
```

Funkcia *pocet\_vrcholov* vracia počet vrcholov grafu  $n$  (vrcholy sú číslované  $1, 2, \dots, n$ ), funkcia *hrana*( $a, b$ ) vracia *true* práve vtedy, ak z vrchola  $a$  do vrchola  $b$  vedie hrana (nemusi platiť  $hrana(a, b) = hrana(b, a)$ ). Obe funkcie vracajú výsledok okamžite, t.j. v jednotkovom čase. Pre iné programovacie jazyky si vhodné definície ľahko domyslíte sami.

Prirodzene, každému grafu prináleží vlastná čierna skrinka. Profesorovi k úspešnému vedeckému výskumu chýba už len jediná maličkosť: Pre graf daný čiernou skrinkou zistiť, či sa dá dostať z každého vrchola do každého.

Neprijemné je, že počet vrcholov je obvykle veľmi veľký a  $PP_{\text{Amálka inside}}$  má obmedzenú pamäť (aby sa do počítača zmestila čierna skrinka, musel profesor vybrať väčšinu pamäťových modulov).

ÚLOHA: Napíšte program pre  $PP_{\text{Amálka inside}}$ , ktorý (nebude mať žiadny vstup) skončí s výsledkom **áno**, ak daná čierna skrinka popisuje graf s uvedenou vlastnosťou. V opačnom prípade všetky možnosti vetvenia výpočtu musia viesť k odpovedi **nie**. Snažte sa minimalizovať pamäťové nároky. Na tom, ako dlho bude výpočet trvať, vôbec nezáleží. Predpokladajte, že *pocet\_vrcholov* sa akurát zmestí do premennej typu **integer**, resp. **int** ( $2 \cdot \text{pocet\_vrcholov}$  sa už nezmestí). Skúste porozmýšľať nad tým, koľko pamäte potrebujete na počítači  $PP_{\text{Amálka inside}}$  a koľko by ste potrebovali, ak by ste nemali k dispozícii služby víly Amálky.

#### z1611. Zoči-voči – voči-zoči

Jeden z tradičných obľúbených turnajov na Kiribati je hra zoči-voči. Pravidlá sú jednoduché: dve družstvá sa postaví zoči-voči a naraz začnú kričať: ZOČI-VOČI NÁM, NEVYHRÁŠ LEN TAK ĽAHKO SÚPER SÁM!

Potom jedno družstvo začne hovoriť: ZOČI-VOČI ZOČI-VOČI... a druhé VOČI-ZOČI VOČI-ZOČI... To družstvo, ktoré sa prvé pomýli, prehráva. Mnohých však už turnaje omrzeli, pretože v nich je víťaz vždy len jeden. Preto sa tohto roku rozhodli pre malú obmenu. Na začiatku začne v turnaji každý účastník sám ako družstvo. Počas turnaja, vždy družstvo, ktoré prehrá hru zoči-voči, pridá sa na stranu vyhrávajúceho družstva a spolu potom pokračujú v turnaji proti ďalším družstvám. Nakoniec ostane jedno veľké družstvo, a tak vyhrávajú všetci. Večer sa všetci zídu a spoločne oslavujú veľkolepé víťazstvo...

ÚLOHA: Napíšte program na vyhodnocovanie priebehu modifikovanej hry zoči-voči. Hru hrá  $n$  hráčov očíslovaných 1 až  $n$ . Váš program na začiatku načíta počet hráčov  $n$  a potom bude v nekonečnom cykle prijímať údaje o priebehu turnaja a otázky o stave a bude na ne príslušným spôsobom odpovedať. Vstupné údaje budú niektorého z týchto typov:

**Porazil** A B Oznámenie, že družstvo, v ktorom je hráč A porazilo družstvo, v ktorom je hráč B. Ak boli A a B už predtým v tom istom družstve, program má podať vhodné chybové hlásenie.

**Spolu** A B Otázka, či sú A a B v tom istom družstve. Program by mal správne odpovedať **áno/nie**.

**Koniec** Aj tak sú všetci spíť na mol, váš program si môže robiť, čo chce, nech si trebárs aj skončí.

Snažte sa, aby váš program reagoval čo najrýchlejšie. Hráčov totiž môže byť veľmi veľa a Kiribatčania sú veľmi netrpeliví.

### **z1612. Zuzkine narodeniny**

Zuzka Bodková bude čoskoro oslavovať narodeniny. Prípravy na veľkú záhradnú oslavu sú v plnom prúde, občerstvenie, torta, ohňostroj... Ešte treba dorobiť zasadací poriadok. Bodkovci majú v záhrade tri dlhočizné stoly. Zuzka chce pozvať na oslavu kopec kamarátok, ktoré treba usadiť k stolom. Ale beda! Niektoré z dievčat sa navzájom neznášajú a v záujme hladkého priebehu osláv by nebolo dobre, aby nejaká dvojica dievčat, ktoré sa neznášajú, sedela pri tom istom stole. Zuzka si spísala zoznam pozvaných kamarátok ako aj zoznam dvojíc dievčat, ktoré sa navzájom neznášajú a začala zostavovať zasadací poriadok. O hodnú chvíľu ju zastihla Kristínka, ako sa stále trápi.

„Pozri, to je jednoduché,“ začala Kristínka. „Najprv treba predsa usporiadať dievčatá podľa toho, s koľkými inými dievčatami sa neznášajú. Aha, Lucia sa znáša s najmenej ľuďmi, večne sa háda s celou triedou. Tú usadíme k prvému stolu. Ďalšia je Danko, tá je s Luciou na nože. Usadíme ju k druhému stolu. Potom máme Janku. Aj ona sa háda s Luciou, s Dankou je však zadobre, takže ju môžeme posadiť tiež k druhému stolu. Takto budeme pokračovať aj ďalej: vždy si vyberieme zo zoznamu dievča, ktoré sa znáša s najmenej inými dievčatami a postupne zistíme, či ju môžeme posadiť postupne k prvému, potom k druhému a napokon k tretiemu stolu. Uvidíš, takto sa nám zaručene podarí všetky usadiť.“ No dievčatá o chvíľu zistili, že ani Kristínkiným spôsobom sa nedajú všetky pozvané priateľky usadiť. „V tom prípade ti neostáva nič iné, ako poprosiť otecka, aby zohnal ešte jeden stôl, lebo žiadny iný spôsob rozsadenia neexistuje,“ povedala Kristínka a odbehla. Zuzku však trápili pochybnosti: nenašiel by sa predsa len nejaký spôsob, ako vystačiť s tromi stolami?

**ÚLOHA:** Zuzkine priateľky si označme číslami 1 až  $n$ . Vašou úlohou bude napísať program, ktorý načíta zoznam dvojíc znepriatelených dievčat a nájde Zuzke jedno možné rozsadenie priateľiek za tri stoly, alebo zistí, že takéto rozsadenie nie je možné. Ak ste presvedčení, že Kristínkina metóda nájde prípustné rozsadenie vždy, keď existuje, naprogramujte ju. V tomto prípade od očakávame aj zdôvodnenie (dôkaz) správnosti vášho programu. Ak si naopak myslíte, že Kristínkina metóda niekedy nenájde rozsadenie priateľiek a pritom takéto rozsadenie existuje, navrhните a naprogramujte svoju vlastnú. Nájdite aj nejaký protipríklad, alebo aspoň zdôvodnite, v čom sa Kristínka mylí.

### **z1613. Z Hanoja**

Po návrate z potuliek po svete si Zoro opäť povedal, že určí dátum konca sveta. Techniku sa naučil od hanojských mníchov, ktorí používajú metódu MPV – metódu presúvania veží. Na posvätnom stole sú tri oceľové tyče, na ktorých sú nastrkané mohutné kotúče rôznej veľkosti. Kedysi dávno, keď bolo vyriešené proroctvo, stáli všetky kotúče na prvej tyči, usporiadané od najväčšieho po najmenší. Vrávi sa, že keď budú všetky kotúče presunuté na druhú tyč, nastane koniec sveta. Avšak na presúvanie kotúčov medzi tyčami sú definované presné pravidlá:

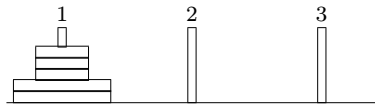
- každý presun trvá presne jednu minútu,
- naraz sa presúva práve jeden kotúč,
- presun pozostáva z odobratia vrchného kotúča z niektorej tyče a jeho navlečenia na niektorú inú tyč,
- nesmie sa položiť väčší kotúč na menší. Nedočkavý Zoro pri tvorbe vlastného modelu tieto pravidlá plne rešpektuje, jediný rozdiel je v tom, že používa také kotúče, ktoré môžu mať rovnakú veľkosť, čo mu ušetrí mnoho presúvania, lebo môže pokladať na seba kotúče rovnakej veľkosti v akomkoľvek poradí.

ÚLOHA: Zoro má  $n$  rôznych veľkostí kotúčov. Najmenších kotúčov je presne  $a_1$ , druhých najmenších  $a_2$ , atď. až najväčších je  $a_n$ . Všetky kotúče sú na prvej tyči od najväčších po najmenšie. Pomôžte Zorovi a napíšte program, ktorý načíta čísla  $n$  a  $a_1$  až  $a_n$  a vypíše najkratšiu možnú postupnosť presunov, pomocou ktorých možno všetky kotúče presunúť z prvej tyče na druhú, pričom sa dodržia všetky pravidlá. Pokúste sa odôvodniť, prečo váš program funguje správne.

Príklad:

Vstup:

$n = 2$ ,  $a_1 = 3$ ,  $a_2 = 2$



Výstup:

z 1 na 3

z 1 na 3

z 1 na 3

z 1 na 2

z 1 na 2

z 3 na 2

z 3 na 2

z 3 na 2

#### z1614. Zanzibarský vláčik

Zanzibarská slonovina vyniesla Zanzibarčanom na ich pomery nemalé zisky a tak sa vláda rozhodla investovať do zlepšenia dopravnej situácie v krajine. Zanzibarské železnice ihneď presadili nákup modernej vlakovej súpravy, ktorú ich železničná sieť potrebuje ako soľ. Do niektorých odľahlých staníc totiž ešte stále zabezpečovala dopravu parná lokomotíva. Pri tvorbe nového cestovného poriadku si však uvedomili, že všetky ich stanice sú primálne na to, aby sa tam nová vlaková súprava dokázala otočiť. Preto pre lokomotívu chcú nájsť takú okružnú trasu, kde by sa nemusela otáčať alebo cúvať. Taktiež chcú, aby okruh neprechádzal viackrát tou istou stanicou.

ÚLOHA: Zanzibarské železnice majú  $n$  staníc očíslovaných  $1, \dots, n$ . Niektoré stanice sú navzájom poprepájané priamymi úsekmi koľajníc, všetky úseky sú obojsmerné. Napíšte program, ktorý načíta počet zanzibarských staníc  $n$  a dvojice staníc, ktoré sú priamo spojené železničnou traťou a nájde okruh, po ktorom môže premávať nová lokomotíva. Ak takých okruhov existuje viac, stačí vypísať ľubovoľný z nich. Ak neexistuje ani jeden, program o tom vypíše príslušnú správu.

Napríklad:

Vstup:

Počet staníc  $n = 6$

Trate:

1 2

3 1

1 4

4 2

2 5

3 4

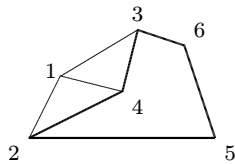
6 3

5 6

Výstup:

Okruh ide cez stanice:

3 4 2 5 6



#### z1615. Z kuchyne

„Ja to takto robiť nebudem,“ vyhlásil kuchtík Karol a dal si ruky vbok. „Ale neblbni, určite sa to nejako dá. Veď si len predstav, že by sme niektorú z tých nádob museli vyliat – a bolo by po dvoch hodinách roboty,“ dôvodil kuchtík Kveťo. „Radšej si rýchlo premyslime, ako to spravíme, lebo o chvíľu tu bude kuchár Kleofáš a... Veď vieš, akú má veľkú varechu“, pokračoval Kveťo.

A čože to našich kuchtíkov takto trápilo? Všetky nádoby, ktoré v kuchyni mali, zaplnili rozličnými krémami do kráľovskej torty. Všetky nádoby až na jeden hrniec – do neho im

ostávalo naliať presne  $n$  decilitrov mlieka na vanilkový krém (div sa svete, ten ešte nemali). Ale beda, hrniec mal objem  $m$  dl. Je síce  $n \leq m$ , ale k dispozícii majú iba veľkú kaďu s mliekom (nevieme, koľko ho tam je, ale je ho tam veľmi veľa) a dve naberačky s objemami  $a$  a  $b$  dl. Hrnec je teraz prázdny a jediné, čo môžu robiť je nasledujúce:

- nabrať ľubovoľnú naberačku plnú mlieka z kade a vyliť ju do hrnca (ak sa celý obsah naberačky do hrnca nezmestí, naleje sa po vrch hrnca a zvyšok mlieka sa vyleje späť do kade),
- alebo nabrať ľubovoľnú naberačku plnú mlieka z hrnca a vyliť ju do kade (ak v hrnci nie je dosť mlieka, naberie sa všetko mlieko z hrnca).

ÚLOHA: Pomôžte kuchtičkom a napíšte program, ktorý načíta celé čísla  $a$ ,  $b$ ,  $m$ ,  $n$  (môžete predpokladať, že platí  $a, b, n \in \{1, 2, \dots, m\}$ ) a vypíše postup operácií, alebo správu, že do hrnca nie je možné dostať uvedené množstvo mlieka.

Napríklad keď máme k dispozícii naberačky s objemami  $a = 4$  a  $b = 5$  a hrniec objemu  $m = 6$  a chceme určiť objem  $n = 1$  musíme postupovať takto: Pridaj 4, Pridaj 2, Odober 5.

### z1621. Z krajiny byrokratov

V krajine byrokratov treba takmer na všetko úradné povolenie. Na úradoch sú teda obrovské rady, v ktorých často stoja aj samotní byrokrati. Ale na takého byrokrata čaká na úrade množstvo ľudí. Preto, aby sa zamedzilo nepokojom súvisiacim s dlhými čakacími dobami, bol do úradov zavedený „Prioritný systém vybavovania byrokratov“. Pri príchode na úrad každý človek oznámi svoje meno a dôležitosť. Dôležitosť je reálne číslo. Čím je väčšie, tým viac je daný človek potrebný v úrade. Žiadni dvaja ľudia nemajú rovnakú dôležitosť. K vybaveniu potom vždy volajú človeka s najväčšou dôležitosťou.

ÚLOHA: Naprogramujte „Prioritný systém vybavovania byrokratov“ ako program pracujúci v nekonečnej slučke, ktorý bude zo vstupu prijímať správy dvoch typov: správy o príchodoch jednotlivých ľudí na úrad a požiadavku na vypísanie mena najdôležitejšieho z čakajúcich (ten je potom vybavený a ďalej už nečaká). Správy sú nasledujúcich tvarov: PRÍCHOD <meno> <dôležitosť> alebo ĎALŠÍ

Na začiatku na úrade nečaká nikto. Môžete predpokladať, že mená sú jednoslovné, dlhé najviac 20 znakov.

Príklad:

```
>PRÍCHOD KATKA 93.3
>PRÍCHOD MATEJ 120.7
>PRÍCHOD PETER -88.9
>PRÍCHOD FILIP 62.34
>ĎALŠÍ
MATEJ
>PRÍCHOD KLEOFÁŠ 1000.3
>ĎALŠÍ
KLEOFÁŠ
>ĎALŠÍ
KATKA
...
```

### z1622. Zoltánov tanečný večer

Zoltán Samba je učiteľom v tanečnej škole. Práve sa končí kurz tanca pre stredoškóľakov a na jeho pleciah spadla zodpovednosť za vydarený priebeh záverečného tanečného večierka – venčeku. Pri prvom tanci by rád videl na parkete čo najviac svojich žiakov. O každom chlapcovi a dievčati vie, či by spolu chceli tancovať, alebo nie. Chcel by žiakov popárovať tak, aby nikto netancoval s niekým, s kým nechce tancovať a súčasne aby vznikol čo najväčší počet párov. A tak si na papier napísal mená chlapcov do jedného stĺpca,

mená dievčat do druhého a skúšal ich popárovať. O chvíľu ho takto zastihla jeho kolegyňa Zita Polková.

„Na to treba ísť systematicky,“ vraví. „Najprv treba popárovať tých, čo sú ochotní tancovať s najmenej partnermi. Tých, ktorí majú veľa potenciálnych partnerov, je dobré spáriť až na konci – pravdepodobnosť, že sa im z nich niekto ujde, je vyššia ako u tých, ktorí nemajú veľký výber partnerov.“

„Pozri, Peter chce tancovať len s Luciou a Lucia len s Petrom, v tomto prípade je to jednoznačné. Ďalší chlapec, pre ktorého existuje najmenší počet potenciálnych partneriek, je Miro: môže tancovať len s Jankou a Dankou. Priradíme mu Danku, lebo Danku môže tancovať s menej chlapcami ako Janka. A máme ďalší pár. Takto postupujeme aj naďalej: Uvažujeme už len o tých chlapcoch a dievčatách, ktorí nemajú pár. Z nich vyberieme chlapca, ktorý je ochotný tancovať s najmenej dievčatami. Z týchto dievčat mu priradíme tú, ktorá z nich môže tancovať s najmenej chlapcami. V prípade nejednoznačnosti výberu (napr. keď niekoľko chlapcov má na výber rovnaký počet dievčat) vyberieme ľubovoľného z nich. Týmto spôsobom vytvoríme maximálny počet párov, aký sa dá.“

ÚLOHA: Počet chlapcov označme  $n$ , počet dievčat  $m$ . Ďalej pre každého chlapca máme zoznam dievčat, s ktorými môže tancovať. Chceme vytvoriť čo najviac tancujúcich párov chlapec – dievča. Samozrejme, každý chlapec môže tancovať s najviac jedným dievčaťom a naopak. Chlapec môže tancovať len s dievčaťom, ktoré sa nachádza v jeho zozname. Ak si myslíte, že metóda Zity Polkovej nájde vždy maximálny počet párov, tak to dokážte a túto metódu naprogramujte. Ak si to nemyslíte, tak uveďte konkrétny príklad, na ktorom metóda zlyhá. (Nemusíte naprogramovať tú správnu, ale môžete sa o to pokúsiť.)

### z1623. Zoro a zlý drak

Zorovi sa podarilo určiť dátum konca sveta a s úľavou zistil, že ešte mu zostal čas na to, aby stihol zabiť zlého draka. Zlý drak žije v jaskyni a má  $n$  hláv. Zabiť draka však nie je vôbec také jednoduché, ako by sa mohlo zdať. Zoro vlastní dva meče – zlatý a strieborný. Ak zaútočí na draka strieborným mečom, zotne mu  $s$  hláv, ak zlatým, zotne  $z$  hláv. Ak však drakovi zostane aspoň jedna hlava, dorastie mu hneď niekoľko nových. Ak Zoro útočil strieborným mečom, dorastie  $a$  nových hláv, ak zlatým, tak  $b$  nových hláv. Zoro nemôže použiť taký meč, ktorý by zotlčil viac hláv ako drak v danom okamihu má. Drak zomrie, ak nemá žiadnu hlavu. Pomôžte Zorovi zistiť, či s danou sadou mečov dokáže poraziť draka, alebo či sa má radšej vrátiť do Hanoja a zohnať iné meče.

ÚLOHA: Vašou úlohou bude napísať program, ktorý najprv načíta čísla  $n$ ,  $s$ ,  $z$ ,  $a$ ,  $b$  a vypíše, či sa Zorovi môže podariť zabiť draka (t.j. odňať všetky jeho hlavy).

Napríklad pre  $n = 5$ ,  $s = 4$ ,  $a = 2$ ,  $z = 2$  a  $b = 3$  je odpoveď áno. (Zoro môže použiť zlatý meč (drakovi zostane 6 hláv) a následne dvakrát strieborný meč, čím draka dorazí.)

### z1624. Zázvorové pivo

Závozník Zachariáš sa živí rozvozom zázvorového piva. Pivo je síce nezdravé, ale všetkým pupkatým zbrojnošom preveľmi chutí, takže Zachariáš pekne zarába a dňom i nocou bohatne. Má to iba jeden háčik. Vždy, keď náš Zachariáš vstúpi aj s vozom piva do hocikákeho mesta, musí zaplatiť mýto a to sa mu veru vôbec, ale vôbec nepáči. Teraz práve je v meste A a chcel by sa dostať do mesta B, kde sa má konať veľký jarmok, na ktorom sa jeho pivo určite bude dobre predávať. Chce tam však docestovať tak, aby musel cestou platiť mýto v čo najmenej mestách. A tak sedí zadumaný na voze a premýšľa.

ÚLOHA: Na vstupe máte zadané prirodzené číslo  $n > 1$  určujúce počet miest (mestá sú číslované číslami 1 až  $n$ ), číslo  $m$  určujúce počet ciest a ďalej zoznam týchto ciest. Každá cesta vedie medzi dvoma mestami a je zadaná dvojicou čísel týchto miest. Ďalej sú zadané čísla miest A a B. Zistite, či sa Zachariáš môže dostať z mesta A do mesta B. Ak áno, vypíšte trasu, vedúcu z A do B cez najmenší možný počet miest. Ak je takýchto trás viac, vypíšte ľubovoľnú.

Príklad:

$n = 7$ ,  $m = 6$

cesty:

1 6

2 3

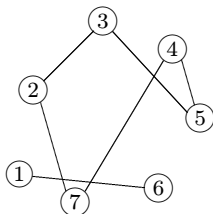
2 7

4 7

4 5

3 5

$A = 3$ ,  $B = 4$



Výsledkom je trasa idúca cez mestá 3, 5 a 4. Existuje ešte trasa 3, 2, 7, 4, tá však ide cez viacej miest. Ak by sa však Zachariáš chcel dostať z mesta  $A = 3$  do mesta  $B = 1$ , odpoveďou programu by malo byť, že cesta neexistuje.

#### z1625. Z centrálneho trhoviska

Raz sa Zoro vybral do mesta nakupovať. Voľakto mu však zákerne premagnetizoval kompas, a tak si po niekoľkých dňoch putovania uvedomil, že pravdepodobne zablúdil. Nezaprel však svoju dobrodružnú povahu, pokračoval vytýčeným smerom, i došiel do mesta takého veľkého, aké nikdy predtým nevidel. Tam sa popýtal náhodného okoloidúceho na smer na trhovisko. Ten mu takto riekol: „Vidím, že nie si tunajší. Máš vôbec peniaze, keď sa ti na trh zachcelo?“. Zoro siahol do mešca, vytiahol tučný zväzok bankoviek a zatriasol s nimi cudzincovi pred očami. Ten odvetil: „S takýmito peniazmi u nás nepochodíš, hybaj ich do banky zameniť. Ale pamätaj, naše nominálne hodnoty sa inakšie počítajú!“

V banke Zoro zistil, že prevodová tabuľka medzi normálnou a tunajšou menou je veľmi komplikovaná – tu totiž uznávajú iba čísla, ktoré sú odpredu rovnaké ako odzadu a nazývajú ich palindrómy. Ako je aj všade inde zvykom, palindrómy majú utriedené, pričom hodnotu palindrómu určuje jeho poradové číslo. Teda napríklad palindróm 6 má hodnotu 6, palindróm 11 má hodnotu 10, alebo palindróm 131 má hodnotu 22. Ako ale Zoro zistí, či má dosť peňazí na zaplatenie meča aj štítu, keď predavač má obe sumy vyčíslené v tunajšej mene?

ÚLOHA: Pomôžte Zorovi a napíšte program, ktorý načíta dva palindrómy  $x$  a  $y$  a vypíše ich súčet – tiež palindróm. Konkrétne, vypíše ten palindróm, ktorého poradové číslo je súčtom poradových čísel palindrómov  $x$  a  $y$ .

Napríklad pre 44 a 2 je výsledok 66 a pre 101 a 101 je výsledok 292.

#### z1631. ZY nevergreeny

Isto viete, že evergreen je pieseň, ktorá zostáva populárna aj dlhý čas po svojom vzniku. Naopak nevergreen je taká pieseň, ktorá ešte nikdy nebola skutočným hitom – či už preto, že je úplne nová, alebo preto, že jej kvalita je príliš nízka (vysoká) na to, aby sa zapáčila davom. ZY Rádio je rozhlasová stanica určená pre náročných poslucháčov, ktorá vysiela iba ZY nevergreeny. To sú také nevergreeny, ktorých meno sa skladá iba z písmen Z a Y.

ÚLOHA: Napíšte program, ktorý pomôže pracovníkom ZY Rádía spoľahlivo určovať, ktorá pieseň je ZY nevergreen. Program bude v nekonečnom cykle prijímať správy od užívateľov a bude na ne vhodným spôsobom reagovať. Správy sú dvoch typov: **Hit**  $m$  a **Skontroluj**  $m$ , kde  $m$  je reťazec obsahujúci iba veľké písmená.

Správa typu **Hit**  $m$  oznamuje programu, že pieseň s názvom  $m$  sa stala hitom a teda už nikdy nemôže byť vysielať na stanici ZY Rádio.

Správa typu **Skontroluj**  $m$  znamená, že niektorý redaktor chce pustiť pieseň s názvom  $m$  a potrebuje vedieť, či táto pieseň je ZY nevergreen, t.j. či je názov pozostáva len z písmen Z a Y a či doteraz nebola zaevidovaná pomocou správy **Hit**  $m$ .

Príklad:

```
> Hit ZYZY
> Skontroluj ABC
ABC nie je nevergreen
> Skontroluj ZZZ
ZZZ je nevergreen
> Hit ZZZ
> Hit AAA
> Skontroluj ZZZ
ZZZ nie je nevergreen
```

### z1632. Zákerný rozvrh

Ako iste všetci viete, Tomáš je veľmi snaživý študent, a preto hneď po zverejnení rozvrhov sa rozhodol zapísať si čo najviac prednášok. Naraz môže byť najviac na jednej prednáške a každú zapísanú prednášku musí absolvovať celú (teda časy prednášok sa nesmú prekryvať). Ako tak pridával a uberal prednášky zo svojho rozvrhu, objavil sa škriatok VIL a povedal: „To máš, Tomáš, jednoduché. Prednášky si budeš zapisovať jednu po druhej. Ako prvú si pekne zapíšeš tú, ktorá skončí najskôr. Potom si vždy zapíšeš takú prednášku, ktorá sa ti neprekrýva so žiadanou už zapísanou, a ktorá končí najskôr. Ak je takých prednášok viac (nutne musia končiť v rovnakom čase), môžeš si vybrať ľubovoľnú z nich. Keď si nebudeš môcť zapísať žiadnu ďalšiu prednášku, vedz, že ich máš zapísaných najviac, ako sa len dá.“

ÚLOHA: Ak si myslíte, že VIL poradil Tomášovi dobre, napíšte čo najefektívnejší program podľa jeho návodu. V tomto prípade zdôvodnite (pokúste sa dokázať), že Tomáš si podľa VILovej rady skutočne zapíše maximálny možný počet prednášok. V opačnom prípade navrhните svoj vlastný spôsob riešenia, naprogramujte ho a nezabudnite zdôvodniť, prečo sa VIL mýli a zdokumentovať to na vhodnom kontrapríklade.

Váš program by mal načítať počet prednášok  $n$ , týždenný rozvrh skladajúci sa z  $n$  riadkov tvaru

$\langle \text{deň v týždni} \rangle \langle \text{začiatok prednášky} \rangle \langle \text{koniec prednášky} \rangle \langle \text{názov prednášky} \rangle$

a vypíše jeden možný maximálny zoznam prednášok, ktoré si môže Tomáš zapísať. Ale pozor! Ak Tomáš zistí, že nemá najvyšší možný počet prednášok, psychicky sa zrúti a máte ho na svedomí.

Príklad:

Vstup:

3 prednášky:

PON 07:20-09:45 Kognitívna  
psychológia

UTO 09:30-10:00 Teória hier

PON 09:00-10:35 Programovanie

Výstup:

Tomáš si môže zapísať max. 2 prednášky.

PON 07:20-09:45 Kognitívna psychológia

UTO 09:30-10:00 Teória hier

### z1633. Zamaskuje Ferdo zamestnanie?

Agent Ferdo pracuje už niekoľko desaťročí pre SIS (Slovenskú informatickú spoločnosť) na istej nemenovanej americkej univerzite. Po dlhé roky sa mu darilo plniť úlohy – získavať technické plány najnovších algoritmov. V súčasnosti je však jeho pozícia ohrozená – tajná služba ho odhalila a vedie ho v databáze pod číslom  $i$ . Ferdo je jeden z najlepších agentov svojho druhu a dokáže sa dostať na krátky čas aj k počítačom tajnej služby. Lenže na to, aby sa dal vymazať niektorý agent z databázy, je potrebné zadať jeho číslo zakódované a na kódovanie sa používa veľmi prefikovaný kódovací algoritmus.

Algoritmus zakóduje každé číslo ako  $k$ -ticu prirodzených čísel v rozsahu od 1 po  $n$ . Pre číslo  $i$  sa príslušná  $k$ -tica nájde takto: Zostrojíme všetky  $k$ -tice z čísel  $1, \dots, n$  a usporiadame

ich lexikograficky (t.j. usporiadame ich najprv podľa prvej cifry; tie, ktoré majú prvú cifru rovnakú, usporiadame ďalej podľa druhej cifry atď.). Z takto vzniknutého zoznamu potom vyškrtnáme všetky  $k$ -tice, v ktorých sa nejaké číslo opakuje. Kódom čísla  $i$  bude  $i$ -ta položka výsledného zoznamu (položky v zozname číslujeme od 1).

ÚLOHA: Pomôžte Ferdovi a napíšte program, ktorý načíta čísla  $n$ ,  $k$  a  $i$  a vypíše kód čísla  $i$ . Snažte sa, aby bol váš program čo najrýchlejší, lebo čísla  $n$ ,  $k$  a  $i$  môžu byť aj pomerne veľké (rozhodne nie je veľmi efektívne vytvárať zoznam všetkých  $k$ -tic z čísel 1 až  $n$ ).

Napríklad pre  $n = 4$ ,  $k = 3$ ,  $i = 3$  je kód je 1 3 2.

### z1634. Zázvorové pivo II

Zázvorové pivo II<sup>TM</sup> sa stále necháva rozvážať závozníkom Zachariášom. Je vyrobené vylepšenou receptúrou a je na to veľmi pyšné. Práve sa necháva predávať v meste A (a mimochodom všetkým výborne chutí), ale chcelo by sa dostať na veľkolepý festival piva do mesta B. Zázvorové pivo II<sup>TM</sup> si brúsi zuby na hlavnú cenu – zlatý krígel a skromne uznáva, že jeho vyhliadky sú ružové. Teda ak sa tam dostane. Má to totiž háčik – zdraželo mýto. Rovnako ako v časoch, keď sa pomocou Zachariáša rozvážalo Zázvorové pivo I, aj teraz treba pri každom vstupe so Zachariášom a vozom do hocikákeho mesta zaplatiť mýto. Ale teraz si každé mesto stanovilo vlastnú výšku mýta. A tak si Zázvorové pivo II<sup>TM</sup> zadumane špliecha a premýšľa, kade ísť, aby cesta vyšla čo najlacnejšie.

ÚLOHA: Na vstupe máte zadané prirodzené číslo  $n > 1$  určujúce počet miest (mestá sú číslované číslami 1 až  $n$ ) a pre každé mesto  $i$  je daná výška mýta  $v_i > 0$  v tomto meste. Ďalej sú zadané čísla miest A a B a číslo  $m$  určujúce počet ciest medzi mestami. Každá z  $m$  ciest je zadaná dvojicou čísel miest, medzi ktorými vedie. Zistite, či sa Zázvorové pivo II<sup>TM</sup> môže dostať z mesta A do mesta B. Ak áno, vypíšte trasu, vedúcu z A do B takú, aby bol súčet mýt v mestách, cez ktoré trasa vedie, najmenší možný. Ak je takýchto trás viac, vypíšte ľubovoľnú.

Príklad:

Vstup:

$n = 7$ ,

mýto po rade: 1, 5, 8, 2, 10, 1, 3

$A = 3$ ,  $B = 4$

$m = 6$

cesty:

1 6

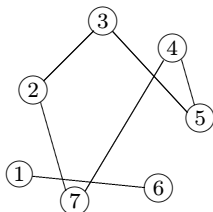
2 3

2 7

4 7

4 5

3 5



Výsledkom je trasa idúca cez mestá 3, 2, 7 a 4, na ktorej treba zaplatiť mýto vo výške  $5 + 3 + 2 = 10$ . Existuje ešte trasa 3, 5, 4, ale na nej treba zaplatiť mýto vo výške 12.

### z1635. Znižovanie zadlženosti

Naše hospodárstvo je vo veľmi zlom stave. Hádam ani nieto podniku, ktorý by nemal nejaké dlhy. A nie je zriedkavý ani prípad, že jeden podnik súčasne niektorým podnikom dlhuje a iné podniky sú naopak jeho dlžníkmi. Ak podnik X dlží podniku Y sumu  $k$  korún, budeme to zapisovať ako  $d(X, Y) = k$ .

Skupina ekonomických expertov vymyslela spôsob ako celkovú zadlženosť znižovať. Navrhujú zaviesť používanie zmluvy o prevode dlhu. Ak pre nejaké podniky X, Y a Z sú dlhy  $d(X, Y)$  a  $d(Y, Z)$  obidva aspoň  $k$  korún, potom podniky X a Y môžu uzavrieť zmluvu o



prevode dlhu veľkosti  $k$ , na základe ktorej sa dlhy  $d(X, Y)$  a  $d(Y, Z)$  obidva znížia o  $k$  korún a naopak dlh  $d(X, Z)$  sa o  $k$  korún zvýši (prípadne vznikne nový dlh). Ak vznikne týmto spôsobom dlh typu  $d(X, X)$ , automaticky zaniká. Takýmito zmluvami medzi podnikmi je možné celkovú zadlženosť postupne znižovať dovtedy, kým už sa žiadna zmluva o prevode dlhu nebude dať uzavrieť. Otázkou zostáva, aký bude výsledok.

ÚLOHA: Napíšte program, ktorý načíta počet podnikov  $n$ , počet dlhov medzi podnikmi  $m$  a jednotlivé dlhy. Každý dlh je zadaný ako trojica čísel  $X, Y$  a  $k$  (kde  $d(X, Y) = k$ ). Podniky sú číslované číslami  $1, 2, \dots, n$ . Váš program má vypísať zoznam dlhov, ktorý je jedným z možných výsledkov používania zmluvy o prevode. Váš zoznam teda musí byť taký, že mohol vzniknúť postupným uzatváraním zmlúv a súčasne už nie je možné uzavrieť žiadnu ďalšiu zmluvu.

Príklad:

Vstup:

$n = 4, m = 4$

Dlhy:

1 2 30

2 3 40

2 4 10

4 1 10

Výstup:

1 2 20

1 3 20

### 1711. O d'Artagnanovi

Keď mladý d'Artagnan dovšil osemnásť rok života, rozhodol sa, že sa vydá do Paríža a stane sa mušketerom. Dostať sa do Paríža však v tej dobe nebolo vôbec jednoduché. Jedinou rozumnou možnosťou bolo vydať sa po kráľovských cestách, spájajúcich jednotlivé mestá. Každá cesta začína aj končí mohutnou mestskou bránou. Problém je, že každú mestskú bránu strážia buď kráľovi mušketeri alebo kardinálovi gardisti a značia si všetkých, čo prišli do mesta. Preto keď človek prišiel do cudzieho mesta bránou stráženou mušketermi, musel z neho aj odísť niektorou bránou stráženou mušketermi – gardisti ho nemajú na zozname a dali by ho zavrieť. A naopak, ak prišiel bránou stráženou gardistami, musel nejakou takou aj odísť. Navyše sa vie, že medzi gardistami a mušketermi vládne nepriateľstvo, preto ak bránu na jednom konci cesty strážia jedni, na druhom konci ju určite strážia druhí. Nečudo, že d'Artagnan už hodiny sedí nad mapou Francúzska a rozmýšľa, ako sa v tejto situácii čo najrýchlejšie dostať do Paríža. (Z rodného mesta môže d'Artagnan odísť hociktorou bránou.)

ÚLOHA: Vo Francúzsku je  $n$  miest číslovaných od 1 po  $n$ . Paríž má číslo 1, d'Artagnanove rodisko má číslo  $n$ . Medzi mestami vedie  $m$  ciest. Všetky cesty sú obojsmerné. O každej ceste vieme číslo jej začiatku a koncového mesta, jej dĺžku ako aj to, akí vojaci strážia jej začiatok. (M – mušketeri, G – gardisti. Koniec pochopiteľne strážia tí druhí.)

Napíšte program, ktorý načíta počet miest  $n$ , počet ciest  $m$  a informácie o jednotlivých cestách a vypíše dĺžku najkratšej trasy z d'Artagnanovho rodiska do Paríža za daných podmienok.

Napríklad pre  $n = 5, m = 7$  a cesty

1 2 3 M

1 4 4 G

1 3 2 G

2 5 3 G

2 4 10 M

5 4 1 G

4 3 1 M

Výstup: Najkratšia cesta má dĺžku 5.

(Je to cesta  $5 \rightarrow 4 \rightarrow 1$ .)

### 1712. Dopravný Podnik Mesta Bratislava

Jedného krásneho dňa sa rozhodli Janko a Marienka so svojimi deťuencami a niekoľkými jazvečikmi ísť do hlavného mesta na výlet. Prišli, ako inak, na hlavnú stanicu a teraz

čakajú na zastávke autobusu. Už asi štvrt hodinu hľadajú do automatu na lístky a špekulujú, aký lístok kúpiť. Janko vraví: „Ja by som kúpil jeden zľavnený lístok + pes, jeden celý a jeden celý + pes.“ „To je hlúposť!“ vraví Marienka. „Bolo by to zbytočne je drahé, lepší by bol jeden celý+zľavnený+pes...“ Ich drahé detúrence na nich nechápavo hľadajú. Aj by svojim rodičom poradili, ale ešte nevedia hovoriť. Preto im musíte pomôcť vy.

ÚLOHA: Napíšte program, ktorý načíta čísla  $a$ ,  $b$ ,  $c$  a  $n$ . Ďalej načíta  $n$  typov cestovných lístkov, pričom každý typ má tvar: „ $x$  dospelých +  $y$  detí +  $z$  psův za cenu  $p$  korún.“ Váš program má vypísať, akú najmenšiu cenu musí zaplatiť za lístky  $a$  dospelých,  $b$  detí a  $c$  psův. (Cestujúci si môžu kúpiť lístky aj pre viac ľudí alebo psův, ak to bude stáť menej.)

Napríklad pre  $a = 2$ ,  $b = 3$ ,  $c = 2$ , Je výstup: 51  
 $n = 6$  a  $n$  riadkov (Sú to lístky za 23+23+5 korún.)

```

1 0 0 10
0 1 0 5
0 0 1 10
1 0 1 19
0 1 1 14
1 1 1 23
```

### 1713. O spartakiáde

Iste viete, že kedysi sa pravidelne usporiadavala spartakiáda. V rok jej konania sa na všetkých školách na telesnej výchove cvičievali rôzne choreografické prvky a najlepší z najlepších žiakov potom predvádzali svoje umenie pred očami prítomných aj televíznych divákov na Pražskom Strahove. Občas to ale úplne nevyšlo a po niektorom náročnom prvku sa cvičenci ocitli vo formácii, v ktorej pôvodne vôbec nemali byť. Našťastie boli vždy na zemi v pravidelných vzdialenostiach nakreslené krúžky, podľa ktorých sa dalo orientovať. Každý z cvičencov vtedy dobehol na najbližší neobsadený krúžok a pokračoval v cvičení. Najväčší problém však mali organizátori so záverom, keď sa všetci zúčastnení mali postaviť vedľa seba a pokloniť sa. Vtedy ich to nenapadlo, ale dnes by s veľkou nádejou čakali na vaše programy.

ÚLOHA: Napíšte program, ktorý načíta počet cvičencov  $n$  a pre každého cvičenca súradnice krúžku, na ktorom stojí. Krúžky sú umiestnené na všetkých miestach s celočíselnými súradnicami. Váš program by mal vypísať počet presunov, ktoré sú potrebné k tomu, aby všetci cvičenci stáli na krúžkoch s rovnakou  $y$ -ovou súradnicou a neboli medzi nimi neobsadené krúžky (t.j.  $x$ -ové súradnice cvičencov budú  $x$ ,  $x+1$ , ...,  $x+n-1$ ). Tento počet má byť najmenší možný.

Cvičenci sa môžu presúvať len z krúžku na krúžok a to tak, že práve jedna zo súradníc nového krúžku sa od príslušnej predošlej súradnice líši presne o 1 (teda môžu ísť na krúžok vľavo, vpravo, dopredu alebo dozadu). V žiadnom momente presúvania by na jednom krúžku nemal stáť viac ako jeden človek.

Napríklad pre  $n = 5$ , súradnice cvičencov:  $(1, 2)$ ,  $(2, 2)$ ,  $(1, 3)$ ,  $(3, -2)$ ,  $(3, 3)$  je minimálny počet presunov 8. Cvičenec so súradnicami  $(1, 3)$  sa presunie na  $(0, 3)$  a odtiaľ na  $(0, 2)$ , cvičenec z pozície  $(3, -2)$  sa presúva cez súradnice  $(3, -1)$ ,  $(3, 0)$ ,  $(3, 1)$  na  $(3, 2)$  a cvičenec z pozície  $(3, 3)$  sa presunie cez  $(4, 3)$  na  $(4, 2)$ .

### 1714. O vekslákoch

Je také miesto, kde ľudia nakupujú peniaze za peniaze, marky za doláre, koruny za jeny, jeny za franky, libry za drachmy a všakovaké iné za všakovakejšie inšie. Motá sa tam mnoho pochybných existencií, ktoré sledujú iba svoje malé, mnohokrát iracionálne ciele. Volajú ich veksláci. Keď má človek šťastie a dostatok informácií, môže prísť na takomto mieste k slušnému majetku tak, že si donesie peniaze v nejakej mene, vymieňa ich, vymieňa, až má opäť peniaze v tej istej mene, ale je ich viac. To sa nie vždy dá dosiahnuť.

ÚLOHA: Zistite, či sa to dá dosiahnuť, ak máte daný zoznam kurzov, za ktoré veksláci menia peniaze. Ak sa to dá, tak uveďte aj postupnosť výmen s príslušnými kurzami. Predpokladajte, že môžete presne zameniť ľubovoľné množstvo danej meny. Uvedomte si však, že ak vekslák mení koruny na doláre, nemusí to robiť naopak v rovnakom kurze.

Príklad

Vstup:

SKK 44.23 na USD 1

USD 1.3 na GBP 1.08

GBP 1.18 na SKK 65.993

USD 0.02 na SKK 1

SKK 50 na USD 1

Výstup:

áno: SKK 44.23 na SKK 50

(SKK  $\xrightarrow{44.23;1}$  USD  $\xrightarrow{0.02;1}$  SKK)

### 1715. Hra s pešiakmi

Za starých dobrých čias trávili Janko s Marienkou večery hraním hry Boxes. Tá ich už však dávno omrzela, tak teraz skúšajú všelijaké nové hry, napríklad hru s pešiakmi.

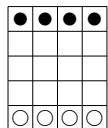
Táto hra sa hrá na šachovnici s  $m$  riadkami a  $n$  stĺpcami. Jeden hráč má  $n$  bielych pešakov a druhý hráč má  $n$  čiernych pešakov. Na začiatku hry sú biele figúrky rozostavené v spodnom riadku šachovnice a čierne v hornom. Hráč, ktorý je na ťahu, môže potiahnuť jednu zo svojich figúrok o ľubovoľný počet políček smerom dolu alebo hore, pričom nesmie preskočiť súperovu figúrku ani vyjsť za okraj šachovnice. Hráči sa pri ťahoch striedajú, pričom začína hráč s bielymi figúrkami. Prehráva hráč, ktorý už nemôže urobiť žiadny ťah.

ÚLOHA:

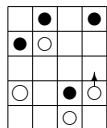
a) Napíšte program, ktorý bude hrať túto hru proti Marienke, keď na ňu Janko nemá čas. Program najprv načíta  $n$  a  $m$  a spýta sa užívateľa, akú farbu figúrok chce mať. Potom váš program striedavo ťahá a pýta si ťah od užívateľa. Sústreďte sa na to, aby váš program hral hru čo najlepšie.

b) Marienka si pamätá mnoho rozohraných hier, ktoré by chcela dohrať. Napíšte preto program, ktorý načíta  $n$ ,  $m$ , farbu užívateľových figúrok a farbu figúrok hráča, ktorý je práve na ťahu. Potom načíta situáciu hry (t.j. pre každý stĺpec šachovnice pozíciu bieleho a čierneho pešiaka) a dohrá túto hru proti užívateľovi. Opäť sa sústreďte na to, aby váš program hral hru čo najlepšie.

Príklad pre  $m = 5$  a  $n = 4$ .



Obr. 1



Obr. 2

Na obrázku 1 je počiatočná pozícia hry. Na obrázku 2 je možná situácia počas hry. Jeden z možných ťahov bieleho hráča (ak je práve na ťahu) je napr. o jeden krok dopredu figúrkou v najpravejšom stĺpci.

### 1721. O krízovom štábe

Mimozemšťan Martow sa práve vrátil z diplomatickej misie na vzdialenej planéte s názvom Zem. Nedočkavo vykukol z okienka svojej vesmírnej lode. Ako však čoskoro zistil, na jeho domácej planéte zďaleka nebolo všetko tak, ako malo byť...

Na Martowovej planéte je  $n$  miest. Medzi každými dvomi mestami bolo vybudované obojsmerné teleportové spojenie. Z neznámych príčin sa však všetky teleports naraz pokazili a teraz každý z nich funguje len jedným smerom. Mimozemšťania sa rozhodli situáciu riešiť ustanovením krízového štábu. Každé mesto bude mať v štábe svojho zástupcu. Štáb

sa musí stretnúť čo najskôr. Za miesto stretnutia je teda nutné zvoliť také mesto  $M$ , aby sa člen, ktorému bude cesta trvať najdlhšie, dostal do mesta  $M$  čo najskôr, t.j. na najmenší možný počet použití teleportov. Inými slovami, vzdialenosť (počítaná v počte použitých teleportov) z najvzdialenejšieho mesta do mesta  $M$  musí byť najmenšia možná.

ÚLOHA: Napíšte program, ktorý načíta počet miest  $n$  a popis jednotlivých teleportov a vypíše číslo mesta  $M$ , v ktorom má zasadiť krízový štáb. Popis teleportov má nasledujúci formát: pre každé mesto  $i$  je daný počet teleportov  $p[i]$ , ktoré vedú z tohto mesta. Ďalej je daný zoznam  $p[i]$  miest, do ktorých vedú teleporty z mesta  $i$ . Môžete predpokladať, že v zozname sa z dvojice  $i \rightarrow j$  a  $j \rightarrow i$  vyskytne práve jeden teleport pre každé  $i, j, i \neq j$ . Ak je vyhovujúcich miest na stretnutie štábu viac, vypíšte ľubovoľné z nich.

Súčasťou riešenia by malo byť aj zdôvodnenie jeho správnosti.

Príklad:

Vstup:

$n = 5$

4 2 3 4 5

3 3 4 5

1 4

1 5

1 3

Výstup:

3

(Z mesta č. 4 sa sem dá dostať pomocou dvoch teleportov, z miest č. 1, 2 a 5 pomocou jedného teleportu.)

## 1722. O telocviku

Riško, Vladko, Dávidko, ako poriadni žiaci, pravidelne chodia na telocvik. Radi naň chodia, môžu sa tam veselo zahrať s loptou spolu s ich telocvikárom – ujom Mariánom. Ujo Marián je príjemný bradatý pán, má ale jednu chybu – rád dáva za trest svojim žiakom klikovať používajúc pri tom s obľubou frázu: „Daj si dvadsať!“

Jedného dňa prišli všetci, celá trieda, neskoro na telocvik. Vbehli spolu do telocvične, ale beda! Zježená brada uja Mariána neveštala nič dobrého. Tak sa žiaci rýchlo postavili do radu, ako to zvyknú robiť vždy na začiatku hodiny. Lenže rad sa akosi ujoini Mariánovi nepáčil. „Musím im dať za trest klikovať. Koľkože klikov?“, hŕta si. „Nestoja pekne podľa výšky, nestihli sa v tom zhone usporiadať.“ Začne rátať, koľko dvojíc žiakov je navzájom vymenených (dvojica je vymenená, ak vyšší stojí pred nižším). Tolko klikov dostane každý z nich za trest. Ale dajako mu to v tom zhone nejde. Pomôžte mu!

ÚLOHA: Na vstupe je daný počet žiakov  $n$ . Žiaci majú výšky celé čísla 1 až  $n$ . Každá výška sa vyskytuje v triede práve raz. Ďalej sú na vstupe dané výšky žiakov  $a_1, a_2, \dots, a_n$  v tom poradí, v akom stoja v rade. Zrátajte počet všetkých „vymenených“ dvojíc v rade, t.j. takých dvojíc  $a_i, a_j$  ( $i < j$ ), že  $a_i > a_j$ .

Napríklad pre  $n = 5$  a 3 2 4 1 5 je výsledok 4. Sú to dvojice: (3, 2), (3, 1), (2, 1), (4, 1).

## 1723. O lyžiaroch

Mt. Kopec je vychýrené lyžiarske stredisko. Je na ňom množstvo svahov ideálnych na lyžovanie, kratších, dlhších, strmších i menej strmých. A čo je najlepšie, na hornom aj dolnom konci každého svahu je horská chata, v ktorej si lyžiari môžu kúpiť teplý čaj s rumom, párky, alebo aspoň horalku. Jediné, čo svahom na Mt. Kopci chýba, sú lanovky. Je totiž veľmi nepríjemné, keď si lyžiar po zidení svahu musí dať lyže na plece a šliapať naspäť do kopca pešo. Preto sa lyžiari (a samozrejme chatári, ktorým to prinesie zisk) rozhodli postaviť sústavu lanoviek tak, aby sa postupnosťou vyvezení lanovkou a zidení svahov dalo dostať na ľubovoľný svah (a do ľubovoľnej chaty, samozrejme). Aby ušetrili, rozhodli sa stavať len jednosmerné lanovky a prirodzene, chceli by ich postaviť čo najmenej. Nevedia však ako na to a boli by radi, keby ste im pomohli.

ÚLOHA: Na Mt. Kopec je  $n$  chát očíslovaných 1 až  $n$ . Medzi nimi vedie  $m$  svahov, každý svah vedie medzi dvoma chatami. Zjazdovať sa po ňom dá iba od vyššie položenej chaty k nižšie položenej. To znamená, že ak sa dá nejakou postupnosťou svahov dojazdovať od

chaty  $i$  po chatu  $j$ , určíte sa nedá dozjazdovať od chaty  $j$  po chatu  $i$ . Napíšte program, ktorý načíta čísla  $M$  a  $n$  a zoznam svahov (každý svah je určený dvojicou vyššie položená chata, nižšie položená chata) a vypíše, medzi ktorými dvojicami chát je potrebné postaviť jednosmernú lanovku, aby sa postupnosťou vyvezení lanovkou a spustení sa po svahu dalo dostať z každej chaty do každej inej. Počet postavených lanoviek má byť minimálny.

Napríklad pre  $n = 4$ ,  $m = 4$  a svahy: z 1 do 2, z 1 do 3, z 2 do 3, z 2 do 4 je treba pridať lanovky z 3 do 1 a zo 4 do 1. (Iné riešenie: z 3 do 2 a zo 4 do 1.)

#### 1724. O prederavenej streche

Janko, Marienka, ich deťúrence a všetci jazvečíci po dlhých výpočtoch lacno docestovali do perníkovej chalúčky hlboko v tmavom lese. Deti s jazvečíkmi zatiaľ veľmi vyhľadli a živelne sa pustili do sladučkej chrumkavej strechy. V streche tak žiaľ vznikli diery, ktorými fúkala chladná nepríjemná jeseň. Marienka sa rozhodla napiecť perníky, ktorými by všetky diery poplátala. Zamiesila cesto na perníky, avšak zistila, že v chalúpke nemajú žiadny válok. Chce preto poslať Janka do mesta nejaký kúpiť, len nevie, aký široký potrebuje. Keďže Janko je veľmi sporivý, chce kúpiť čo najlacnejší (a teda najužší).

ÚLOHA: Marienka chce vyvalkať obdĺžnikový pás cesta, z ktorého potom bude vykrajovať jednotlivé záplaty. Pás môže byť ľubovoľne dlhý, avšak môže byť široký najviac tak ako válok. Pre každú záplatu (konvexný  $n$ -uholník), ktorý musí vykrojiť, je potrebné spočítať, z akého najužšieho pásu cesta sa dá vyrezať. Záplatu a cesto je možné voči sebe navzájom ľubovoľne otáčať a posúvať.

Na vstupe je číslo  $n \geq 3$  a súradnice  $N$  bodov  $x_i, y_i$  ( $i = 1 \dots n$ ) konvexného  $n$ -uholníka. Napíšte program, ktorý zistí, aký najmenej široký pás cesta je nutné vyvalkať, aby sa z neho dal vyrezať daný  $n$ -uholník. Snažte sa nájsť čo najrýchlejšie riešenie, lebo sa blíži búrka a...

Napríklad pre  $n = 4$  a  $(1, 1.5)$ ,  $(2, 0)$ ,  $(2, 10)$ ,  $(0.5, 3)$  je výsledok 1.5. Okraje pásu tvoria priamky  $[2, 0][2, 10]$  a  $[0.5, 0][0.5, 3]$ .

#### 1725. O úbohom kráľovičovi

Kde bolo, tam bolo, niekde uprostred lesa medzi piatym a šiestym kopcom bolo maličké kráľovstvo. V tomto kráľovstve vládol mladý kráľovič Richard. Keďže mu bolo na tróne smutno, rozhodol sa zaobstarať si nejakú rúču manželku. Vybral sa teda do sveta, chodil, blúdil, až navela prišiel do kráľovstva mocného kráľa, ktorý mal zhodou okolností jedinou dcéru Pamelu, akurát súcu na vydaj. Richard ju hneď požiadal o ruku. Keď však zistil, aká to je škrtata, pokúsil sa od svojej ponuky odstúpiť. Nie veľmi úspešne...

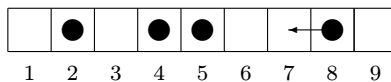
O pár dní neskôr. Na hladomorni sa otvorili dvere. Dohnútra vstúpil kat a kráľ. „Tak ti dám ešte jednu šancu“ prehovoril kráľ. „Budeme hrať. Ak vyhráš, tak ťa pustím domov, ak nie, ech, tak si vezmeš moju dcéru Pamelu za ženu a odídeš s ňou vládnuť. Čo na to vravíš?“ Čo už mal náš úbohý kráľovič robiť, neostalo mu nič iné ako súhlasiť.

Kráľ vytiahol z vrečka hrací plán – dlhočizný úzky pás látky, na ktorom bolo  $n$  (veľmi veľa) políčok vedľa seba očíslovaných od 1 po  $n$ . Potom na štyri rôzne políčka položil štyri kamienky. Hráč, ktorý je na ťahu, môže zobrať jeden kamienok a posunúť ho o niekoľko (aspoň jedno) políčok smerom k začiatku plánu (t.j. na políčko s menším číslom), pričom nesmie preskočiť žiaden iný kamienok. V žiadnom okamihu nesmie byť na žiadnom políčku viac ako jeden kamienok. Ten kto nemôže ťahať, ani keby sa potrhal, tak ten prehral. Kráľ ako starý gavalier ponúkol Richardovi prvý ťah.

Keď úbohý Richard podumal nad hrou, zľakol sa, že sa mu nepodarí vyhrať a zbledol. Musíte mu preto pomôcť.

ÚLOHA: Napíšte program, ktorý načíta pozíciu štyroch kamienkov na začiatku hry. Potom bude striedavo vypisovať svoje a načítavať kráľove ťahy až do konca hry. Snažte sa, aby váš program hral čo najlepšie.

Pozícia počas hry a možný Richardov ťah:



### 1731. O veľkom smäde

„Stavme sa, že vypijem 10 kofol, ak mi ich niekto kúpi,“ navrhne Mišo. „Dobre, ale ak prehráš, tak ty mne kúpiš 15 kofol! Nech sa mi vrátia aj s úrokmi,“ súhlasí Braňo. „Platí?“ „Platí.“ A tak sa poberú k výčapu. Prvá, druhá, tretia kofola... to je ešte v pohode. Štvrtá, piata, šiesta... začína byť problém. Siedma, ôsma... Mišo je už zeleno-modrý a vzdáva sa. „Prehral si! Teda mi už visíš 43 kofol.“

A takto to v KSPáckych kruhoch chodí už dlhé týždne. Každý je niekomu dlžný množstvo kofol. Pomaly ale iste sa v zložitých dlžobných vzťahoch už prestávajú všetci vyznať. Preto sa rozhodli, že sa dnes vyrovnajú. Lenže, keď vypijú takých 50 kofol v takejto zime, to im hrozí podchladenie, ba až smrť. A čo by bolo potom?

Vtom skrsla Dávidovi v hlave skvostná idea (tzv. bezkofolové vyrovnanie):

Nech dlhuje KSPák A KSPákovi B  $k_1$  kofol a B dlhuje C  $k_2$  kofol. Potom môžu svoje dlhy upraviť tak, aby B vypil o  $x = \min(k_1, k_2)$  kofol menej. Potom A bude dlžiť B iba  $k_1 - x$  a B bude dlžiť C  $k_2 - x$  kofol. Zároveň však bude KSPák A dlžiť KSPákovi C o  $x$  kofol viac. Ak sa takto stane, že niekto dlží niečo sám sebe ( $A = C$ ), tak si nebude samozrejme nič kupovať.

ÚLOHA: V prvom riadku vstupu sú dve celé čísla  $n$  – počet KSPákov a  $m$  – počet dlžôb. Potom nasleduje  $m$  riadkov. Na každom sú tri celé kladné čísla  $A, B, k$ , čo znamená: KSPák A je dlžný KSPákovi B práve  $k$  kofol. Vašou úlohou napísať program, ktorý zistí, koľko minimálne kofol musia dokopy vypiť a vypíše jeden možný záverečný stav, kto komu koľko kofol dlží.

Príklad:

Vstup:

```
3 4
1 2 4
2 3 10
3 1 3
3 2 2
```

Výstup:

```
Počet vypitých kofol: 5
1 3 1
2 3 4
```

### 1732. O automate na lístky

Dano študuje už dlhé roky programovanie na univerzite kdesi v Amerike. Nedávno prišiel na návštevu Bratislavy. Aké bolo jeho prekvapenie, keď zistil, že v Bratislave už dávno neplatia staré lístky na autobus (ktorých mal pre každý prípad niekoľko v zálohe).

Nové lístky sa dajú kúpiť napríklad v automate na zastávke. Automat vydáva  $n$  druhov lístkov s cenami  $c_1, c_2, \dots, c_n$  (celé čísla). Lístky sa kupujú tak, že lístkovčtivá osoba postupne stláča tlačidlá zodpovedajúce lístkom, ktoré chce kúpiť (ak chce viac lístkov rovnakého druhu, stlačí príslušné tlačidlo viackrát) a na displeji automatu sa priebežne objavuje ich celková cena (na začiatku displej ukazuje 0 korún). Potom dotýčny vhodí do automatu príslušnú sumu mincí a z automatu vypadnú žiadané lístky.

Dano potreboval veľa rôznych lístkov (plánuje sa nejaký čas zdržať, navštíviť rodičov, priateľov, jednu nemenovanú slečnu, matematicko-fyzikálnu fakultu a zubára) a tak začal odušu stláčať tlačidlá na automate. Ako tak stláčal, zistil, že automat nie je ochotný akceptovať viac ako  $p$  stlačení tlačidiel a nie je ochotný akceptovať ďalšie stlačenie, ak je na displeji suma prevyšujúca  $s$  korún. Dano sa ako správny programátor rozhodol, že zistí, akú najväčšiu sumu je možné na displeji automatu stláčaním tlačidiel dosiahnuť. Potom si však uvedomil, že ak chce stihnúť všetky návštevy, musí sa poponáhľať a nechal túto úlohu na vás.

ÚLOHA: Napíšte program, ktorý načíta počet lístkov, ich ceny, maximálny počet stlačení a sumu  $s$  a vypíše, akú najväčšiu sumu je na displeji možné dosiahnuť.

Napríklad pre  $n = 3$ , ceny: 6, 17, 35,  $p = 10$  a  $s = 100$  je výsledok 134.

(Treba stlačiť tlačidlá pre dva lístky za 35, jeden za 17, dva za 6 a nakoniec ešte jeden za 35 korún.)

### 1733. O veľkom neporiadku

„Keď mi už tak veľmi chceš pomáhať, tak roztrieď tamtie skrutky!“, povedal otec ukazujúc na veľkú kopu skrutiek a matic v rohu garáže. Tomáš bol celý nešťastný. Rád otcovi vo všetkom pomáhal. Nemohol za to, že všetko, čoho sa dotkol, sa hneď lámalo a kazilo. Otec sa väčšinou nezlostil, aj keď často musel robiť všetko odznova. Tentoraz to ale vyzeralo vážne. Výraz v otcovej tvári hovoril jednoznačne: „Už žiadne kazenie, prekážanie a práca navyše. Teraz ťa zamestnám prácou, pri ktorej nič nepokazíš.“ Tomáš teda smutne pozrel na kopu v rohu a pustil sa do práce. „Presne ako Popoluška,“ pomyslel si. „Popoluške prišli pomôcť holuby. Lenže mne nikto nepomôže. Ibaže by...“

ÚLOHA: Máme  $n$  skrutiek a  $n$  matic rôznych veľkostí, z ktorých ku každej skrutke prislúcha práve jedna matica a naopak. Napíšte program, ktorý pomože Tomášovi usporiadať skrutky a matice. Program načíta počet skrutiek a matic  $n$  a potom môže klásť Tomášovi otázky typu „ $i$ -ta skrutka a  $j$ -ta matica?“. Tomáš odpovie buď „OK“ ak skrutka a matica patria k sebe, „VÄČŠIA MATICA“ ak  $j$ -ta matica je väčšia ako  $i$ -ta skrutka a „VÄČŠIA SKRUTKA“ ak je skrutka väčšia ako matica.

Úlohou vášho programu je nájsť ku každej skrutke jej zodpovedajúcu maticu. Tomáš chce mať túto robotu rýchlo hotovú, preto by ocenil, keby sa ho program pýtal čo najmenej otázok, avšak dôležitejšie je, aby mu program vyhodil správny výsledok.

Napríklad pre  $n = 3$  a otázky s odpoveďami: 1. skrutka a 2. matica? OK a 2. skrutka a 1. matica? VÄČŠIA MATICA sú správne páry: (1,2), (2,3), (3,1).

### 1734. O dlhých tyčiach

Stál Babylon, najslávnejšia to ríša. I jedného dňa povedali si Babylončania: „Nám už žiť na zemi neprináleží. Až po samé nebo vežu postavíme, a potom tam žiť budeme!“. Nedbali oni varovania veštca, že kto sa Bohu rovnať chce, kruto potrestaný bude. Každý z Babylončanov sa domov pobral, i všetky tyče, čo doma našiel, na námestie doniesol. Keď už všetky tyče pokope boli, začali z nich konštrukciu veže stavať. Lenže čo to? Tyče boli tak rôznych dĺžok, že nik, ani ten najmúdrejší z nich, nebol schopný nájsť tri také, čo by strany trojuholníka tvorili. A tak hľadali, hľadali, no stále nenachádzali a stavbu začať nemohli.

A tak by sa im zišiel program, ktorý by tento problém riešil za nich. (A tiež by sa im zišiel počítač...)

ÚLOHA: Napíšte program, ktorý dostane na vstupe číslo  $n$  a následne  $n$  dĺžok tyčí  $d_1 \dots d_n$ , čo sú reálne čísla, pre ktoré platí  $1 \leq d_i \leq 1000000000$ . Program má vypísať, či je medzi nimi taká trojica, z ktorej sa dá zložiť trojuholník.

Dobre sa zamyslite nad časovou a pamäťovou zložitosťou vášho programu.

Napríklad pre dĺžky tyčí 3, 1.0, 1.1 a 2.1 program vypíše NIE a pre dĺžky 5, 1.7, 4.9, 51.0, 174.9, a 4.957 vypíše ÁNO

### 1735. O Santovi a fľašiach II

Zlatokopi z Vyšnej Klondiky už po sedem rokov radi chodievajú do hostinca v Dolnom Kelčove posilňovať sa ohnivou vodou, hrať hazardné hry a uzatvárať stávky. Aj Santo a Banto tam pravidelne radi chodia. Tentokrát sa im opäť podarilo vyhrať stávku s krčmárom a za odmenu im krčmár postavil na stôl do kruhu mnoho fliaš ohnivej vody rôzneho objemu a povedal: „Z týchto fliaš pite koľko vám hrdlo ráči, ale dodržujte kultúru pitia!“

Santo sa už-úž chcel rozbehnúť ku fľašiam, ale Banto ho zadržal. „Nezabúdaj na tie prihlúple krčmárove pravidlá, nemôžeme piť hocijako, nevypité fľaše musia v každom okamihu tvoriť súvislý úsek. Musíme si dávať pozor, ináč nás opäť vyhodí! A budeme sa pekne striedať. Jednu fľašu ty, jednu ja, až kým sa nám všetky neminú.“

ÚLOHA: Na začiatku je na stole  $2n$  fliaš s objemami 1 až  $2n$  decilitrov (každý objem sa vyskytuje práve raz) rozostavených do kruhu. Hráč, ktorý je prvý na ťahu, môže vypíť ľubovoľnú fľašu. V každom ďalšom ťahu môže príslušný hráč vypíť ľubovoľnú fľašu susediacu s nejakou už vypitou fľašou (aby sa zachovala súvislosť úseku nevypitých fliaš). Cieľom hry je samozrejme preliať hrdlom viac ohnivej vody ako ten druhý.

- Je dané  $n$  a  $2n$  rôznych celých čísel od 1 po  $2n$  reprezentujúcich objemy fliaš po rade v kruhu. Napíšte program, ktorý bude radiť Santovi (ktorý je prvý na ťahu), ktorú fľašu má kedy vypíť tak aby dokopy vypil viac ohnivej vody ako Banto. Váš program by mal v každom ťahu vypísať, ktorú fľašu má Santo vypíť a načítať, ktorú fľašu vypil Banto. Môžete predpokladať, že Banto hrá podľa pravidiel.
- Napíšte program, ktorý načíta ľubovoľnú legálnu pozíciu (t.j. takú, ktorá mohla vzniknúť postupnosťou legálnych ťahov), a bude striedavo radiť hráčovi na ťahu ktorú fľašu má vypíť a načítavať ťahy súpera.

Na konci by mal program vypísať, kto koľko vypil a kto vypil viac.

Príklad priebehu hry:

- Pre  $n = 3$  a fľaše: 2 6 3 1 5 4  

Vstup:	Výstup:
Santo, vypi 6dl fľašu.	Banto vypil: 3dl
Santo, vypi 2dl fľašu.	Banto vypil: 4dl
Santo, vypi 5dl fľašu.	Banto vypil: 1dl

 Dokopy Santo vypil 13dl ohnivej vody, ale Banto len 8. Chudák Banto.
- Pre  $n = 3$  a fľaše: 0 6 3 1 0 0 (prázdne fľaše označené nulou)  

Santo už vypil: 6	
Banto už vypil: 5	
Na ťahu je: Banto	
Vstup:	Výstup:
Banto, vypi 6dl fľašu.	Santo vypil: 3dl
Banto, vypi 1dl fľašu.	

 Dokopy Santo vypil 9dl ohnivej vody, ale Banto až 12. Chudák Santo.

#### 1741. O vojenských zátarasoch

Kiribatské kráľovstvo začalo vojnu s Kráľovstvom Zimbabwe. Vraj hackeri z Kiribati napadli WWW stránky na oficiálnom kráľovskom serveri v Zimbabwe. Nevedno, čo je na tom pravdy. Možno to bola len zámienka. Ale v tejto chvíli už Zimbawejskí vojaci začali pochod na Kiribati. Na pokyn kiribatského kráľa zasadli vojenski stratégovia a taktici a začali organizovať obranu.

Medzi Kiribati a Zimbabwe vedie sieť ciest. Každá cesta má danú svoju šírku. Aby kiribatské vojsko úspešne obránilo svoju krajinu pred nepriateľom musia postaviť krížom cez celú šírku niektorých ciest zátarasov. Zistite, koľko najmenej metrov zátarasov musí kiribatská armáda postaviť, a ktoré cesty pritom zatarasíť aby si mohla byť istá, že Zimbawejskí vojaci nevstúpia do ich vlasti.

ÚLOHA: Je daný počet uzlov  $n$  cestnej siete medzi Kiribati a Zimbabwe. Uzol číslo 1 sú Kiribati a uzol číslo  $n$  je Zimbabwe. Ďalej je daný počet ciest  $m$  spájajúcich uzly. A nakoniec je na vstupe  $m$  trojíc celých čísel  $x, y, s$  popisujúcich jednotlivé cesty. Trojica  $x, y, s$  ( $1 \leq x, y \leq n$ ) znamená, že cesta vedie z uzla  $x$  do uzla  $y$  a jej šírka je  $s$  metrov. Cesty sú obojsmerné. Vašou úlohou je zistiť, ktoré cesty treba zatarasíť, aby sa postavilo, čo najmenej metrov zátarasov a pritom neexistovala cesta, po ktorej by sa mohli Zimbawejskí vojaci dostať do Kiribatského Kráľovstva. Vypíšte aj celkovú dĺžku zátarasov. Ak je možnosť ako cesty zatarasíť viac vypíšte ľubovoľnú z nich.



Příklad:

Vstup:

$n = 4$ ,  $m = 5$

1 2 8

1 3 3

2 3 3

2 4 3

3 4 5

Výstup:

Treba zatarasiť cesty (1,3), (2,3) a (2,4).

Celková dĺžka zátarasov bude 9 metrov.

#### 1742. O Kiribatských ponorkách

Blíži sa leto a s ním aj potápačská sezóna. Cítiť to aj v Kiribatskom Spolku Potápačov, kde sú prípravy na ňu v plnom prúde. Na zvýšenie bezpečnosti potápania treba podrobne preskúmaťorské priekopy medzi Kiribatskými ostrovmi, a tak sa členovia spolku rozhodli zadovážiť si novú ponorku.

Všetky ponorky dostupné na kiribatskom trhu sú dvojrozmerné a pozostávajú z rovnako veľkých štvorcových segmentov. Dva segmenty jednej ponorky buď nemajú žiaden prienik, alebo majú spoločnú celú jednu stranu. Každá ponorka pozostáva z jedného kusa, teda z ľubovoľného jej segmentu sa dá do ľubovoľného iného prejsť cez niekoľko ďalších jej segmentov. Navyše platí, že keď zoberieme dva segmenty ponorky, ktoré sú v rovnakej hĺbke, a spojíme ich stredy myslenou čiarou, ani jeden bod tejto úsečky neleží mimo ponorky.

Morská priekopa má tvar obdĺžnika, ktorý pozostáva z  $m \times n$  štvorcových segmentov rovnakej veľkosti ako segmenty ponorky. Niektoré segmenty priekopy sú zarastené koralmi, ktoré prekážajú pri jej prieskume. Horná strana obdĺžnika predstavuje morskú hladinu.

Ponorka sa na začiatku potápania nachádza na hladine (t.j. žiadna jej časť nie je pod hladinou). Počas celého potápania musia byť horné strany jej segmentov rovnobežné s hladinou. Posádka môže počas zostupu ponorku posúvať doprava, doľava alebo dole. V žiadnom okamihu potápania sa však žiaden jej segment nesmie nachádzať na mieste zarastenom koralmi.

Členovia Kiribatského Spolku Potápačov teraz smutne sedia nad katalógom predávaných ponoriek a mapou morskej priekopy, v ktorej je zakreslené, ktoré jej segmenty sú zarastené koralmi. Dumajú, dumajú, ale vydumať nevedia, ktorú z ponoriek kúpiť, aby sa mohli ponoriť čo najhlbšie. Pomôžte im a napíšte program, ktorý im túto ťažkú úlohu pomôže vyriešiť.

ÚLOHA: Sú dané čísla  $m$  a  $n$  a mapa morskej priekopy s rozmermi  $m \times n$ . Ďalej sú dané čísla  $a$ ,  $b$  a tvar ponorky s rozmermi  $a \times b$ . Napíšte program, ktorý zistí, ako hlboko sa môže daná ponorka do danej priekopy ponoriť.

Příklad:

Vstup:

7 8            2 2

..#####    .#

..#....    ##

.....

.....

..#....

..#.#..#

...#..

...#..

Výstup:

Ponorka sa môže ponoriť do hĺbky 6.

#### 1743. O Veľkej cene Formule 1

Táto sezóna bola pre Schumachera veľmi zlá. Niektoré preteky síce vyhral on, dosť veľa ich ale vyhral aj Häkkinen. Našťastie má vo vedení pretekov Formula 1 priateľa, ktorý

mu je čo-to vďačný. Taký vďačný, že mu je ochotný za každú cenu pomôcť. Avšak jediné, čoho je schopný, je iniciovať odhlasovanie zmeny bodovania. A tak teraz Schumacher sedí nad výsledkovými listinami a dumá, či sa dá vymyslieť bodovanie, pomocou ktorého by získal titul majstra sveta.

ÚLOHA: Napíšte program, ktorý dostane na vstupe výsledkové listiny  $n$  pretekov a rozhodne, či existuje také obodovanie prvého až tretieho miesta, pri ktorom by mohol Schumacher vyhrať.

Na vstupe je počet pretekov  $n$ , a tabuľka, koľkokrát sa ktorý pretekár umiestnil na ktorom mieste. Výstupom bude **SCHUMACHER MOŽE VYHRAŤ**, ak existuje také obodovanie prvých troch miest  $p_1 \geq p_2 \geq p_3 \geq 1$ , pri ktorom by Schumacher vyhral, to znamená, že by mal najviac bodov zo všetkých.

Napríklad pre  $n = 5$  a

Meno	1. miesto	2.miesto	3.miesto
Häkkinen	4	0	1
Schumacher	1	4	0
Hill	0	0	2
Irvine	0	1	1
Davidko	0	0	0

Je výsledok: **SCHUMACHER MOŽE VYHRAŤ** (pri bodovaní  $p_1 = 10$ ,  $p_2 = 9$ ,  $p_3 = 5$ )

#### 1744. O meteorológovi Ferdovi

Ferdo ako čerstvo vyštudovaný meteorológ nastúpil do práce do meteorologického ústavu. Veľmi sa tešil, že si bude môcť zapredpovedať počasie do ľubovôle, ba že ho možno občas pustia aj vystupovať do televíznych správ o počasí a teraz toto... Ako nového zamestnanca ho totiž previelili na oddelenie štatistiky. Ferdo teraz smutne sedí nad stĺpcami záznamov o teplotách nameraných každý deň od vzniku meteorologického ústavu. Nadriadení mu totiž dali za úlohu zistiť, kedy bolo najteplejšie obdobie. Obdobie je ľubovoľný súvislý úsek aspoň  $k$  dní. Teplotou obdobia rozumieme priemernú teplotu za jednotlivé dni. Pomôžte chudákovi Ferdovi, bezmocne sediacemu nad stohom záznamov, lebo inak skolabuje a budete ho musieť kriesiť.

ÚLOHA: Napíšte program, ktorý načíta počet dní existencie ústavu  $n$ , minimálnu dĺžku obdobia  $k$ , teploty  $t_1, t_2, \dots, t_n$  (reálne čísla) namerané v jednotlivých sledovaných dňoch a vypíše prvý a posledný deň obdobia, ktoré je dlhé aspoň  $k$  dní a má pritom maximálnu možnú priemernú teplotu.

Napríklad pre  $n = 8$ ,  $k = 3$ , teploty:  $-1.9, 4.5, -3.2, 0.5, -4.5, 5.8, -1.6, -2.7$  bolo najteplejšie v období od 2. po 6. deň. Priemerná teplota bola 0.62.

#### 1745. O nových zlatokopoch

Opäť prišla vlna zlatej horúčky, a tak aj na Vyšnú Klondiku dorazila skupinka nových zlatokopov. Vystúpili z vláčiku, ktorý ich doviezol do Puerto Paža, na okraj civilizácie, poobzerali sa a... Ktovie či šťastnou, ale určite náhodou natrafili práve na Santa a Banta. „Ehm... Prosím vás, neviete nám povedať, ako sa dostaneme do Dolného Kelčova?“ „To musíte najskôr do Nalomenej Triesky.“ zahlásil suverénne Santo. „To musíte najskôr do Naštiepeného Íveru.“ zahlásil naraz s ním nemenej suverénne Banto. „Viete čo, veď my sa tam vlastne tiež vraciame... Odvedieme vás tam. Samozrejme, nebude to zadarmo...“

Cesta do Dolného Kelčova je úsečka, na nej leží  $n + 1$  miest, očíslovaných od 0 (Dolný Kelčov) po  $n$  (Puerto Paža, kde práve sú). Od poslednej reformy VHDD (Vyšnoklondická hromadná dostavníková doprava) však dostavníky premávajú zvlášť. Presnejšie, z mesta  $A$  do mesta  $B$  ide dostavník len ak  $A > B$  a  $A - B$  nie je zložené číslo. (Zložené číslo je prirodzené číslo, ktoré má iného deliteľa ako 1 a seba.) Uznáte, že za týchto okolností nie je vôbec ľahké niekam sa dostať. Preto niet divu, že sa Santo a Banto stavili – budú na

striedačku vyberať, do ktorého mesta sa ide. Kto dovedie skupinku do Dolného Kelčova, vyhráva a dostane všetky peniaze, ktoré im za sprievod greenhorni zaplatia.

ÚLOHA: Napíšte program, ktorý načíta číslo  $n$  a následne bude hrať túto hru za začínajúceho zlatokopa.

Priklad priebehu hry, keď  $n = 11$ :

Santo: Poďme do mesta 8.

Banto: Teraz poďme do mesta 3.

Santo: A teraz do mesta 0.

Santo vyhral.

#### z1711. Z učtárne

Predstavte si stredne veľkú fabriku spoločnosti Vidličky a Nože. Nevysoká budova v príjemnom prostredí, všade však vládne čulý ruch. Odlievači, kováči, brusiči, leštiči, ohýbači (neverili by ste, koľko to dá roboty správne vytvarovať zuby na vidličke), zlatíči, pracovníci na obchodnom oddelení ( – Samozrejme, vážený pane, ten nôž je z čistého zlata. Azda si o nás nemyslíte, že by sme vám boli schopní predať nejakú pozlátenú napodobeninu? – ), tetušky v baliarni, šoféri odvážajúci výrobky do obchodov – tí všetci pracujú pre dobro firmy. A všetkých týchto ľudí má na starosti sused Záviš.

Sused Záviš totiž pracuje ako účtovník. Jeho starosťou je, aby každý zo zamestnancov dostal každý mesiac správne vypočítanú výplatu. Výplata sa zamestnancom počíta podľa počtu odpracovaných dní. Môžete predpokladať, že zamestnanci cez pracovné dni (t.j. pondelok až piatok) pracujú, v sobotu a nedeľu odpočívajú. Účtovanie je náročná práca, preto by sused Záviš ocenil každú pomoc.

ÚLOHA: Napíšte program, ktorý načíta mesiac a rok a vypíše, koľko pracovných dní bolo v danom mesiaci. Váš program by mal fungovať pre ľubovoľný rok, špeciálne by nemal robiť problémy po roku 2000. Sviatky nemusíte uvažovať.

Priklad:

Vstup:

SEP 1999

FEB 2000

Výstup:

September 1999: 22 pracovnych dni

Februar 2000: 21 pracovnych dni

#### z1712. Zase SoDr

V softvérovom družstve SoDr sa rozhodli naprogramovať nový editor. Editor je už takmer hotový a zostáva dorobiť iba niekoľko funkcií. Teraz si všetci lámu hlavu nad tým, ako spraviť presúvanie bloku textu. Potrebovali by vašu pomoc.

ÚLOHA: Editor má globálne pole znakov  $A$  (veľmi veľké). V tomto poli je uložený editovaný text. Užívateľ editora si v texte vyznačil blok, t.j. súvislý úsek textu začínajúci  $i$ -tým a končiaci  $j$ -tým znakom v poli  $A$ . Užívateľ chce tento blok presunúť tak, aby po presune začínal v  $k$ -tom prvku poľa. Vašou úlohou je naprogramovať procedúru  $presun(i, j, k)$ , ktorá dostane čísla  $i$ ,  $j$  a  $k$  a presunie v poli  $A$  blok textu od  $i$ -teho znaku po  $j$ -ty z pôvodného miesta tak, aby začínal na mieste  $k$ . Môžete predpokladať, že  $i$ ,  $j$  a  $k$  sú zadané korektne, t.j. blok má dĺžku aspoň 1 a zmestí sa do poľa tak, aby začínal od pozície  $k$ .

Vaša procedúra nesmie používať ďalšie polia okrem poľa  $A$  a ani iné veľké dátové štruktúry (použiť môžete iba konštantný počet pomocných premenných – znaky, čísla, logické hodnoty a pod.). Súčasne sa snažte, aby vaša procedúra pracovala čo najrýchlejšie.

Například keď je pôvodný obsah poľa  $A$ : 'abcdefgh' po vykonaní  $presun(3, 4, 2)$  bude obsah poľa  $A$ : 'acdbefgh' a po vykonaní  $presun(2, 3, 7)$  je 'abefghcd'.

#### z1713. Zo stavby

Závišovci práve dokončili hrubú stavbu nového domu na Ostrove pri Zmytej kvapke, chystajú sa dokončiť podlahy, kachličkovať, osadiť okná, zaviesť elektrinu a ostatné inžinierske siete. Sami to však nezvládnu, a tak chcú pozvať odborníkov – remeselníkov. Tí si

ale účtujú za prácu veľmi veľké peniaze, a preto im treba naplánovať, kedy má kto prísť. Problémom je, že jednotlivé činnosti musia vykonávať viacerí naraz, napríklad na podlahu v pracovni treba obkladača a elektrikára, na kúpeľňu navyše vodára (či vodníka?), na steny zase omietača a elektrikára. Keď si to pani Závišová dôkladne rozpisala, zistila, že úplne všetky kombinácie rôznych typov remeselníkov majú v dome uplatnenie. Zároveň si zaumierila, že všetkým činnostiam v jej dome bude robiť dozor, takže sa nemôžu vykonávať dve naraz. Samozrejme, nikto nebude zaháľať a len tak postávať – tí čo sú práve nepotrební, musia odísť. Keďže nový dom je na ostrove, chudák pán Záviš ich musí prevážať kompou. A v nej má voľné miesto iba pre jedného.

ÚLOHA: Pre dané  $n$  – počet typov remeselníkov, ktorých označíme  $1, 2, \dots, n$ , vypíšte postupne všetky podmnožiny množiny remeselníkov, a to v takom poradí, že dve za sebou idúce podmnožiny sa líšia iba v jednom prvku – buď niekoho pán Záviš kompou privezie, alebo odvezie. Na začiatku je pritom v dome prázdna množina remeselníkov.

Napríklad pre  $n = 3$  budú podmnožiny vypísané takto: 1.  $\emptyset$ , 2.  $\{1\}$ , 3.  $\{1, 2\}$ , 4.  $\{2\}$ , 5.  $\{2, 3\}$ , 6.  $\{1, 2, 3\}$ , 7.  $\{1, 3\}$ , 8.  $\{3\}$ .

#### z1714. Zblúdilec v zrúcanine

Dr. Jones je archeológom. K jeho zamestnaniu samozrejme patrí aj skúmanie zrúcanín a starých podzemných chodieb. A Jonesovi sa teraz prvýkrát stalo, že zablúdil. Ešte že mu kedysi priateľ poradil fintu – pravidlo pravej ruky. Stačí položiť pravú ruku na stenu a ísť stále tak, aby sa ruka steny dotýkala. Výsledkom má vraj zaručene byť zodretá dlaň a nájdený východ. Tak teraz Jones stojí v chodbe uprostred zrúcaniny a rozmýšľa, ktorá ruka je pravá.

ÚLOHA: Napíšte program, ktorý na obrazovke simuluje prechod bludiskom podľa pravidla pravej ruky. Vstupom sú rozmery bludiska  $m$  a  $n$  ( $m < 80$ ,  $n < 25$ ), súradnice vchodu a východu a mapa bludiska. V mape  $\#$  znamená stenu a  $.$  znamená prázdny priestor. Na celom obvode bludiska je stena, iba vchod a východ sú voľné. Taktiež môžete predpokladať, že vchod a východ sa vždy nachádzajú na obvode bludiska. Na začiatku je Jones vo vchode do bludiska a chce sa dostať k východu. Váš program môže postupne cez prechádzané políčka presúvať nejaký vhodný znak (napr. písmeno J).

Príklad:

Vstup:

$n = 8$ ,  $m = 8$

Vchod: (1, 4), východ: (8, 4)

Mapa:

```
#####
#...#.#
#.#...#
..####.
#...###
#.#...#
#.#...#
#####
```

Výstup:

Dr. Jones podľa pravidla pravej ruky

pôjde cez políčka:

```
(1, 4), (2, 4), (2, 5), (2, 6), (2, 7), (2, 6),
(2, 5), (3, 5), (4, 5), (4, 6), (4, 7), (4, 6),
(5, 6), (6, 6), (6, 7), (7, 7), (7, 6), (6, 6),
(5, 6), (4, 6), (4, 5), (3, 5), (2, 5), (2, 4),
(2, 3), (2, 2), (3, 2), (4, 2), (4, 3), (5, 3),
(6, 3), (7, 3), (7, 4), (8, 4).
```

#### z1715. Zimbabwejský internet

Zimbabwejský kráľ Zimba–Zimba sa rozhodol, že jeho krajina veľmi súrne potrebuje internet. Zimba–Zimba, ako každý správny Zimbabwejský kráľ, veľmi dlho rozmýšľal, ako by toto svoje rozhodnutie mohol vykonať, až nakoniec si vybral firmu ZimNET, ktorá ako jediná ponúkala pripojenie na internet. Teraz už Zimba–Zimba má internet, ale nemôže sa pozeráť na žiadne WWW stránky, pretože firma ZimNET kráľovi nenainštalovala žiaden prehliadač. Zimba–Zimba je teraz veľmi smutný, lebo mu nikto nechce pomôcť.

ÚLOHA: Pomôžte Zimbo–Zimbovi a naprogramujte jednoduchý prehliadač WWW stránok. Ako vstup program načíta WWW stránku vo formáte HTML a zobrazí ju na výstup (do súboru) v čitateľnej forme. Samotný HTML súbor vyzerá podobne ako ten v príklade – na začiatku je vždy `<HTML><BODY>`, na konci `</BODY></HTML>`. Všetky texty uzavreté v `< >` označujú príkazy HTML, ktoré ovplyvňujú, ako sa stránka zobrazí. Niektoré takéto príkazy musia byť v dvojici, napríklad `<HTML>` a `</HTML>` – ten bez lomítka je vždy začiatok a s lomítkom je koniec bloku, na ktorý má daný príkaz vplyv. Takže napríklad `<CENTER>Nadpis</CENTER>` spôsobí, že text *Nadpis* bude tvoriť samostatný odstavec, ktorý bude zarovnaný (vodorovne) na stred stránky. Druhý typ príkazov je nepárový a ide napríklad o príkaz `<P>`, ktorý označuje koniec odstavca. Text žiadnych príkazov sa do výstupu nezapíše. Samotný obsah stránky je tvorený viacerými odstavcami, pričom každý z nich môže zaberáť aj viac riadkov vo výstupe. Vo vstupnom súbore môže byť text jedného odstavca umiestnený na viacerých riadkoch, môže obsahovať veľa medzier, ale toto formátovanie vstupu nemá vplyv na formát výstupu. Každá skupina medzier alebo prázdnych riadkov vo vstupe znamená len oddelenie jednotlivých slov. Tieto jednotlivé slová sa potom zapisujú na výstup, pričom medzi každými dvoma slovami je len jedna medzera. Výstupné zariadenie má obmedzenú šírku (v našom prípade 80 znakov), a preto slovo, ktoré by túto hranicu prekročilo, sa zapíše do nasledujúceho riadku. Medzi každými dvoma odstavcami je vo výstupe jeden prázdny riadok (vo vstupe nemusí byť, napríklad „koniec odstavca 1<P>Druhý odstavce“ je tiež oddelenie odstavcov).

Vašou úlohou je teda podľa načítanej WWW stránky v opísanom formáte vytvoriť súbor, ktorý bude obsahovať text stránky (nie príkazy) zarovnaný na šírku 80 znakov. Predpokladajte, že vstupný súbor obsahuje len opísané príkazy, teda `<HTML>`, `</HTML>`, `<BODY>`, `</BODY>`, `<CENTER>`, `</CENTER>` a `<P>`.

Príklad:

Vstup:

`<HTML>`

`<BODY>`

Zimbabwejsky internet - ukazkový subor<P>

Toto je druhy        odstavce

textu, nezávisle od clenenia vo        vstupnom súbore.

`<CENTER>`

Toto bude vycentrovany odstavce

`</CENTER>`

A nakoniec zase jeden obyčajny odstavce

`</BODY>`

`</HTML>`

Výstup:

(pre šírku strany 40 znakov; rámček nevypisujte – to je len zobrazenie hraníc stránky)

```

+-----+
|Zimbabwejsky internet - ukazkový subor |
|                                           |
|Toto je druhy odstavce textu, nezávisle |
|od clenenia vo vstupnom súbore.         |
|                                           |
|      Toto bude vycentrovany odstavce    |
|                                           |
|A nakoniec zase jeden obyčajny odstavce |
+-----+
```

**z1721. Záhada najkratšej cesty**

Bol teplý júlový večer a traja pátrači opäť sedeli v hlavnom stane. Tentoraz prebiehala vášnivá debata o tom, aká je najkratšia cesta z mestskej knižnice do hlavného stanu. Bobovi to trvalo 17 minút, Jupiter to cez podchod zvládol za 16:30 a Peter, krížom cez čínsku štvrť, to vraj znákol za rovnú štvrthodinku. Všetky tieto cesty však mali jednu vec spoločnú: žiadna netrvala rovnako dlho, a podľa pátračov žiadna z nich nebola najkratšia. Keď sa chýlilo ku polnoci, Jupiter vyhlásil:

„Tá najkratšia cesta sa pred nami nejako skrýva, a existuje vôbec? Už generácie programátorov sa skláňajú pred najkratšou cestou, páni Dijkstra, Floyd, Warshall sú slávni na celom svete. Využime ich služby, aby nám zistili, či vlastne existuje aspoň jedna najkratšia cesta.“

ÚLOHA: Pomôžte svojim kamarátom a napíšte im program, ktorý načíta popis mesta Rocky Beach a vypíše, či existuje najkratšia cesta z významného bodu číslo 1 (čo je hlavný stan) do významného bodu číslo  $n$  (čo je knižnica). Mesto sa skladá z  $n$  významných bodov očíslovaných od 1 po  $n$ , a  $m$  uličiek medzi nimi. Váš program na vstupe dostane čísla  $n$ ,  $m$  a pre každú uličku čísla  $A_i, B_i$  a  $C_i$ , to znamená, že  $i$ -ta ulička dĺžky  $C_i > 0$  spája významné body  $A_i$  a  $B_i$ . Všetky uličky sú obojsmerné.

Napríklad pre  $n = 3$ ,  $m = 2$  a uličky (1, 2, 15) a (2, 3, 7) program vypíše: **najkratšia cesta predsa len existuje!** a pre  $n = 3$ ,  $m = 1$  a uličku (1, 2, 42) vypíše: **Márna vaša snaha, tú najkratšiu nikdy nenájdete!**

**z1722. Záh(r)adné mravenisko**

„Z!“ zarevala zlá kráľovná. Mravec Z zúfalo zrýchlil. Za pár sekúnd už bol v hlavnej komnate. „Ponížene prosím o odpustenie...“ vrhol sa jej k nohám. „Sedemnást tisíc tristo dvadsaťštyri.“ vydýchol. Nato sa kráľovná zamyslela. „A koľkože mravčekov žije na... na... na sto dvadsiatom treťom až štyristo siedmom poschodí?“ Mravec Z sa zdesene zdvihol a vydal sa opäť počítať obyvateľstvo. Z dverí ho vyprevádzal zlomyseľný smiech kráľovnej.

Keď tu zrazu mravec Z dostal nápad. Prešiel si celé mravenisko a na každom poschodí si zrátal počet mravcov, ktoré tam žijú a zaznačil si ich do svojho počítača. Teraz by však potreboval program, ktorý by vedel čo najrýchlejšie povedať, koľko mravcov žije v určitej časti mraveniska.

ÚLOHA: Napíšte program, ktorý dostane na vstupe číslo  $n$  – počet poschodí mraveniska, následne  $n$  čísel, ktoré udávajú počty mravcov na jednotlivých poschodiach. Potom bude načítavať dvojice čísel  $x, y$  ( $x \leq y$ ) a zakaždým vypíše, koľko mravcov žije na  $x$ . až  $y$ . poschodí. Vstup je ukončený dvojicou (0 0).

Také mravenisko má veľmi veľa poschodí, keby ich malo ešte viac, nemuseli by sa počty mravcov na jednotlivých poschodiach hádam ani popratať do pamäte počítača. Kráľovná je veľmi netrpezlivá, preto sa snažte, aby váš program bol čo najrýchlejší. Je aj poriadne škodoradostná, preto počet otázok, ktoré úbohému Z položí, môže byť nepríjemne veľký.

Napríklad pre 13 poschodí a počty mravcov na jednotlivých poschodiach od prvého 1, 3, 4, 7, 6, 9, 4, 5, 2, 4, 123, 1444, 32538 je pre dvojicu 3, 4 odpoveď **Na poschodiach 3-4 žije spolu 11 mravcov.** pre 13, 13 je odpoveď **Na poschodiach 13-13 žije spolu 32538 mravcov.** a pre 1, 13 je odpoveď **Na poschodiach 1-13 žije spolu 34150 mravcov.**

**z1723. Zostarnutý papagáj**

V jednom nemenovanom bytíku v Bratislave žije jedno slušné dievčatko menom Naňka. Keď bola ešte menšia, spadla jej na hlavičku kalkulačka ruskej výroby. Dlhú ju potom boľela hlavička a od tejto chvíle z duše znenávidela všetky kalkulačky. Komplikácie nastali, až keď Naňka chodila do školy. Keď spočítavali malé čísla, stačilo jej jej desať prstov. Potom

začala používať prsty na nohách. Až raz na vianoce jej priletel na balkón papagáj. A nebol to len taký obyčajný papagáj... vedel počítat. Keď mu Ňaňka zakričala: „plus mínus desať krát dva tri osem“, papagáj začal opakovať: „dvanásť, dvanásť, dvanásť...“ A neprestal, pokiaľ nedostal ďalší príklad. Utišil sa iba vtedy, ak ho niekto prekričal pokrikom: „Píš-táá!“ A tak to išlo dlhé časy a Ňaňka sa naučila počítat iba v takomto divnom zápise. Lenže papagáj starne a Ňaňka potrebuje náhradu.

ÚLOHA: Na vstupe máte matematický výraz zapísaný v prefixovej notácii. A čo to vlastne znamená?

1. celé číslo je výraz v prefixovej notácii a jeho hodnota je toto číslo.
2. ak  $v_1, v_2$  sú výrazy v prefixovej notácii s hodnotami  $h_1, h_2$ , tak aj  $+v_1v_2, -v_1v_2, *v_1v_2, /v_1v_2, @v_1$  sú výrazy v prefixovej notácii a ich hodnoty sú  $h_1 + h_2, h_1 - h_2, h_1 * h_2, h_1 \div h_2$  a ciferný súčet  $h_1$ .
3. nič iné nie je výraz v prefixovej notácii

Vašou úlohou je vypočítať hodnotu prefixového výrazu na vstupe. Napríklad pre  $+ - 10 * 2 3 8$  je výstup 12 a pre  $@ 12$  je výstup 3.

#### z1724. O včielkach

Dr. Jones, aj vďaka vašej výdatnej pomoci, šťastne našiel východ zo zrúcaniny. Dľaň na pravej ruke ho pálila, mal ju celú rozodratú až do krvi. Našťastie pred východom zo zrúcaniny tiekla riečka. Sadol si k nej, ponoril ruku do vody a čakal, kým sa mu z ruky vyplavlia choroboplodné zárodky. Voda bola príjemne chladná.

Potom si lahol na chrbát. Cítil sa ako nikdy predtým, slniečko svietilo, bzukotali okolo neho včielky...

V horizontálnej rovine nad jeho hlavou bzukotalo  $n$  včielok so súradnicami  $[x_1, y_1], [x_2, y_2], [x_3, y_3], \dots [x_n, y_n]$ . Ako ich tak pozoroval, všimol si nečakané súvislosti. Všetky včielky sa pohybovali rovnakou konštantnou rýchlosťou. Navyše prvá včielka naháňala druhú, druhá tretiu, atď. až  $(n-1)$ -vá  $n$ -tú a konečne  $n$ -tá včielka prvú.

ÚLOHA: Napište program, ktorý načíta  $n$  – počet včielok a počiatočné súradnice včielok  $[x_1, y_1], [x_2, y_2], [x_3, y_3], \dots [x_n, y_n]$  a bude simulovať ich pohyb po grafickej obrazovke. Včielky znázorníte ako malé bodky.

Prémiová úloha: Zistite a hlavne zdôvodnite, čo by mal robiť váš program pre vstup:  $n = 4, [x_1, y_1] = [0, 0], [x_2, y_2] = [0, 100], [x_3, y_3] = [100, 100], [x_4, y_4] = [100, 0]$ .

#### z1725. Zibabwejský internet II.

Zimbabwejský kráľ Zimba-Zimba smutne hľadel na monitor. Má už síce nový prehliadač, ale čo keď na zimbabwejskom internete nie je veľa zaujímavých stránok. Ten je totiž veľmi mladý. A tak sa Zimba-Zimba rozhodol, že sa sám naučí písať HTML kód.

I učil sa, i učil, až sa nakoniec naučil. A tak celý natešený sa rozhodol, že napíše jednu krásnu kráľovskú WWW stránku. A ako ináč by taká stránka mala vyzerat', ak nie s veľkou hĺbkou obrázkov. I napísal ju a plný očakávania spustil svoj prehliadač. Ale čo to? Missing Image? Tak to hádam nie! Kto ale nájde ten správny súbor? A čo ak chýbajú aj nejaké iné obrázky? Alebo nebodaj aj iné HTML súbory?

Pomôžte Zimbo-Zimbovi a napíšte mu program, ktorý mu vypíše všetky odkazy na obrázky a HTML stránky zoradené v abecednom poradí. Formát HTML súboru ostáva rovnaký ako v predošlej úlohe. To znamená že obsahuje príkazy `<HTML>`, `<BODY>`, `</BODY>`, `</HTML>`, `<CENTER>`, `</CENTER>` a `<P>`. Okrem toho pribudol príkaz na vykreslenie obrázku `<IMG SRC="meno">`, kde meno je názov súboru vo formáte JPEG alebo GIF (s príponami JPG, JPEG alebo GIF). Taktiež odkaz na inú stránku `<A HREF="meno">`, kde meno je názov súboru html (s príponou HTM alebo HTML). Za ním nasleduje text určený na zobrazenie ako odkaz. Ten je ukončený `</A>`. Váš program by mal vypísať počet zobrazených obrázkov vo formáte GIF a ich mená zoradené abecedne, počet obrázkov JPEG

a ich abecedný zoznam. Taktiež vypíše počet odkazov na iné stránky a abecedný zoznam týchto stránok. Dva odkazy na rovnaké súbory, alebo dva rovnaké obrázky sa počítajú len raz, taktiež sa vypisujú len raz. Mená súborov nerozlišujú malé a veľké písmená, takže INTERNET.JPEG je to isté ako iNTerNEt.JpeG.

Nezabúdajte, že formátovanie vo vstupnom súbore nemá vplyv na samotnú HTML stránku, čiže celá stránka môže byť napríklad v dvoch riadkoch – asi takto:

```
<HTML><BODY>Stranka<IMG SRC="auticko.gif"><CENTER>
riadok 1<P>riadok 2</CENTER></BODY></HTML>
```

Príklad:

Vstup:

```
<HTML>
<BODY>
<CENTER> Oficiálna kráľovská stránka Zimba=Zimbu </CENTER>
<CENTER>   <IMG SRC="KORUNA.GIF">   </CENTER> <P>
<IMG SRC="TRON.GIF"> <P>
<A HREF="HOSTINA.HTML"> Viac o kráľovskej hostine TU </A>
<P>
<IMG SRC="TORTA.JPEG"> <IMG SRC="KOLAC.JPG">
Príďte nás všetci pozrieť, ale nezabudnite si prečítať
<A HREF="PRAVIDLA.HTM"> pravidiel. </A> <P>
<P> <P>
<IMG SRC="KORUNA.GIF"> Pozrite si ďalšie naše stránky.
Váš Zimba-Zimba.
</BODY>
</HTML>
```

Výstup

Pocet GIF obrazkov: 2

KORUNA.GIF

TRON.GIF

Pocet JPEG obrazkov: 2

KOLAC.JPEG

TORTA.JPG

Pocet odkazov na stranky: 2

HOSTINA.HTML

PRAVIDLA.HTM

### z1731. Z rodinnej kroniky

Bonifác je veľmi nešťastný. Jeho svokra Emília ho nemá rada. Stále sa s ním iba háda. Bonifác si vzal do hlavy, že Emília má hádavosť vrozenú a na podporu tejto idey začal v rodinnej kronike hľadať všetkých Emíliiných príbuzných a zmienky o ich hádavosti. Ako si tak listoval v kronike, napadla ho zaujímavá úloha:

ÚLOHA: Napíšte program, ktorý pre daných dvoch ľudí zistí, či sú v nejakom príbuzenskom vzťahu a ak áno, vypíše ich najbližší príbuzenský vzťah. Vstupom programu budú výpisy z kroniky tvaru:

SYN ⟨meno syna⟩ ⟨meno rodiča⟩

DCÉRA ⟨meno dcéry⟩ ⟨meno rodiča⟩

MANŽEL ⟨meno manžela⟩ ⟨meno manželky⟩

Prvé dva zápisy hovoria, že prvý uvedený človek je synom resp. dcérou druhého, tretí zápis hovorí, že uvedení dvaja ľudia sú manželia.



Váš program bude ďalej načítavať dvojice mien a pre každú dvojicu vypíše popis najbližšieho príbuzenského vzťahu prvej osoby k druhej.

Popis vzťahu je postupnosť slov syn, dcéra, otec, matka, manžel, manželka v správnych pádoch. Napríklad vzťah „brat“ sa popíše ako „otcov syn“ alebo „matkin syn“. Vzťah je tým bližší, čím menej slov je potrebných na jeho popis. Niektoré vzťahy majú svoje skrátené názvy, napr. „matka manželky“ sa obvykle volá svokra, „syn otca“ je brat. Pri výpise môžete použiť tieto skrátené názvy, avšak toto skrátenie nemá vplyv na blízkosť vzťahu.

So skloňovaním vo výstupe si nelámte hlavu ...

Príklad:

Vstup

SYN Bonifác Filoména

DCÉRA Alena Filoména

MANŽEL Peter Filoména

DCÉRA Anička Emília

MANŽEL Bonifác Anička

Otázky:

Bonifác Emília

Emília Bonifác

Peter Emília

Alena Bonifác

Výstup:

manžel dcéry

matka manželky

otec manžela dcéry

dcéra matky (alebo dcéra otca)

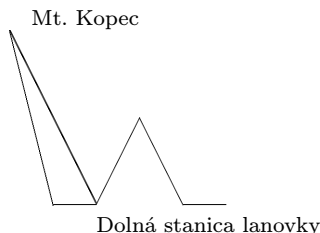
### z1732. Zaujímavá lanovka

V pohorí High Tatras je okrem Mt. Kopca aj veľa iných kopcov a kopčekov. Ale zo žiadneho z nich nevedie lanovka priamo na vrchol Mt. Kopca. Zväz lyžiarov a turistov sa rozhodol tento nedostatok odstrániť vybudovaním novej lanovky. A nie hocijakej, ale najdlhšej možnej. V rámci úsporných opatrení sa rozhodlo, že lanovka sa postaví bez stĺpov. Z tohto dôvodu bude jej dráha vyzeráť ako úsečka (nosné lano bude pevne napnuté medzi koncovými stanicami). Pomôžte funkcionárovi zväzu a napíšte program, ktorý im nájde vhodné miesto na stavbu koncovej stanice lanovky.

ÚLOHA: Hrebeň pohoria dĺžky  $n$  kilometrov vyzerá pri pohľade zvrchu ako rovná úsečka. Pri pohľade z boku vidíme  $n + 1$  význačných bodov (na každom kilometri jeden), pričom každá dvojica susedných význačných bodov je spojená úsečkou. Tieto úsečky dokopy tvoria hrebeň. Pre každý význačný bod  $i$  ( $0 \leq i \leq n$ ) je samozrejme známa jeho nadmorská výška  $A_i$  v kilometroch (reálne číslo).

Vrchol Mt. Kopca leží na kilometri 0 a tvorí jednu koncovú stanicu. Vašou úlohou je napísať program, ktorý zistí, na ktorom význačnom mieste na hrebeni treba postaviť druhú koncovú stanicu tak, aby dráha lanovky viedla vždy nad povrchom a jej dĺžka bola pritom najväčšia možná.

Napríklad pre  $n = 5$ ,  $A_0 = 4$ ,  $A_1 = 0$ ,  $A_2 = 0$ ,  $A_3 = 2$ ,  $A_4 = 0$ ,  $A_5 = 0$  treba nástupnú stanicu umiestniť na 2. kilometer a lanovka bude dlhá 4,47 km.



### z1733. Zelené svahy Kiribati

Kiribatské detičky majú prázdniny, a tak sa najlepší priatelia Mwango a Itab-irik rozhodli, že sa pôjdu lyžovať. Pekná myšlienka, až na to, že na Kiribati býva pomerne teplo

a nie je vôbec jednoduché nájsť nejaký zasnežený kopec. „Podme na ostrov, na ktorom je najviac kopcov. Tak máme najväčšiu nádej, že si tejto zimy zalyžujeme!“ povedal Mwango. A tak si sadli nad mapu Kiribati a začali počítať kopce na jednotlivých ostrovoch. Po chvíli to však vzdali a vyhlásili, že to je robota pre koňa. Alebo pre počítač.

ÚLOHA: Na vstupe sú dve čísla  $m, n$  – rozmery mapy Kiribati a výšková mapa Kiribati. Vode zodpovedá výška 0, súši výška väčšia ako 0. Okraj mapy tvorí voda. Dve políčka sú susedné, ak sa dotýkajú stranou. Dve políčka súše patria k tomu istému ostrovu práve vtedy, keď sa dá z jedného na druhý prejsť po súši tak, že vždy prejdeme na susedné políčko. Políčko súše je kopec práve vtedy, keď je vyšší od všetkých štyroch svojich susedov. Napíšte program, ktorý vypíše súradnice  $x_p, y_p$  niektorého políčka ostrova, na ktorom leží najviac kopcov.

Príklad:

Vstup:

$m = 6, n = 8$

```
0 0 0 0 0 0 0
0 2 4 1 0 0 6 0
0 3 4 1 0 7 2 0
0 0 1 0 0 0 4 0
0 0 0 0 9 0 0 0
0 0 0 0 0 0 0 0
```

Výstup:

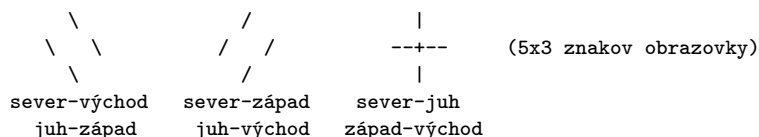
(2,7)

Na tomto ostrove sú 3 kopce. Na ostrove obsahujúcom (5,5) je jeden kopec a na ostrove obsahujúcom (2,2) nie je žiaden.

#### z1734. Zbytočne dlhý ropovod

Nový nález ropy na Ukrajine podnietil slovenskú vládu čo najrýchlejšie vybudovať tranzitný ropovod na našom území. Stavieť sa začalo narýchlo, bez poriadneho projektu, takže plánovanie väčšinou vyzeralo tak, že za stavbyvedúcim prichádzali striedavo zástupca SPP a splnomocnenec vlády pre energetiku a prikazovali, ktorým smerom má viesť najbližší úsek.

Na zjednodušenej mape Slovenska (štvorec  $n \times n$  kilometrov) ropa priteká na políčko v pravom hornom rohu mapy. Vyústenie ropovodu má byť na políčku v ľavom dolnom rohu. Každý úsek zaberá jedno políčko mapy. Skladá sa z dvoch rúr, z toho jedna sa vždy napája na už postavenú časť ropovodu a druhá je kvôli možnému budúcemu rozšíreniu (tzv. budúca rúra). Úseky sú jedného z troch druhov:



Prvý druh úseku sa skladá z rúry spájajúcej stred severnej a východnej strany políčka a rúry spájajúcej stred západnej a južnej strany. Druhý druh vznikne otočením prvého o 90 stupňov. Posledný druh sa skladá z dvoch krížiacich sa rúr spájajúcich severnú s južnou a východnú so západnou stranou políčka.

Môže sa stať, že rúra novopostaveného úseku, ktorá je jedným koncom pripojená k hotovej časti ropovodu sa svojím druhým koncom nadviaže na nejaký úsek budúcej rúry. V takomto prípade sa samozrejme tento úsek a všetky nadväzujúce úseky pripájajú k ropovodu. Nasledujúci úsek ropovodu sa potom stavia na políčku nadväzujúcom na koniec pripojenej časti.

ÚLOHA: Urobte program zjednodušujúci plánovanie stavby ropovodu. V nultom ťahu sa dá vybrať, či ropa na územie Slovenska vteká na západ, alebo juh. V každom ďalšom

ťahu sa vyberá ďalšie smerovanie ropovodu – písmenom 's','j','v','z' označujúcim svetovú stranu na ktorú má smerovať rúra nového úseku pripojená na doteraz postavenú časť.

Pre  $n = 3$ , hra po šiestom ťahu:

```

.....
.      /   XXXX.
.      /   /XXXX.
.      /   XXXX.
. /    |    \   .
./ /---\ \   \
. /    |    \   .
.YYYYY \    /   .
.YYYYY \  \ /   /
.YYYYY \    /   .
.....

```

Boli stlačené klávesy:

```

0.ťah: 'z'
1.ťah: 'j'
2.ťah: 'j'
3.ťah: 'v'
4.ťah: 's'
5.ťah: 'z'
6.ťah: 'j'

```

Môžete použiť aj krajšie znaky pre potrubia, ale používajte iba textový režim zobrazovania.

### z1735. Zimbabwejský internet III.

Zimbabwejský internet nebol ešte nikdy na kráľovom hrade taký populárny, ako teraz, keď sa sám veľký Zimba-Zimba naučil HTML stránky vytvárať. Vytváral, vytváral, až jedného pekného zimného dňa prišiel na to, že vytvorenými stránkami zaplnil celý disk a že sa mu tam už žiadna nová stránka nepomestí. Rozhodol sa preto vymazať všetky nepotrebné stránky, teda také, na ktoré sa nedá dostať prechádzaním po odkazoch z hlavnej kráľovskej stránky. Na to najprv potrebuje zistiť, na ktoré stránky sa to vlastne dostať dá. Napíšte program, ktorý mu s tým pomôže, inak bude musieť vymazať celý disk a začať tvoriť všetky stránky od začiatku.

ÚLOHA: Na disku je uložené množstvo html súborov (s príponou .HTM) obsahujúcich stránky vytvorené Zimbom. Hlavná kráľovská stránka je uložená v súbore INDEX.HTM. Všetky stránky majú rovnaký tvar ako v predchádzajúcich úlohách, teda môžu obsahovať príkazy <HTML>, </HTML>, <BODY>, </BODY>, <CENTER>, </CENTER>, <P>, <IMG SRC="...">, <A HREF="..."> a </A>. Príkaz <A HREF="MENO.HTM"> na html stránke znamená odkaz na stránku uloženú v súbore s menom MENO.HTM. Odkazovaná stránka môže samozrejme obsahovať ďalšie odkazy. Hovoríme, že zo stránky A sa dá dostať na stránku B, ak existuje taká postupnosť stránok  $A = s_1, s_2, \dots, s_n = B$ , že stránka  $s_i$  obsahuje odkaz na stránku  $s_{i+1}$  pre  $1 \leq i < n$ . Váš program má vypísať mená súborov so všetkými stránkami, na ktoré sa dá dostať z hlavnej stránky v súbore INDEX.HTM. Na poradií vypisovania nezáleží, ale každú stránku vypíšte iba raz.

Príklad:

Súbor INDEX.HTM:

```

<HTML><BODY>
<CENTER>Kralovska stranka</center><p>
<A HREF="MENO.HTM">Ako sa volam</a><p>
<A HREF="SUSEDIA.HTM">Susedne krajiny</A>
</BODY></HTML>

```

Súbor MENO.HTM:

```

<HTML><BODY>
Ja som Zimba-Zimba <IMG SRC="ZIMBA.JPG"><p>
Chodim rad do <A HREF="KINO.HTM">kina</A>.
</BODY></HTML>

```

Súbor KINO.HTM:

```

<HTML><BODY>

```

Toto je naše kino.

</BODY></HTML>

Súbor SUSEDIA.HTM:

<HTML><BODY>

Tato stránka sa práve pripravuje.

Môžete sa vrátiť <A HREF="INDEX.HTM">naspat</A>.

</BODY></HTML>

Výstup:

INDEX.HTML

MENO.HTM

KINO.HTM

SUSEDIA.HTM



## Návody k riešeniam

V tejto časti sú uvedené návody k väčšine úloh. K vybraným úlohám je viacero návrhov, ktoré vedú k rôzne efektívnym riešeniam, je to spravidla vtedy, keď sa nám zdalo, že najlepšie riešenie obsahuje netriviálny trik, alebo vtedy, keď postupnosť riešení je sama o sebe zaujímavá.

**111.** Stačí deliť postupne číslami  $2, 3, 5, 7, \dots, 2k+1, 2(k+1)+1, \dots$ . Ak nájdeme deliteľa  $p_i$ , je to istotne prvočíselný deliteľ  $n$ , delíme ním  $n$  koľkokrát sa dá, čím dostaneme príslušný exponent  $e_i$ . Ďalej hľadáme ďalších (väčších) deliteľov čísla  $n' = n/p_i^{e_i}$ .

**112.** Riešenie, v ktorom sa prepisujú nenulové prvky do druhého-pomocného poľa, čím sa aj zhrusťujú, považujeme za nešikovné. V riešení s jedným poľom sa postupne prezerajú prvky od najnižšieho indexu, ak  $A[i] \neq 0$ , posunie sa na miesto  $A[i-z]$ , kde  $z$  je doteraz objavený počet núl. Posledný nenulový prvok je  $A[n - \text{Počet.núl}]$  a zvyšok poľa vynulujeme.

**114.** Pozor, ak generátor dáva rovnomerne rozložené náhodné čísla, musíme generovať všetky hody kockami a nie len náhodné čísla z intervalu  $(6, 36)$ , lebo napríklad súčet 7 môžeme dostať 6 spôsobmi, teda zákonite je súčet 7 pravdepodobnejší než súčet 36.

**115.** Prvky musíme generovať vo vzostupnom poradí podľa ich hodnoty.  $q[0] = 1$  sem patrí a  $q[i]$  musí spĺňať podmienku ii) zadania. Takže  $q[i]$  je najmenšie  $> q[i-1]$ , ktoré má tvar  $2x+1$ , alebo  $3x+1$  pre nejaké  $x \in q[0 : i-1]$ . Treba si zaviesť premenné  $x_2, x_3, j_2, j_3$  také, že  $x_2 = 2 * q[j_2] + 1$  minimálna hodnota  $> q[0 : i-1]$  tvaru  $2x+1$  pre nejaké  $x \in q[0 : i-1]$  a  $x_3 = 3 * q[j_3] + 1$  minimálna hodnota  $> q[0 : i-1]$  tvaru  $3x+1$  pre nejaké  $x \in q[0 : i-1]$ . Riešenie v [v 17 kapitole v 7, 8, v 19.2 v 16].

**121.** b)  $\sim \frac{137}{60}$ . Vo všeobecnosti, keď máme  $n$  pokladníc, je priemerný počet vylomených pokladníc rovný  $H_n = \sum_{k=1}^n \frac{1}{k}$ , [3.2.1 v 20, 1.2.10 a 1.3.3 v 21].

**122.** Celú časť podielu vypočítame ľahko, v ďalšom budeme uvažovať len desatinnú časť. a)  $z_i$  je v poradí  $i$ -ty zvyšok po delení. Keď vypočítame  $k$ -ty zvyšok porovnáme ho postupne so všetkými predchádzajúcimi zvyškami. Ak sa napríklad  $z_j = z_k$ ,  $k \neq j$ , dĺžka predperiódy je  $j$  a periódy  $k-j$ . b)  $v[z]$  obsahuje číslo kroku delenia, kedy sa prvý raz vyskytol zvyšok  $z$ . Pole  $v$  obsahuje na začiatku len 0, počas delenia doň zapisujeme informácie o výskyte zvyškov. Nech je  $x$  prvý zvyšok, pre ktorý  $v[x] \neq 0$ , potom je dĺžka periódy  $k-v[x]$ , kde  $k$  je aktuálne číslo kroku delenia. c) Metóda dvoch bežcov<sup>18</sup>. Počítame postupne zvyšky, ale si ich nepamätáme v poli. V každom kroku delíme dvoma spôsobmi. Prvým spôsobom vypočítame jeden zvyšok. Pri druhom spôsobe vypočítame dva zvyšky – dve cifry podielu. Pri každom z oboch spôsobov si pamätáme len posledný vypočítaný zvyšok a počet delení, ktoré sme vykonali. Keď sa prvý raz zvyšky v oboch prípadoch rovnajú, zistíme dĺžku predperiódy a periódy. Označme si dĺžku predperiódy  $p'$ , periódy  $p$  a  $r$  pozíciu, kde sa prvý raz rovnali zvyšky. Platí, že  $2(p' + r) = p' + kp + r$ , odkiaľ dostaneme, že  $p' = kp - r$ . d) Bez ujmy na všeobecnosti môžeme predpokladať, že  $\text{nsd}(m, n) = 1$ . Nech  $n = 2^\alpha 5^\beta q$  a  $\text{nsd}(10, q) = 1$ . Potom je dĺžka predperiódy  $d = \max\{\alpha, \beta\}$ . Dôkaz.  $\frac{m}{n} = c.b_1b_2\dots b_d a_1\dots a_p$ , kde  $p$  je dĺžka periódy. Platí, že  $10^d \frac{m}{n} = x + \frac{a_1\dots a_p}{10^p-1}$ , kde  $x = cb_1b_2\dots b_d$ . Dostaneme  $\frac{10^d m - nx}{n} = \frac{a_1\dots a_p}{10^p-1}$ . Vzhľadom na predpoklady platí, že  $\text{nsd}(10^d m - nx, q) = 1$ , teda  $q | 10^p - 1$ . Hľadáme základnú periódu, preto chceme  $p$  najmenšie také. Ešte dĺžku predperiódy. Zrejme  $d = \min\{r \in \mathbb{N} \mid 10^r \frac{m}{n} = \frac{y}{10^e-1} \ \& \ y \in \mathbb{Z}\} = \min\{r \in \mathbb{N} \mid \frac{10^r m (10^e-1)}{2^\alpha 5^\beta q} = y \ \& \ y \in \mathbb{Z}\} = \min\{r \in \mathbb{N} \mid 2^{r-\alpha} 5^{r-\beta} \in \mathbb{Z}\} = \max\{\alpha, \beta\}$ . Dôkaz je od Mirka Chlebíka. Iné riešenie je v [cvič. 20.9 v 16].

<sup>18</sup> Táto metóda sa dá úspešne využiť pri zisťovaní cyklickej dátovej štruktúry.

**123.** a) Pole  $k$ -krát posunieme o jeden prvok, na čo treba  $O(n^2)$  posunov. Prípadne ak  $|k| > n/2$ , posúvame  $|k| - n/2$ -krát opačným smerom; b) Využitím pomocného poľa. Prvok  $a_i$  zapíšeme na pozíciu  $(i + k) \bmod n$ .  $2n$  posunov; c)

$$a_0 a_1 \dots a_{n-k-1} b_{n-k} b_{n-k+1} \dots b_{n-1} \longrightarrow a_{n-k-1} \dots a_1 a_0 b_{n-1} \dots b_{n-k+1} b_{n-k} \longrightarrow \\ b_{n-k} b_{n-k+1} \dots b_{n-1} a_0 a_1 \dots a_{n-k-1}.$$

$6n$  posunov; d) [s.73–75 v 8] Nech má počiatočná postupnosť  $p = p_0$  tvar  $AB$ , chceme dostať  $p' = BA$  (posun o  $|B|$  vpravo). Nech  $|A| > |B|$ , pre  $|A| < |B|$  by sa postupovalo analogicky. Označme si  $A = A_0 A_1$  tak, že  $|A_0| = |B|$ . Teraz môžeme v  $A_0 A_1 B$  vymeniť navzájom  $A_0$  s  $B$  a dostaneme  $p_1 = BA_1 A_0$ . Aby sme dostali  $p'$  musíme vymeniť  $A_1$  s  $A_0$ , čo je riešenie pôvodnej úlohy, ale menšieho rozsahu, lebo  $|A| < |p|$ . Všimnite si, že pri vzájomnej výmene dvoch prvkov, vždy aspoň jeden bude definitívne na svojom mieste, teda potrebujeme najviac  $|p_0|$  výmen. e) Kam sa posunie  $a_0$ ? Na  $a_k$  a kam sa posunie pôvodné  $a_k$ ? Na  $a_{2k}$ , atď. Kedy sa vrátíme k prvku, kde sme už boli? Keď  $ik \bmod n = 0$ , pre najmenšie kladné  $i$ , inak povedané  $ik = qn$ , pre  $i, q > 0$ , t.j.  $ik = ns(n, k)$ . Platí, že  $ns(n, k) = nk / \text{nsd}(n, k)$ . Cyklus bude mať dĺžku  $n / \text{nsd}(n, k)$ . Počet cyklov bude  $\text{nsd}(n, k)$ .

**124.** Skúste bez použitia poľa. Pod dĺžkou strany rozumieme počet čísiel v riadku alebo stĺpci bez jednej. Vodorovná strana má dĺžku  $\lfloor \frac{n+1}{4} \rfloor$  a zvislá  $\lceil \frac{n}{4} \rceil$ . Pre párne  $n$  vyjde štvorec úplný a pre nepárne bude na ľavej zvislej strane jedno číslo chýbať. Dokážte, že „štvorec“ s horeuvedenými dĺžkami strán je riešením úlohy.

**125.** a) Určite koľko je permutácií pred permutáciou začínajúcou  $a_0 \ a_1 \ \dots \ a_i$ , ak viete koľko ich je pred takou, čo začína  $a_0 \ a_1 \ \dots \ a_{i-1}$ . b) Podobne ako a). Treba určiť akou cifrou začína permutácia s kódom  $k$ . Kód  $k$  treba zmenšiť o počet permutácií začínajúcich menšími ciframi a uviesť si, že daná cifra sa už vo zvyšku permutácie nemôže vyskytovať.

**211.** Backtracking. Medzi jednotlivé cifry vkladáme znamienka  $+$ ,  $-$ , alebo nič. Všetkých možností je zrejme  $3^8$  (prečo?). Pre každý výraz spočítame jeho hodnotu. Uvedomte si, že vo výraze nemôže byť číslo väčšie než 234 prípadne 123, takže netreba skúšať úplne všetky možnosti. Úloha má 11 riešení.

**212.** Pozor! rekurzívny popis dobre uzátvorkovaných výrazov v zadaní úlohy zvädza k priamočiaremu prepisu do rekurzívneho programu. Takýto program vypíše niektoré dobré výrazy viackrát (vďaka pravidlu 3 v zadaní). Skúste nájsť príklad takýchto opakujúcich sa dobrých výrazov. a) Najjednoduchšie, ale najmenej šikovné je vygenerovať všetky  $2n$  ciferné binárne čísla, 0 a 1 kódujú  $)$  a  $($ , a vybrať z nich len tie, ktoré reprezentujú dobré výrazy. b) Snažíme sa vytvárať dobre uzátvorkované výrazy postupným pridávaním zátvoriek zľava do prava. Zátvorky budeme vypisovať tak, aby sme každý neúplný výraz vedeli vždy doplniť na dobre uzátvorkovaný. Označme si  $l$  počet ľavých zátvoriek, ktoré ešte môžete použiť a  $p$  počet pravých, ktoré musíme vypísať aby bol doteraz vypísaný výraz dobre uzátvorkovaný. Teraz stačí rozobrať akú zátvorku môžete pridať na koniec v prípadoch, keď  $p > 0$  &  $l' > 0$ ;  $p = 0$  &  $l' > 0$ ;  $p > 0$  &  $l' = 0$ . Skúste riešiť bez použitia rekurzie. Počet rôznych dobre uzátvorkovaných výrazov zostavených z  $n$  párov zátvoriek je  $\frac{1}{n+1} \binom{2n}{n}$ <sup>19</sup>. Odvodenie je napríklad v [29, 2.3.4.4 v 21, 1.12 v 27].

**213.** a)  $O(n^3)$ , nájdeme najväčší spomedzi súčtov od  $i$ -teho po  $j$ -ty prvok, pre všetky  $i, j$  také, že  $1 \leq i \leq j \leq n$ . b)  $O(n^2)$  ako a) ale využijeme súvislosť medzi súčtami prvkov  $x_i$  až  $x_{j-1}$  a  $x_i$  až  $x_j$ , pre  $1 \leq i < j \leq n$  c)  $O(n \log n)$  (metóda rozdeľ a panuj). Rozdelíme postupnosť na dve časti, ktorých dĺžka sa líšia najviac o jedna. V oboch častiach nájdeme rovnakým spôsobom, maximálne podpostupnosti. Výslednú maximálnu postupnosť celej

<sup>19</sup> Catalanové číslo, podľa belgického matematika Eugena Charlesa Catalana, 1814-1894.

postupnosti vyberieme z maximálnych postupností oboch častí a postupnosti ktorá „leží“ na hranici oboch častí. d)  $O(n)$ , rozšírenie riešenia pre podpostupnosť  $x_1$  až  $x_{i-1}$  na riešenie pre podpostupnosť  $x_1$  až  $x_i$ .

```

max_doteraz := minint; sucet_sem := minint;
for i := 1 to n do begin
  sucet_sem := max(sucet_sem + x[i], x[i]);
  max_doteraz := max(max_doteraz, sucet_sem);
end;
```

Doplňte program tak, aby určil aj kde hľadaná podpostupnosť začína a akú má dĺžku. Riešenia sú prevzaté z kapitoly 7 v [3].

**214.** Netreba tisícprvkové pole. Zistíte koľko rôznych súčtov môžeme pre trojciferné čísla dostať. Riešenie by sa malo dať ľahko zovšeobecniť aj na viac než trojciferné čísla. Pokúste sa dokázať, že postup pre ľubovoľnú počiatočnú hodnotu vždy skončí číslom 1 alebo 4, [19].

**215.** Problém: ako zistiť koniec vstupného textu? Nešikovné je vyžadovať počet znakov, lepší je prázdny riadok, alebo ukončovací znak, alebo štandardná možnosť programovacieho jazyka. Ak Váš počítač používa kód ASCII, je šikovné využiť kódy znakov pri vytváraní tabuľky početností – početnosť znaku bude uložená v poli na indexe rovnom kódu daného znaku. Percentuálne zastúpenie je dobré vypočítať až pri výpise a nulové výskyty nevypisovať.

**221.** Treba si uvedomiť, že stačí sledovať len vzdialenosť častice od vnútornej steny reaktora, označme ju  $y$ . Na začiatku častica vletí do steny pod uhlom  $\alpha_1$ ,  $y_1 = \sin \alpha_1$ . Ďalej platí,  $y_{i+1} = y_i + \sin(\alpha_{i+1} + \gamma_i)$ , kde  $\gamma_i = \gamma_{i-1} + \alpha_i$ ,  $\gamma_0 = 0$  a  $1 \leq i \leq 20$ . Ak  $y_{20} < 0$ , častica sa vrátila späť do reaktora, ak  $y_{20} > H/25$ , častica stenou preletí,  $H$  je hrúbka steny v centimetroch.

**222.** Refazové zlomky, [40 v 4. kapitole]. Platí, že medzi zlomkami, ktorých menovatele sú menšie alebo rovné  $Q_k$  aproximuje najlepšie číslo  $p/q$  refazový zlomok  $P_k/Q_k$ .  $P_k = q_k P_{k-1} + P_{k-2}$ ,  $Q_k = q_k Q_{k-1} + Q_{k-2}$ ,  $P_{-1} = 0$ ,  $P_0 = 1$ ,  $Q_{-1} = 1$ ,  $Q_0 = 0$ .  $q_k$  je neúplný podiel v  $k$ -tom kroku výpočtu najmenšieho spoločného deliteľa čísel  $p$  a  $q$ . Platí, že pre ľubovoľné dve celé čísla  $p, q$  existuje práve jedna dvojica čísel  $q_1, r_1$  takých, že  $p = q q_1 + r_1$  a  $0 \leq r_1 < |q|$ .  $r_1$  je zvyšok, ak  $r_1 \neq 0$ ,  $q_1$  sa nazýva neúplným podielom, inak je to úplný podiel. Iné riešenie založené na Stern-Brocotovom<sup>20</sup> strome je v [12].

**223.** Presne podľa definície dostaneme algoritmus zložitosti  $O(n^2)$ . Využitím intervalového stromu sa dá vytvoriť algoritmus zložitosti  $O(n \log n)$ . Koreň bude reprezentovať interval  $\langle 1, n \rangle$  a jeho ľavý syn po polovicu a pravý syn od polovice. V každom vrchole si pamätáme koľko doteraz načítaných čísel patrí do tohto intervalu. Ak načítame nové číslo tak ho vieme v logaritmickom čase zaradiť do stromu a takisto v logaritmickom čase spočítať koľko čísel bolo od neho väčších [23 v 5.1.1.6]. .

**224.** Backtracking. Binárny refazec s hľadanými vlastnosťami sa nazýva de Bruijnovým<sup>21</sup> cyklom dĺžky  $2^n$ .

**225.** Využite výsledok 215. Urobte interaktívny program a postupným vylepšovaním dekódujte text.

**231.** ii).

**232.** 13. deň v mesiaci padol na pondelok 173, utorok 169, stredu 173, štvrtok 171, piatok 171, sobotu 172 a nedeľu 171 krát. a) Pre každý mesiac určite o koľko je dlhší než 28 dní. Keby mali všetky mesiace presne 28 dní úloha by sa veľmi zjednodušila. Vtedy stačí vedieť,

<sup>20</sup> volá sa podľa francúzskeho hodinára Achilla Brocota a nemeckého matematika Moriza Abrahama Sterna, 1807-1894.

<sup>21</sup> de Bruijn, Nicolaas Govert, ????



ktorý deň bol 1. 1. 1901. b) d.m.r je dátum,  $1582 \leq r \leq 4902$ . Nech d je deň (0 – nedeľa, 1 – pondelok, ..., 6 – sobota),  $y = r \bmod 100$ ,  $c = r \operatorname{div} 100$  a  $m'$  je mesiac (1 – marec, 2 – apríl, ..., 11 – január, 12 – február)

$$\text{deň v týždni} = [2.6m' - 0.2] + d + y + \left\lfloor \frac{y}{4} \right\rfloor + \left\lfloor \frac{c}{4} \right\rfloor - 2c$$

Pri odvádzaní horeuvedeného vzorca si treba uvedomiť, ktoré roky sú priestupné<sup>22</sup>. Predpokladajte, že viete, ktorý deň v týždni bol 1. marec niektorého roku (napríklad tohoto) a určite, ktorý deň v týždni je 1. marec roku r. Potom treba ešte pripočítať počet dní od 2. marca po daný dátum d.m, vrátane, modulo 7. Treba pri tom využiť aproximáciu. Odvodenie je napríklad v [8.2 v 33].

**233.** Riešenia, v ktorých sa kontrolujú všetky postupnosti, alebo riešenia využívajúce dvojkovú sústavu nepovažujeme za dobré. Skúste nájsť vzťah medzi počtom hľadaných postupností dĺžky i a  $i + 1$ . Označme počet postupností dĺžky i, neobsahujúcich za sebou idúcich k núl  $p_k(i)$ . O koľko je  $p_k(i + 1)$  menej než  $2p_k(i)$ ? Teda keby sme z každej postupnosti dĺžky i vyrobili pridaním nuly a jednotky dve dlhšie. Presne  $p_k(i - k)$  postupností z  $p_k(i)$  je zlých lebo ich predĺžením o nulu by sme dostali postupnosť obsahujúcu za sebou idúcich k núl.

**234.** Zjednodušená pivotizácia podľa 0.

**235.** Počiatok súradnicovej sústavy dajte do bodu  $[\frac{n}{2}, \frac{n}{2}]$ , aby sa mriežka ľahko otáčala.

**311.** Uvedomte si, kedy už pre niektorý prvok istotne viete, že nebude vo všetkých riadkoch. Veľmi pomôže to, že riadky sú usporiadané. a) Pre každý riadok si udržiavame index prvku, po ktorý sme už v ňom pri kontrole prišli. Každý prvok testujeme najviac raz. Ak sú v niektorom riadku všetky čísla menšie než kandidát na spoločný prvok, môžeme skončiť. b) Držíme si vzostupne usporiadaných kandidátov na spoločný prvok. Na začiatku sú to všetky prvky prvého riadku. Kandidátov počas čítania ďalších riadkov aktualizujeme. Ak všetci kandidáti vypadnú, môžeme skončiť. Každý prvok poľa kontrolujeme najviac raz. Navyše si stačí pamätať najviac jeden riadok z celého poľa.

**312.** Určite, kedy sa dá skupina istotne zapísať [(voľné miesto v poli) + (dĺžka skupiny s rovnakou hlavičkou) – (dĺžka danej skupiny)  $\geq 0$ ].

**313.** Uvedomte si, že netreba používať žiadne pole, stačí si náhodne vybrať jedného z voľčov a niektorého z jeho ôsmich susedov (tri náhodné čísla). Voľby budú prebiehať nasledovne: začnú sa vytvárať čoraz väčšie skupiny rovnako hlasujúcich, pokiaľ neostanú len dve a po čase napokon zostane len jedna. Úloha je z [9].

**314.** Použite matematickú indukciu vzhľadom na počet závaží.

**315.** Konštruujte triedy ekvivalencie podľa danej relácie R.

**321.** Máme dve možnosti:  $b = \sqrt{c^2 - a^2}$ ,  $0 < a \leq c/\sqrt{2}$ , alebo  $a = \sqrt{c^2 - b^2}$ ,  $c/\sqrt{2} \leq b < c$ . Všimnite si, že druhý spôsob je efektívnejší. Ďalšie zefektívnenie dosiahneme využitím, že ak je c deliteľné 2 a/alebo 3 môžeme kontrolovať menej odvesien.

**322.** Pripočítavanie 1 v dvojkovej sústave.

**323.** a) rovnica úsečky s koncovými bodmi  $(x_1, y_1)$ ,  $(x_2, y_2)$ , kde  $x_1 < x_2$ , je

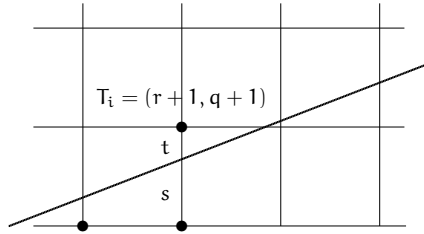
$$y = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1, \quad x_1 \leq x \leq x_2.$$

<sup>22</sup> Priestupné roky boli zavedené v Juliánskom kalendári, uvažovali, že rok má  $365\frac{1}{4}$  dňa, čo je oproti astronomickému roku o niečo viac, ten má 365.2422 dňa. Preto pápež Gregor XIII prijal roku 1582 reformu kalendára. Vypustili desať dní 11. marec sa stal 21. marcom. Reformu ovplyvnil významný posun kalendára vzhľadom na začiatky ročných období. V Anglicku bola reforma prijatá r. 1752, 3. september sa stal 14. septembrom a v Rusku prijali reformu r. 1918, 1. február sa stal 14. februárom.

Nevýhodou je, že pre „strmé“ úsečky dostaneme jednotlivé body „ďaleko“ od seba, čo sa dá ale ľahko odstrániť použitím duálneho vzťahu

$$x = \frac{x_2 - x_1}{y_2 - y_1}(y - y_1) + x_1, \quad y_1 < y_2, \quad y_1 \leq y \leq y_2.$$

b) Bresenhamov<sup>23</sup> algoritmus. Nech je  $y = kx + l$ , kde  $k \in (0, 1)$ . Úsečku s koncovými bodmi  $[x_1, y_1], [x_2, y_2]$ ,  $x_1 \leq x_2$ , ležiacu na tejto priamke posunieme tak, aby sa  $[x_1, y_1]$  posunul do  $[0, 0]$ .  $[x_1, y_1] \rightarrow [0, 0]$ ,  $[x_2, y_2] \rightarrow [x_2 - x_1, y_2 - y_1] = [d_x, d_y]$ . Priamka má teraz rovnicu  $y = \frac{d_y}{d_x}x$ . Úsečka má počiatočný bod  $[0, 0]$  a koncový bod  $[d_x, d_y]$ . Zobražíme ju prostredníctvom pixlov, ktoré sú najbližšie v smere y-ovej súradnice. Pixle majú celočíselné súradnice.



$P_{i-1} = (r, q)$   $S_i = (r+1, q)$

$D_i \geq 0$ , vyberieme bod  $T_i$ , takže  $D_{i+1} = D_i + 2(d_y - d_x)$ . Ak  $D_i < 0$ , vyberieme bod  $S_i$ , vtedy  $D_{i+1} = D_i + 2d_y$ . Ešte  $D_1 = 2d_y - d_x$ .

Úsečky zobrazujeme vo všetkých oktancoch, každá sa dá vhodnou transformáciou previesť na náš prípad. Stačí vhodne zvoliť prírastky  $s_x, s_y, d_x, d_y$ , prípadne vzájomne vymeniť x-ovú a y-ovú súradnicu. Riešenie je z [14].

```

procedure ciara(x, y, x2, y2 : integer);
var i, sx, sy, dx, dy, e : integer;
    strma : boolean;
begin
    dx := abs(x2 - x); sx := sgn(x2 - x);
    dy := abs(y2 - y); sy := sgn(y2 - y);
    strma := (dx < dy);
    if strma then
        begin Vymen(x, y); Vymen(sx, sy); Vymen(dx, dy) end;
    e := 2 * dx - dy;
    for i := 1 to dx do
        begin
            if strma then PutPixel(y, x)
            else PutPixel(x, y);
            while 0 <= e do
                begin y := y + sy; e := e - 2 * dx end;
                x := x + sx; e := e + 2 * dy
            end;
            PutPixel(x2, y2)
        end;

```

<sup>23</sup> Jack E. Bresenham

**324.** Prehľadávanie do šírky. (Nie backtracking!)

**325.** Pozri 115.

**331.** Orezávanie. Rovinu rozdelíme na 9 častí. Bodu  $(x, y)$  priradíme  $s(x)$  a  $s(y)$ , kde

$$s(x) = \begin{cases} -1 & \text{ak } x < x_{\min} \\ 0 & \text{ak } x_{\min} \leq x \leq x_{\max} \\ 1 & \text{ak } x_{\max} < x \end{cases} \quad s(y) = \begin{cases} -1 & \text{ak } y < y_{\min} \\ 0 & \text{ak } y_{\min} \leq y \leq y_{\max} \\ 1 & \text{ak } y_{\max} < y \end{cases}$$

Teraz vieme priamo určiť, ktoré úsečky ležia celé v zobrazovacom okne a niektoré z tých, ktoré sú istotne celé mimo. Pre zvyšné tri prípady treba vypočítať prienik úsečky s hranicami okna [13, 15].

**332.** Priamočiara realizácia. Pozor! Odchody domorodcov treba simulovať po jednom, alebo uvažovať reálne čísla a na koniec zaokrúhliť.

**333.** Označme  $m_n$  číslo, ktoré hľadáme a  $P$  množinu všetkých prvočísel. Platí, že

$$m_n = \prod_{p \in P} p^{\max\{l | p^l \leq n\}}.$$

Teraz už stačí len vedieť násobiť veľké čísla.

**334.** Pre tri po sebe idúce členy  $\frac{m'}{n'}$ ,  $\frac{m''}{n''}$  a  $\frac{m'''}{n'''}$  platí

$$m''' = \left\lfloor \frac{n' + n}{n''} \right\rfloor m'' - m', \quad n''' = \left\lfloor \frac{n' + n}{n''} \right\rfloor n'' - n'$$

Označme pravé strany  $\hat{m}$  a  $\hat{n}$ . Platí  $\hat{m}n'' - m''\hat{n} = 1$ . Z čoho dostaneme  $\hat{m}/\hat{n} \geq m'''/n'''$ , ak by neplatila rovnosť,  $n''' = (\hat{m}n'' - m''\hat{n})n''' = n''(\hat{m}n''' - m''\hat{n}) + \hat{n}(m'''n'' - m''n''') \geq n' + \hat{n} > n$ , čo je spor. Riešenia v [21, 12].

**335.** Rozmyslite si, kam má zmysel dávať '( a ' )' a že  $-(a_i - (a_{i+1} + a_{i+2})) = -(a_i - a_{i+1}) + a_{i+2}$ .

**411.** Označme cifry riešenia  $a_n, a_{n-1}, \dots, a_1, a_0$  a počet cifier  $p_c = n + 1$ . Platí,  $p_c = a_0 + a_1 + \dots + a_{n-1} + a_n = 0 \cdot a_0 + 1 \cdot a_1 + \dots + n \cdot a_n$ ,  $\forall j > 6 : a_j = 0$ ,  $\forall j > 2 : a_j < 2$  a najviac jedno  $a_j > 0$  je také, že  $j > 2$ .

**412.** Stačí si uvedomiť, že pri podmnožinách nezáleží na poradí prvkov. Označme si prvky v  $k$ -prvkovej podmnožine  $a_0 a_1 \dots a_{k-1}$ , pre  $i < j$  platí, že  $a_i < a_j$ . Treba vlastne zistiť, koľko je takýchto  $k$ -tic začínajúcich s  $a'_0 < a_0$  plus  $(k-1)$ -tíc začínajúcich s  $a'_1 < a_1$  atď. až po  $a_{k-1}$ . Teda

$$\text{kód}(a_0 a_1 \dots a_{k-1}) = 1 + \sum_{i=0}^{k-1} \sum_{l=a_{i-1}+1}^{a_i-1} \binom{n-l}{k-i-1}, \quad a_{-1} = 0,$$

čo sa dá upraviť do tvaru

$$1 + \binom{n}{k} - \left[ \binom{n-a_0}{k} + \binom{n-a_1}{k-1} + \dots + \binom{n-a_{k-2}}{2} \right] - \binom{n-a_{k-1}+1}{1}.$$

**413.** a)  $O(n^4)$ . Bod isto nie je vrcholom konvexného obalu (KO), keď leží vo vnútri nejakého trojuholníka s vrcholmi z danej množiny bodov. Či bod leží vo vnútri trojuholníka sa dá zistiť v konštantnom čase. Z  $n$  bodov vieme vytvoriť  $n^3$  trojuholníkov a pre každý skontrolujeme  $n-3$  bodov. b)  $O(n^3)$ , Budeme hľadať hrany KO. Úsečka určená dvoma bodmi je hrana KO, práve vtedy, keď všetky ostatné body množiny ležia v jednej polrovine

ňou určenou. Takže pre každú z  $n^2$  úsečiek zistíme, či zvyšných  $n - 2$  bodov leží v jednej polovine určenej niektorou úsečkou. c)  $O(n^2)$ , Jarvisov algoritmus; v predchádzajúcom spôsobe netreba skúmať všetky dvojice bodov. Keď máme hranu KO  $pq$ , stačí skúmať úsečky s koncovým bodom  $q$ . Algoritmus môžeme zapísať takto:

1. nájdeme bod  $s$  s najmenšou  $x$ -ovou súradnicou, ak je takých viac, ten s najmenšou  $y$ -ovou, označme ho  $p_1$ .
2.  $p_2$  nájdeme, ako bod s najmenším polárnym uhlom vzhľadom na  $p_1$ . Každý nasledovný vrchol KO vieme nájsť v lineárnom čase, vzhľadom na počet bodov.
3. Podobne postupujeme pri hľadaní  $p_i$ , je to bod s najmenším polárnym uhlom vzhľadom na  $p_{i-1}$ .

Ak je veľa bodov vo vnútri KO je tento algoritmus výhodný, pretože vyžaduje len približne  $kn$  operácií, kde  $k$  je počet vrcholov KO. Ďalšou výhodou je, že sa dá zovšeobecniť pre body v priestore. Ako určiť najmenší uhol ukážeme v ďalšom. d)  $O(n \log n)$ , Grahamov<sup>24</sup> algoritmus. Hlavná myšlienka je v utriedení bodov podľa ich polárnych súradníc vzhľadom na nejaký bod z vnútra konvexného obalu (ťažisko trojuholníka, ktorého vrcholy sú ľubovoľné tri body z danej množiny). Ak sú polárne uhly niektorých bodov rovnaké, usporiadame ich podľa rastúcej vzdialenosti. Idea algoritmu je založená na skúmaní po sebe idúcich trojíc bodov  $p_1 p_2 p_3$ . Postupujeme v smere hodinových ručičiek a zisťujeme, či body tvoria dutý uhol, ak áno hovoríme, že  $p_1 p_2 p_3$  sú vpravoorientované, inak sú ľavoorientované. Dá sa to zistiť nasledovne:

$$\begin{array}{l} p_1 = (x_1, y_1) \\ p_2 = (x_2, y_2) \\ p_3 = (x_3, y_3) \end{array} \quad P = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} > 0 \iff \text{keď } p_1, p_2, p_3 \text{ sú vpravoorientované.}$$

Každé tri po sebe idúce vrcholy KO budú ľavoorientované. Takže keď  $p_1, p_2, p_3$  sú vpravoorientované,  $p_2$  nemôže byť vrcholom KO. Štartovací bod  $p_1$  nájdeme rovnako ako v Jarvisovom algoritme, ale spomedzi už utriedených bodov. Ako zarážku pridáme bod  $p_0 = (x_1 + 1, y_1)$ . Výsledný algoritmus je nasledovný

1.  $i := 1$
2. skončíme keď  $i = n - 1$ .
3. ak sú  $p_i, p_{i+1}, p_{i+2}$  vpravoorientované, vylúč  $p_{i+1}$  a pokračuj s  $p_{i-1}, p_i, p_{i+2}$ .
4. ak sú  $p_i, p_{i+1}, p_{i+2}$  ľavoorientované, pokračuj s  $p_{i+1}, p_{i+2}, p_{i+3}$ . Algoritmy sú popísané v [35.3 v 6, 8.4 v 30, VIII.2 v 31 35, 25 v 37].

**414.** a) Riešenie využívajúce smerníky. Každý človek v kruhu si pamätá svojho aktuálneho nasledovníka. b) Prekvapujúci súvis s úlohou 223 [5.1.1.2 v 23]. Poradie vypadávaní je permutácia. Tabuľka inverzií tejto permutácie sa dá vypočítať takto  $b_1 = (m - 1) \bmod n$ ,  $b_{j+1} = (b_j + m - 1) \bmod (n - j)$ . Iné riešenie je v [12], riešenie pre  $m = 3$  je v [26 v 2.5.10].

**415.** Binárne vyhľadávanie, pozri napríklad [4. kapitolu z 3, alebo 1.6 v 42].

**421.** Zvolíme si smer a postupne počítame súčet uhlov (uhly budeme brať z intervalu  $\langle 0^\circ, 360^\circ \rangle$ ), o ktoré sa musíme otočiť aby sme nakreslili zadaný  $n$ -uholník. Platí, že pre ľubovoľnú uzavretú čiaru je súčet rovný  $k \cdot 360^\circ$ , pre nejaké  $k$ . Ak je čiara konvexná, je  $k = 1$ .

**422.** Ihla pretne priamku práve vtedy, keď  $x \leq h \sin \alpha$ , kde  $x \in \langle 0, h \rangle$  a  $\alpha \in \langle 0, \pi \rangle$ . (Pozor, treba generovať náhodný uhol  $\alpha$  a nie priamo  $\sin \alpha$ .) Pravdepodobnosť prípadov, že pre dvojicu  $(x, \alpha)$  je splnená nerovnosť  $x \leq h \sin \alpha$  sa rovná

$$\frac{\int_0^\pi h \sin \alpha d\alpha}{h\pi} = \frac{2}{\pi}.$$

<sup>24</sup> Ronald Lewis Graham, 1935-, americký matematik

**423.** Backtracking.

**424.** Urobte načítanie celého čísla so znamienkom. Potom stačí prečítať číslo pred desatinnou bodkou, za desatinnou bodkou a exponent a z nich reálne číslo spočítať. Pozri [1.11.3 v 42].

**425.** Pre nenulové prvky vieme vypočítať, kde majú byť po preusporiadaní. Číslo  $i$  má byť na mieste  $i + (i - 1) \operatorname{div} (\frac{n}{R} - 1)$ .

**431.** Pole utriedime.

**432.** a) V  $i$ -tom kroku dáme na miesto kartičku s číslom  $i$ . Na jeden krok treba najviac dve otočenia. b) Políčko  $k$  rozdelí kartičky na dve časti, označme ich  $A$  a  $B$ . Najprv vymeníme navzájom prvky nepatriace do časti  $A$  s tými, čo nepatria do časti  $B$ . Ďalej ako v a). c) optimálne riešenie – backtracking?

**433.** Backtracking.

**434.** Čísla menšie než  $2^k$  majú najviac  $k$  cifier. Stačí teda generovať všetky čísla  $x < 2^{\lfloor k/2 \rfloor}$  a ak sa to dá, z každého vyrobiť tri symetrické čísla  $xx^R$ ,  $x0x^R$  a  $x1x^R$ , kde  $x^R$  je zrkadlový obraz  $x$ .

**435.** „Odzadu“ stačí vypisovať do buffera cifry čísla  $10^m r$  a po  $m$  cifrách vypísať desatinnú bodku a zvyšné cifry. V prípade, že je dĺžka  $p$  vypísaného čísla menšia alebo rovná  $n$ , vypíšeme ešte  $n - p$  medzier, inak buffer naplníme '\*', čo znamená, že sa číslo  $r$  v danom formáte nedá vypísať. Pozri [1.11.3 v 42].

**511.** Vypočítajte, koľko medzier treba pridať medzi slová v riadku, aby bol výstupný riadok zarovnaný. Pozor na dlhé slová. Vyriešte otázku viacnásobných medzier vo vstupnom texte. Text čítajte po riadkoch. Riešenie v [B.2 v 28, 16].

**512.** Určte  $Q_i$  na základe  $Q_{i-1}$ . Označme indexy od 0. Všimnite si, že  $i$ -ty prvok v  $Q_i$  je  $i$ -ty aj v  $Q$ , takže stačí určiť jeho pôvodný index v  $Q_0$ . Ak má prvok v  $Q_i$  index  $k_i$ , v  $Q_{i-1}$  bolo pred ním  $v = (k_i \operatorname{div} i) = k_i - k_i \bmod i$  prvkov, ktoré sme vyškrtnú. Platí teda  $k_{i-1} = k_i + v$ ,  $i > 0$ .

**513.** Na reprezentáciu rytierov použite spájaný zoznam, prípadne pole.

**514.** Počítajte len plochu  $\frac{1}{8}$  kruhu. Prípadne vpíšte do kruhu najväčší možný štvorec a spočítajte len plôšky, ktoré ostali.

**515.** a) Použijeme zarážku okolo poľa. Oblasti vyfarbujeme postupne po jednej, vždy celú. Z každého políčka súvislej oblasti sa snažíme rozšíriť na jeho štyri susedné políčka. b) Stačí si pamätať dva susedné vodorovné pásiky poľa. Každú súvislú oblasť vyfarbujeme rôznou farbou, ak zistíme, že sa nám dve súvislé oblasti spojili, farby zjednotíme, využijeme na to pomocné pole farieb.

**521.** Rozdeľte  $k$  na tisícky, stovky a desiatky s jednotkami.

**522.** Nie backtracking! a) [16] Pre každý prvok  $a_i$  postupnosti určíme číslo  $b_i$ , t.j. dĺžku maximálnej podpostupnosti začínajúcej prvkom  $a_i$ . Čísla  $b_i$  počítame od  $b_n$ . Platí, že  $b_i = 1 + \max\{b_j \mid i < j \leq n \text{ a } a_i < a_j\}$ . Inými slovami  $b_i$  je o jedna viac než maximum dĺžok tých podpostupností, ku ktorým môžeme spredu pridať prvok  $a_i$ . Riešenie vyžaduje  $O(n^2)$  operácií. b) [6.11.1 v 30, 20.2 v 16, 8] Označme  $P_k$  momentálne najdlhšiu podpostupnosť dĺžky  $k$  s čo najmenším posledným prvkom  $P_k.\text{last}$ . Pre postupnosť menšej dĺžky než  $m$  vieme určiť  $P_k$  pre všetky  $1 \leq k < m$ . Ak  $k = 1$  je to ľahké. Keď  $k$  postupnosti pridáme  $x_m$  treba určiť, ktoré  $P_k$  sa zmení. Musí platiť  $P_k.\text{last} < x_m$  a súčasne  $x_m < P_{k+1}.\text{last}$ . Nech je  $l$  dĺžka doteraz najdlhšej podpostupnosti. Ktoré  $P_i$  zmeníme určíme takto: spomedzi  $i = l, l-1, \dots$  nájdeme najväčšie také, že  $P_i.\text{last} < x_m$ , ak také neexistuje,  $P_1$  bude  $x_m$ . Ak  $i = l$ , tak  $P_{l+1} = P_l x_m$  a  $l$  zvýšime o 1. Inak  $P_{i+1} = P_i x_m$ . Vyhľadávať môžeme binárne (úloha 415), takže celkovo potrebujeme  $O(n \log n)$  operácií. V programe si stačí pamätať

$P_k$ .last a  $x_m$ .prev, ktorý určuje index predchádzajúceho prvku podpostupnosti v pôvodnej postupnosti.

**523.** Dijkstrov<sup>25</sup> algoritmus. Mestá rozdelíme na dve množiny, množinu kandidátov  $K$  a množinu definitívne určených  $D$ . Na začiatku je v  $D$  len počiatočné mesto  $u$ , ostatné sú v  $K$ . Každé mesto  $z$  z  $K$  má určenú do teraz minimálnu vzdialenosť od počiatočného mesta, ale vedúcu len cez mestá z  $D$ . Z  $K$  vyberieme v každom kroku to mesto  $x$ , ktorého  $c(x)$  je minimálna vzdialenosť z  $u$  a dáme ho do  $D$ . Tým sa mohli zmeniť zatiaľ minimálne  $c(v)$  pre  $v$  z  $K$ , tých miest, do ktorých vedie kratšia cesta cez  $x$ . Takže opravíme:  $c(v) = \min\{c(v), c(x) + (x, v)\}$ , kde  $(x, v)$  je dĺžka cesty z  $x$  do  $v$ , podľa zadania je to  $A[x, v]$ . V každom kroku jedno mesto z  $K$  vypadne a pre každé mesto musíme v každom kroku robiť opravu. Takže celkovo treba  $O(n^2)$  operácií. Zrejme platí, že  $(x_i, x_{i-1}) \geq c(x_i) - c(x_{i-1})$ , pre ľubovoľné dva vrcholy ciest z  $u$ . Nech  $u = x_0 x_1 \dots x_m = v$ , sčítaním dostaneme  $\sum_{i=1}^m (x_i, x_{i-1}) \geq c(x_m) - c(x_0) = c(v)$ . Pozri [6.1 v 25, 4.13 v 34, 3.3 v 27].

**524.** Pozri 515.

**525.** Určite uhly otočenia jednotlivých kĺbov pre body  $P_1$  a  $P_2$ ,  $(\alpha_1, \beta_1, \gamma_1)$  a  $(\alpha_2, \beta_2, \gamma_2)$ . Počet operácií *OTOC* potrebných na presun ramena je  $\max\{|\alpha_1 - \alpha_2|, |\beta_1 - \beta_2|, |\gamma_1 - \gamma_2|\}$ . Na koniec treba ešte vybrať najbližší mrežový bod k bodu  $P_2$ .

**531.** Nájdite výskyt slova tisíc a sto. Pozor na výnimky: desať, jedenásť, štrnásť, dvesto, dvetisíc.

**532.** Pozri 122.

**533.** Rozmyslite si, čo treba robiť keď prirovnávate dve konštanty, premennú, ktorá nemá hodnotu a konštantu, premennú, ktorá nemá hodnotu s premennou, ktorá má hodnotu a dve premenné, buď obe s, alebo bez hodnôt. Algoritmus by mal mať zložitosť  $O(n)$  operácií, kde  $n$  je dĺžka vstupného reťazca.

**534.** Ak si nevšímate čísla  $< k$ , bude  $k$  prvý raz vedúcim v 1. kroku. Ak je  $k$  prvý raz vedúcim v  $p$ -tom kroku, keď uvažujeme  $i, i+1, \dots, k$ , koľko krokov s vedúcim členom  $i-1$  sme vynechali? Ak si nevšímate čísla  $< i-1$ , číslo  $i-1$  bude vedúcim v každom 4. kroku. Takže sme vynechali  $(p-1) \div 3$  krokov.

**535.** Nie backtracking! Riešenie vždy existuje. Pokúste sa to vyriešiť pre  $n=1$ ,  $n=2$  a  $n=3$  a výsledky zovšeobecnite.

**541.** Postupne odčítavame od zvyšnej sumy jednotlivé mince a spočítavame koľkokrát sa nám podarilo sumu rozmeniť. Jednoduchšie sa to dá urobiť niekoľkými vnorenými cyklami, vtipnejšie použitím rekurzcie. Program s cyklami sa môžete snažiť ešte ďalej upravovať tým, že sa pokúsite spočítať explicitne súčty počítané v cykloch.

**542.** (zlé) Úlohu vieme ľahko vyriešiť pre interval  $\langle x_1, x_2 \rangle$ , na ktorom nemajú žiadne dve úsečky spoločný bod a každá úsečka obsahuje body  $\langle x_1, z' \rangle, \langle x_2, z'' \rangle$ , pre vhodné  $z', z''$ . Takže stačí zobrať intervaly, určené všetkými koncovými bodmi úsečiek a všetkými prienikmi úsečiek ( $x$ -ovými súradnicami týchto bodov.)

**543.** a) Využite, že ak viete vygenerovať všetky permutácie z  $i$  prvkov, potom viete vygenerovať ľahko aj z  $i+1$  prvkov. Stačí prvok  $i+1$  umiestniť na všetky možné pozície v každej permutácii z  $i$  prvkov. b) generujte lexikograficky po sebe nasledujúce permutácie. c) Existuje riešenie, že každú ďalšiu permutáciu dostaneme z predchádzajúcej jedinou výmenou [27].

**544.** a) Uvedomte si, že  $|uv| = |st|$ . Teda  $u$  stačí voliť dĺžky  $1, \dots, |uvst|/2$ . Tým ste určili aj  $v$  a ostáva overiť, či ste zvolili správne.  $O(n^2)$ , kde  $n$  je dĺžka celého reťazca. b) Šikovnejšie riešenie od Maja Dvorského je založené na hľadaní výskytu  $(uv)^R$  v reťazci  $stst$ . Dokážte, že

<sup>25</sup> Edsger W. Dijkstra, 1930-, holandský matematik, informatik a fyzik.

sa tam nachádza prave vtedy, keď má slovo uvst požadovaný tvar. Vyhľadávanie podreťazca sa dá realizovať lineárne od súčtu dĺžky vzorky a textu, teda  $O(n)$ .

**545.** Ak  $m > k$ , stačí nájsť takú postupnosť prevozov, ktorých výsledkom klesne o jedného počet misionárov aj kanibalov na jednom brehu. Ak  $m = k$ , tak pre  $m < 3$  je to ľahké, prípad  $m = 3$  je v zadaní a pre  $m > 3$  sa to nedá. Dokázať to môžete sporom.

**611.** Pozri 233. Výsledok sa dá zovšeobecniť.

**612.** Zakódujte vpravoľadiacich vojakov 1 a vľavoľadiacich 0. Teraz stačí v každom kroku nahradiť všetky dvojice 10 dvojicou 01. Pozor, otáčania treba vykonať „paralelne“. Otáčanie skončí vždy po  $n - 1$  krokoch. Skupina je  $k$  dvojíc  $k$  sebe otočených vojakov. Všimnite si, čo sa môže stať v nasledujúcom kroku. Podstatné je, že skupina sa buď zväčšuje, buď zostáva rovnako veľká, buď sa zmenšuje. Ak sa skupina začne zmenšovať, potom sa už len zmenšuje. Určite, maximálne koľko krokov môže skupina existovať.

**613.** Huffmanov<sup>26</sup> kód. Úloha sa dá redukovať na nájdenie binárneho stromu, ktorého súčet vážených ciest od koreňa k listom je minimálny. Znaky sú v listoch. Vážená cesta je súčin početnosti znaku a dĺžky cesty k nemu. Dĺžka cesty je rovná počtu hrán. Označme súčet všetkých vážených ciest  $c(S)$ . Algoritmus na nájdenie stromu  $S$  s minimálnym  $c(S)$  je založený na nasledujúcich dvoch tvrdeniach.

**Tvrdenie 1.** Ak  $p_1$  a  $p_2$  sú dve minimálne početnosti spomedzi  $p_1, p_2, \dots, p_k$  (ak je minimálnych viac, ľubovoľné z nich), existuje optimálny kódovací strom  $S$ , v ktorom sú listy  $L_1$  a  $L_2$  zodpovedajúce týmto písmenám bratia. Bratia sú listy, ktoré majú spoločného otca („čerešničky“). Dôkaz. Nech  $T$  je optimálny strom, v ktorom  $L_1, L_2$  nie sú bratia. Prerobíme ho na strom  $S$ ,  $c(S) = c(T)$ , v ktorom  $L_1$  a  $L_2$  budú bratia. Nech  $p_1 \leq p_2$ . Cesta vedúca k  $L_1$  nie je kratšia než cesta k  $L_2$ . Označme v  $T$  brata  $L_1$  ako  $L_3$ . Platí  $p_1 \leq p_2 \leq p_3$  a  $L_3$  je list. Ak vymeníme  $L_2$  a  $L_3$ , zrejme  $c(S) \leq c(T)$ , ale  $T$  je optimálny, takže platí rovnosť.

**Tvrdenie 2.** Ak nájdeme optimálny strom  $T$  pre početnosti  $p_1 + p_2, p_3, \dots, p_k$ , pričom  $p_1$  a  $p_2$  sú dve najmenšie, potom pre  $p_1, p_2, \dots, p_k$  je optimálny strom  $S$ , ktorý dostaneme z  $T$  nahradením listu zodpovedajúcemu znaku s početnosťou  $p_1 + p_2$  čerešničkami, bratmi zodpovedajúcimi znakom s početnosťami  $p_1$  a  $p_2$ . Dôkaz (sporom). Predpokladajme, že  $S'$  je pre  $p_1, \dots, p_k$  lepší ako  $S$ , teda  $c(S') < c(S)$ . Podľa Tvrdenia 1. vieme, že k  $S'$  existuje  $S''$  taký, že  $c(S'') \leq c(S')$ , ktorý má  $L_1$  a  $L_2$  ako bratov. Nahradením  $L_1$  a  $L_2$  dostaneme  $T''$  pre početnosti  $p_1 + p_2, p_3, \dots, p_k$  taký, že  $c(S'') = c(T'') + p_1 + p_2$ . Dostali sme  $c(T'') = c(S'') - p_1 - p_2 \leq c(S') - p_1 - p_2 < c(S) - p_1 - p_2 = c(T)$ , čo je spor.

**614.** a) (Backtracking) Asi najhoršie riešenie je postupne predlžovať cestu a keď sa už nedá, snažiť sa posledné predĺženie nejako zmeniť. Malé vylepšenie dosiahneme, keď cestu prestaneme predlžovať v prípade, že by bola dlhšia než doteraz minimálna, alebo ak by sme ju chceli predĺžiť na políčko, kadiaľ už vedie. b) Namiesto políčiek si predstavíme neorientovaný graf, v ktorom budú vrcholmi políčka pospájané podľa susednosti. Teraz stačí vedieť hľadať najkratšiu cestu v neorientovanom grafe. Dijkstrov algoritmus, pozri 523. c) Zaujímavé je aj riešenie, v ktorom stále prechádzame cez všetky políčka. V každom políčku je uložená dĺžka najkratšej cesty od políčka  $[1, 1]$ . Každému políčku skontrolujeme susedov a ak sa dá, zmenšíme dĺžku doteraz najkratšej cesty, ktorá doň vedie. Skončíme, ak sa nám nepodari zmeniť ani jednu hodnotu.

**615.** Aký by mal byť dobrý labyrint? Nemal by mať príliš dlhé rovné chodby, mal by mať dosť križovatiek a ideálne je, keď k pokladu vedie od vchodu len jedna cesta.

**621.** Označme  $c(y_{lk})$ ,  $0 < k \leq n$ ,  $1 \leq l \leq 2$ , minimálnu dĺžku cesty z bodu  $[x_0, 0]$  do bodu  $[x_k, y_{lk}]$ .  $c(y_{10}) = c(y_{20}) = 0$ ,

$$c(y_{lk}) = \min_{0 \leq j < k} \{c(y_{1j}) + |y_{1j}y_{lk}|, c(y_{2j}) + |y_{2j}y_{lk}|\}$$

<sup>26</sup> Huffman, David Albert, ???.1925-7.10.1999

**622.** Idea nasledujúceho riešenia je od Ilju Martišovitsša. Označme po sebe idúce vrcholy číslami 1 až  $n$ . Nech  $n \geq 3$ .

$$\begin{array}{ccc} v_1 & v_2 & v_n \\ v_1 + v_2 & v_2 + v_3 & v_n + v_1 \\ \vdots & \vdots & \vdots \\ v_1 + v_2 + \dots + v_n & v_2 + v_3 + \dots + v_n + v_1 & v_n + v_1 + \dots + v_{n-1} \end{array}$$

Nech je  $v_2 < 0$ . Pred aplikovaním operácie na po sebe idúce vrcholy máme:

$$\begin{array}{ccc} (1) & (2) & (3) \\ v_1 & v_2 & v_3 \\ v_1 + v_2 & v_2 + v_3 & v_3 + v_4 \\ \vdots & \vdots & \vdots \\ v_1 + v_2 + \dots + v_n & v_2 + v_3 + \dots + v_n + v_1 & v_3 + v_4 + \dots + v_n + v_1 + v_2 \\ & v_2 + v_3 + \dots + v_n + v_1 + v_2 & \end{array}$$

Po aplikovaní operácie dostaneme:

$$\begin{array}{ccc} (1') & (2') & (3') \\ (v_1 + v_2) & -v_2 & (v_2 + v_3) \\ (v_1 + v_2) - v_2 & -v_2 + (v_3 + v_2) & (v_2 + v_3) + v_4 \\ (v_1 + v_2) - v_2 + (v_3 + v_2) & -v_2 + (v_3 + v_2) + v_4 & (v_2 + v_3) + v_4 + v_5 \\ \vdots & \vdots & \vdots \\ (v_1 + v_2) - v_2 + \dots + v_n & -v_2 + (v_3 + v_2) + \dots + v_n + (v_1 + v_2) & (v_2 + v_3) + v_4 + \dots + v_n + (v_1 + v_2) - v_2 \\ & -v_2 + (v_3 + v_2) + \dots + v_n + (v_1 + v_2) - v_2 & \end{array}$$

Ak stĺpce porovnáme, zistíme, že sa zmenil iba prvý prvok v stĺpci (2'). Teda, že ubudlo jedno záporné číslo. Koľko riadkov musia mať stĺpce, aby sme mali istotu, že riadok, ktorý sme do stĺpca (2') pripísali, bude kladný? Zachráni nás predpoklad  $v_1 + \dots + v_n > 0$ . Odhadneme počet riadkov, ktorý bude isto stačiť. Označme  $s_{ij} = \sum_{k=i}^{j+n} v_k \bmod n+1$ , ak  $j < i$ , inak  $\sum_{k=i}^j v_k$ . Nech  $s_{\min} = \min\{s_{ij} \mid 1 \leq i, j \leq n\}$ . Nech  $s_{\min} < 0$  a  $s = \sum_{k=1}^n v_k$ . Potom isto  $s_{\min} + |s_{\min}|s \geq 0$ . Ešte nám ostali prípady  $n = 1, 2$ . Nech  $n = 1$ , ak  $v_1 < 0$ , úloha nemá riešenie, inak nie je čo riešiť. Nech  $n = 2$ , problematický je len prípad jedného záporného vrcholu, nech je to  $v_1$ . Po aplikovaní operácie dostaneme  $-v_1, v_2 + 2v_1$ , ak  $v_2 + 2v_1 < 0$ ,  $v_1 < v_2 + 2v_1$ , takže po konečnom počte aplikácií operácie budú oba vrcholy nezáporné.

**623.** Určite súradnice ľavého horného a pravého dolného vrcholu prieniku obdĺžnikov. Označme  $X_{A \cup H}, Y_{A \cup H}, X_{A \cap D}, Y_{A \cap D}$  súradnice ľavého horného a pravého dolného vrcholu obdĺžnika A a analogicky pre B. Ak majú A a B neprázdny prienik, je to obdĺžnik so súradnicami vrcholov,  $X_{\cup H} = \max\{X_{A \cup H}, X_{B \cup H}\}$ ,  $X_{\cap D} = \min\{X_{A \cap D}, X_{B \cap D}\}$ ,  $Y_{\cup H} = \min\{Y_{A \cup H}, Y_{B \cup H}\}$ ,  $Y_{\cap D} = \max\{Y_{A \cap D}, Y_{B \cap D}\}$ . Ich prienik je prázdny práve vtedy, keď  $X_{\cup H} - X_{\cap D} > 0$  a/alebo  $Y_{\cap D} - Y_{\cup H} > 0$ .

**624.** a) Deliteľov zistíme ako pri testovaní prvočíselnosti; preveríme všetky čísla z intervalu  $\langle 1, \sqrt{x} \rangle$ . b) Nech je prvočíselný rozklad  $x = p_1^{a_1} p_2^{a_2} \dots p_k^{a_k}$ . Označme súčet všetkých deliteľov čísla  $x$ ,  $SVD(x)$ . Platí  $SVD(x) = \frac{p_1^{a_1+1}-1}{p_1-1} \frac{p_2^{a_2+1}-1}{p_2-1} \dots \frac{p_k^{a_k+1}-1}{p_k-1} =$



$(1+p_1+p_1^2+\dots+p_1^{a_1}) \cdot \dots \cdot (1+p_k+p_k^2+\dots+p_k^{a_k})$  a  $s(x) = SVD(x) - x$ . c) à la Eratostenovo<sup>27</sup> sito. Použijeme pole  $S$ ,  $S[i] = s(i)$ .  $S[i]$  určíme podobne ako pri metóde Eratostenovho sita na hľadanie prvočísel, ale namiesto vyškrtávania deliteľov ich budeme sčítovať. Takže si predvypočítame pole  $S$  a pre dané  $x$  pokiaľ môžeme, ho využijeme, inak použijeme a) alebo b). Prvých desať spriateľných dvojíc: (220, 284), (1184, 1210), (2620, 2924), (5020, 5564), (6232, 6368), (10744, 10856), (12285, 14595), (17296, 18416), (63020, 76084), (66928, 66992).

**625.** Je šikovné najprv z analyzovaného riadku vymazať všetky medzery už pri jeho čítaní. V programe môžeme čítať buď číslo, alebo oddeľovač, alebo identifikátor. Problematický je len identifikátor, lebo ten môže byť ešte kľúčovým slovom. Uvedomte si, že AORC sa musí rozložiť na A, OR a C. Keď začneme čítať identifikátor, musíme kontrolovať, či jeho nejaká podčasť nie je kľúčovým slovom a ak je, načítali sme identifikátor a kľúčové slovo. Ak sa pri čítaní identifikátora vyskytne číslica, potom si všimame iba číslice a sú súčasťou identifikátora. Číslo môže začínať iba po kľúčovom slove alebo oddeľovači. Aby sme kľúčové slová ľahko našli, je výhodné si ich pamätať v tabuľke a v pomocnom poli indexovanom písmenami si pamätať počiatočný a koncový index kľúčových slov začínajúcich daným písmenom. Pozorne treba vyriešiť ešte viacznakové oddeľovače.

**631.** Nech  $a(x) = a_n x^n + \dots + a_0$ ,  $b(x) = b_m x^m + \dots + b_0$  a  $a_n \neq 0$ ,  $b_m \neq 0$ . Potom  $\frac{a(x)}{b(x)} = q(x) + r(x)$  a  $q(x) = a(x) \operatorname{div} b(x) = \frac{a_n}{b_m} x^{n-m} + (a(x) - \frac{a_n}{b_m} x^{n-m} b(x)) \operatorname{div} b(x)$ ,  $r(x) = a(x) \bmod b(x) = (a(x) - \frac{a_n}{b_m} x^{n-m} b(x)) \operatorname{div} b(x)$  ak  $n \leq m$  a  $r(x) = a(x)$  ak  $n > m$ .

**632.** Minimálny počet skokov, ak je  $n$  párne, je  $n(n+1)/2$  a ak je nepárne, je  $n(n+3)/2 - 4$ .

**633.** Dá sa vždy nakresliť jedným ťahom (Eulerov<sup>28</sup> ťah). Pozor! Nasledovný algoritmus je zlý: Začneme s krokom  $k = 2$ . Z prvého vrcholu kreslíme uhlopriečky s krokom  $k$ , až pokiaľ sa sem nevrátime, vtedy sa posunieme do susedného vrcholu. Ak sme odtiaľto už kreslili s krokom  $k$ ,  $k$  o jedna zväčšíme a pokračujeme z tohoto vrcholu rovnakým spôsobom ďalej, až pokiaľ nevykreslíme všetky uhlopriečky s krokom  $(n-1) \operatorname{div} 2$ . Vtedy prejdeme zvyšné vrcholy s krokom  $k = 1$ . Najmenší počet vrcholov, keď tento algoritmus zlyhá, je 24. Problém je v tom, že začneme znovu chodiť po už nakreslených čiarach. Súvisí to so vzťahom medzi  $n$  a súčtom najväčších spoločných deliteľov čísel 1 až  $(n-1) \operatorname{div} 2$ .

Správny algoritmus je takýto: Z každého vrcholu nakreslíme všetky uzavreté cik-caky s krokmi  $k = 2, 3, \dots, (n-1) \operatorname{div} 2$ . Potom sa posunieme do susedného vrcholu. Problém je, s ktorým krokom treba cik-caky z tohoto vrcholu domaľovať. To sa dá riešiť pomocným polom  $n$  bitov pre každý vrchol. Lepšie je však využiť, že rôzne uzavreté cik-caky začínajú vo vrcholoch 1, ...,  $\operatorname{nsd}(n, k)$ .

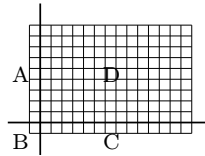
**634.** Nezáleží na poradí prvkov, zapíšeme ich v rastúcom poradí. Označme  $g_n(i, k)$  počet  $k$ -prvkových podmnožín množiny  $\{i, i+1, \dots, n\}$  začínajúcich prvkom  $i$ . Zrejme  $g_n(i, k) = \binom{n-i}{k-1}$ . Algoritmus môžeme zapísať takto

1.  $s := x + 1$ ;  $i := 1$ ;
2. ak  $g_n(i, k) \geq s$ , ďalší prvok podmnožiny je  $i$ ;  $i := i + 1$ ;  $k := k - 1$ ;
3. ak  $g_n(i, k) < s$ , tak  $s := s - g_n(i, k)$ ;  $i := i + 1$ ;
4. ak  $k = 0$ , sme hotoví, inak pokračujeme bodom 2.

Zistíte, ako šikovne vypočítať hodnoty  $g_n(i, k)$  v 2. a 3. kroku.

**635.** Pozri [VIII.5.1.3 v 31], riešenie použitím segmentových stromov na  $O(n \log n)$ .

**641.** Určite, aké veľké pole v závislosti od počiatočnej konfigurácie isto stačí na simulovanie. Prefarbovanie skončí vždy po  $n$  krokoch. Pri dôkaze si rozdeľte najmenší pravouholník, v ktorom leží celý obrazec, na štyri časti A, B, C, D.



<sup>27</sup> Eratostenes Cyrénsky, 276-194 p.n.l., grécky učenec.

<sup>28</sup> Leonhard Euler, 1707-1783, švajčiarsky matematik, fyzik, mechanik a astronóm.

**642.** Rozdeľte  $M$  na  $M_1$  a  $M_2$  také, že  $M_1$  obsahuje len nepárne a  $M_2$  len párne prvky. Takto máme zaručené, že medzi prvkom z  $M_1$  a prvkom z  $M_2$  nebude ich aritmetický priemer. Takže stačí rekurzívne preusporiadať  $M_1$  a  $M_2$ . Preusporiadavanie prvkov sa dá robiť ľahko, keď máme k dispozícii dvojkovú reprezentáciu prvkov, stačí ich preusporiadať podľa hodnoty  $k$ -teho bitu.

**643.** Začneme z ľubovoľného vrchola  $v_0$ . Po čase sa do  $v_0$  musíme vrátiť (prečo?). Ak sme nakreslili celý obrázok, skončíme. V opačnom prípade nám ostali nenakreslené časti. Tieto majú vždy aspoň jeden spoločný vrchol už s nakreslenou časťou. Nakreslíme ich rovnakým postupom ako pôvodný obrázok, pričom začneme zo spoločného vrcholu. Celý obrázok nakreslíme jedným ťahom tak, že vždy, ak môžeme, „odbočíme“ do neskôr nakreslených častí. Algoritmus vyžaduje  $O(n + m)$  operácií, kde  $n$  je počet vrcholov a  $m$  počet hrán grafu, pozri napríklad [9.A v 34, 2.7 v 27].

**644.** Úloha sa dá riešiť podobnou technikou ako pivotizácia. Pole počas riešenia môžeme udržiavať v tvare biele, modré, červené a zatiaľ neupratané, alebo v tvare biele, modré, zatiaľ neupratané a červené, pričom si pamätáme konce, alebo začiatky skupín. Šikovnejší je druhý tvar. Skúste úlohu vyriešiť jedným cyklom, pričom si budete všímať prvý prvok zo zatiaľ neusporiadaných krabíc a podľa jeho farby ho dávať na jeho miesto. Keď to urobíte šikovne, cyklus prebehne najviac  $m$  krát a v každom cykle urobíte najviac jednu výmenu krabíc. Nech  $j$  a  $k$  sú indexy prvej a poslednej zatiaľ neusporiadanej krabice, všimnite si rozdiel  $k - j$ .

**711.** Uložme konce intervalov do poľa  $x$ . Nech  $x[i].h = a_i$ ,  $x[i].t = A$ ,  $x[i + n].h = b_i$  a  $x[i + n].t = B$ . Všimnite si, že pole je utriedené vzostupne podľa kľúča  $t$ . Pole  $x$  utriedime stabilným triediacim algoritmom<sup>29</sup> vzostupne podľa hodnoty kľúča  $h$ . Teraz stačí prejsť utriedené konce intervalov a počítať si, v koľkých intervaloch sa práve nachádzame. Vypíšeme len tie intervaly, ktoré susedia s intervalmi obsahujúcimi body ležiace v nepárnom počte intervalov. Uvedomte si, že pri prezeraní nezáleží na tom, ktorému intervalu bod patrí. Dôležité je len, či je počiatočný, alebo koncový, teda hodnota  $t$ .

**712.** a) Z každého voľného políčka sa pohneme na štyri strany, samozrejme, ak je to možné. Začneme v  $[1, 1]$ , ak sa dostaneme do  $[n, n]$ , sme hotoví. Toto riešenie sa ľahko programuje využitím rekurzcie. b) Spojenie existuje práve vtedy, keď neexistuje cesta po zničených uzloch z horného alebo pravého okraja k dolnému alebo ľavému. Takže rovnako ako a), ale po zničených uzloch. Susednosť políček v tomto prípade musíme rozšíriť aj na políčka po uhlopriečkach.

**713.** a) Číslo „rozbalíme“ tak, aby žiadna menšia cifra nepredchádzala väčšiu. Pri sčítavaní rovnaké cifry dávame spolu. Na koniec treba ešte upraviť tie cifry, ktoré sa dajú zapísať špeciálne (IV, IX, XL, XC, CD, CM). Pri odčítavaní postupujeme podobne, len cifry vyškrtávame a keď treba, „rozmeníme“ väčšiu cifru na menšie. b) Lepšie riešenie je rozložiť číslo do poľa  $C[1..7]$ ,  $C[i]$  určuje počet výskytov jednotlivých cifier. Ak sa menšia cifra vyskytuje pred väčšou, je v poli reprezentovaná  $-1$ . Pri sčítovaní a odčítavaní stačí sčítat prípadne odčítat príslušné prvky poľa a výsledok upraviť tak, aby sa vo výsledku nevyskytovali symboly V, L a D viac než raz a symboly I, X a C viac než tri razy.

**714.** Najšikovnejšie je vypísať všetky najpravdepodobnejšie správy. Netreba si pamätať celý text, stačí pole  $p[1..k, 1..26]$  (dĺžka správy  $\times$  počet písmen abecedy),  $p[i, z]$  určuje počet výskytov znaku  $z$  na pozícii  $i$ . Uvedomte si, na to, aby sa niektoré písmeno nachádzalo na  $i$ -tom mieste, musí sa najviackrát vyskytovať na tých miestach, kde by bolo  $i$ -te písmeno správy. Pre text v zadaní boli správy NIC a SEMINAR.

**715.** a) Snažte sa pridávať kamene s čo najväčším číslom v dolnej polovici a čo najmenším v hornej. b) Backtracking. Úloha sa nazýva Postov<sup>30</sup> korešpondenčný problém [14.2 v 18].

<sup>29</sup> Zachováva poradie vzhľadom na už predtým utriedené kľúče.

<sup>30</sup> Emil, Leon Post, 1897-1954, americký logik a matematik.

**721.** a) Podpostupnosti prezeráme od najdlhšej všetky a počítame, či ich aritmetický priemer je  $\geq p$ . Vyžaduje približne  $O(n^3)$  operácií. b) Spočítame čiastočné súčty  $s_0 = 0$ ,  $s_i = s_{i-1} + a_i$ , pre  $0 < i \leq n$ . Priemer podpostupnosti  $a_i, \dots, a_j$  je  $(s_j - s_{i-1})/(j - i + 1)$ . Čo možno upraviť takto: hľadáme maximálnu hodnotu  $j - i$ , aby súčasne  $s'_j - s'_{i-1} \geq 0$ , kde  $s'_i = s_i - ip$ .  $O(n^2)$  operácií.

**722.** a) Ak  $\text{nsd}(a_i, b_j) \neq 1$  pre niektoré  $1 \leq i \leq n$  a  $1 \leq j \leq m$ , vydelíme ním  $a_i$  aj  $b_j$ . b) Využijeme, že  $a_i, b_j \leq 500$ . Čitateľ aj menovateľ stačí rozložiť na prvočinitele, na to stačí deliť  $a_i, b_j$  prvočíslami  $\leq \sqrt{500}$ , t.j. 2, 3, 5, 7, 11, 13, 17, 19. Pri rozklade čitateľa pripočítavame k výskytu príslušného prvočísla 1, pri rozklade menovateľa zasa  $-1$ .

**723.** Rozloženie lajstier všetkých úradníkov označme  $x = (x_1, \dots, x_n)$ . Stačí vyriešiť sústavu lineárnych rovníc:

$$\begin{aligned} \frac{x_1 u_{11}}{100} + x_2 \left( \frac{u_{21}}{100} + 1 \right) + \dots + x_n \left( \frac{u_{n1}}{100} + 1 \right) &= 100000000 \\ &\vdots \\ x_1 \left( \frac{u_{1n}}{100} + 1 \right) + x_2 \left( \frac{u_{2n}}{100} + 1 \right) + \dots + x_n \frac{u_{nn}}{100} &= 100000000 \end{aligned}$$

**724.** Dôležité je uvedomiť si, že ide len o zistenie poradia zjednocovania množín vzhľadom na minimalizovanie počtu operácií, a nie o reprezentovanie prvkov zjednotených množín. Množiny stačí usporiadať podľa počtu prvkov a zjednocovať vždy dve množiny s najmenším počtom prvkov. Všimnite si, že novovzniknuté množiny sú usporiadané podľa veľkosti, takže si ich stačí dávať bokom. Nájdienie dvoch množín s najmenším počtom prvkov teda vyžaduje iba konštantný počet operácií, nezávislý od  $n$ .

**725.** Uvedomte si, že viacnásobný výskyt  $*$  nemá zmysel a môže sa odstrániť. Znak  $?$  nemá zvláštny význam. Ak si vstupný reťazec zapíšeme ako  $a_0 * a_1 * a_2 * a_3 \dots * a_i \cdot a_{i+1} * a_{i+2} \dots * a_j$ , kde  $a_k$ ,  $0 \leq k \leq j$ , obsahuje iba písmená abecedy alebo znak  $?$ . Všimnite si, že  $a_0$  a  $a_j$  majú v slove pevné miesto a ľahko overíte, či sa v mene nachádzajú. Takže nám ostane riešiť prípad, keď slovo začína aj končí  $*$ . V mene stačí nájsť výskyt  $a_1$  a vieme, čo sa schová pod  $*_1$ . Vzor môžeme zmenšiť a pokračujeme rovnakým spôsobom ďalej. Je jasné, kedy bude meno spĺňať vzor a kedy nie. Otázne ostáva, či by sa nám nemohlo stať, že v mene nájdeme najľavejší výskyt  $a_k$ , v dôsledku čoho nám vyjde, že meno nespĺňa vzor. Ale keby sme pod  $*_k$  „schovali“ viac, podarilo by sa nám ukázať, že meno vzoru vyhovuje. Skúste si dokázať, že ozať stačí uvažovať iba najľavejší výskyt. Vyhľadávanie podreťazca je v 1222.

**731.** Backtracking. Je dobré využiť, že ak nie je súčet políčok patriacich miestnosti deliteľný štyrmi, nemá zmysel sa o nič pokúšať. Miestnosť môžeme začať parketovať hociak a je výhodné použiť okolo miestnosti plôtk z nulových políčok. Je šikovné postupovať z horného ľavého rohu do pravého dolného po ešte voľných políčkach zľava doprava a zapamätať si všetkých 19 rôznych polôh tetramín, prípadne ich relatívne súradnice, vzhľadom na ich ľavý horný roh.

**732.** Pre jednoduchosť si celý výraz vložíme do jednej zátvorky. Operátory a ľavé zátvorky je potrebné ukladať do zásobníka, operandy rovno vypisujeme na výstup. Ak by mala nastať situácia, že druhý operátor z vrchu zásobníka má väčšiu alebo rovnakú prioritu ako ten, čo je navrchu, vypíšeme ho a vyhodíme zo zásobníka (výnimku tvorí len operátor mocniny, kde sa rovnaké operátory zo zásobníka nevytláčajú). Pri načítaní pravej zátvorky je potrebné vypísať obsah zásobníka až po najbližšiu ľavú zátvorku a tú zo zásobníka vyhodíť. Špeciálne treba ošetriť unárne mínus.

**733.** a) Nájdeme prvého maniaka, ktorého sme ešte nevypísali. Vypíšeme všetkých mania- kov, ktorých telefónne čísla vie a ktorí ešte neboli vypísaní. S každým, ktorého vypíšeme,

vykonáme rovnakú operáciu (možno realizovať rekurzívne, ale aj nerekurzívne). Keď vypíšeme týmto spôsobom všetkých, na ktorých ukazoval prvý, hľadáme ďalšieho nevypísaného a opakujeme celý proces. Časová zložitosť tohto algoritmu je  $O(n^2)$ . b) Union-Find.

**734.** Zisťovať maximálny cólštok pre všetky možné prvočíselné počty ramien (ak počet ramien  $p$  je zložené číslo, mohli sme nájsť cólštok rovnakej dĺžky s menším počtom ramien takým, že delí  $p$ ). Počet ramien je najmenej dva a najviac sa rovná maximálnemu počtu rovnakých znakov. Nájsť maximálny cólštok znamená nájsť minimálny počet nepoužitých znakov. Ich počet sa v prípade cólštoku prvého typu vypočíta triviálne, je rovný súčtu zvyškov, ktoré dávajú počty jednotlivých znakov po delení počtom ramien. V prípade cólštoku druhého typu je potrebné rozdiskutovať niekoľko prípadov rozmiestňovania znakov.

**735.** Pre každú dvojicu bodov zistíme, či ich spojnice prechádza tučnou úsečkou. Rovnica priamky prechádzajúcej cez body  $[x_1, y_1], [x_2, y_2]$  je  $(y_2 - y_1)x + (x_1 - x_2)y + x_2y_1 - x_1y_2 = 0$ . Ak do ľavej strany za  $x, y$  dosadíme dva body a dostaneme čísla s opačnými znamienkami, tak tieto dva body ležia v opačných polrovinách vzhľadom na túto priamku. Túto myšlienku môžeme využiť pri zisťovaní, či sa dve úsečky pretínajú. Je potrebné ošetriť pomerne veľa okrajových prípadov, ako napríklad body ležiace na tučnej úsečke alebo spojnice bodov, ktorá sa úsečky dotýka.

**811.** Ak je súčet  $k$  najmenších prvkov menší než  $t$ , tak ANO, inak NIE. Šikovné je nájsť  $k$  najmenších prvkov tak, že odhadneme, aké by mali byť veľké. Napríklad najprv skúsime sčítať  $k$  prvkov menších než  $t/k$ , ak sme ich našli  $k_1 < k$  a ich súčet bol  $s_1$ , skúsime nájsť  $k - k_1$  prvkov menších než  $(t - s_1)/(k - k_1)$ , atď. Skúste si dokázať, že takto vybrané prvky sú riešením.

**812.** Komplikovaný príklad. Uvedieme iba orientačný návod. a) Napište si, čo má platiť pre prvky, dostanete sústavu nerovníc. Koľko ich bude, aby bolo neznámych rovnako ako nerovníc? Teraz treba nájsť riešenie čo najväčšej sústavy nerovníc (rovníc). b) Dokázať, že hľadaná postupnosť má dĺžku  $d \leq m + n - 1 - \text{nsd}(m, n)$ . Keď  $m = n$  je to ľahké. Predpokladajte, že  $m < n$ . Rozlíšte prípady  $\text{nsd}(m, n) = 1$  a  $\text{nsd}(m, n) \neq 1$ . Keď vieme dĺžku postupnosti, už ju ľahko určíme.

**813.** Táto úloha je klasická úloha z teórie grafov, nazýva sa úloha o najlacnejšej kostre a má dve štandardné riešenia založené na tzv. pažravej (greedy) metóde a) Kruskalov algoritmus. Utriedime podľa veľkosti vzdialenosti miest a postupne spájame najbližšie mestá tak, aby sme nevytvorili spojeniami cyklus. Na triedenie potrebujeme  $O(n^2 \log n)$  a na kontrolu, či sme nevytvorili kružnicu  $O(n^2)$  operácií, teda preváži triedenie. b) Primov algoritmus. Mestá rozdelíme na už spojené a ešte nespojené. Na začiatku je spojené iba jedno mesto (ľubovoľné, samo zo sebou). Do skupiny spojených pridávame vždy to mesto, ktorého vzdialenosť od niektorého už spojeného mesta je minimálna. Pre každé pridané mesto musíme aktualizovať minimálne vzdialenosti do ešte nespojených miest. Nájsť najlacnejšieho a aktualizácia nás stojí  $O(n)$  operácií a musíme prejsť všetky mestá, celkovo  $O(n^2)$  operácií. Pozri napríklad [5.B v 34; 6.2 v 25; kap. 24 v 6; 7.6 v 30; 5.1 v 1].

**814.** Topologické triedenie. a) Pre každého spočítame, koľko má podriadených (podriadení sa myslia všetci a nielen bezprostrední podriadení). Ak je niekto sám sebe aj nadriadený aj podriadený, želané usporiadanie neexistuje. b) Šikovnejšie je hľadať tých, čo nemajú nadriadených. Ak taký nik nie je, úloha nemá riešenie. Ak takého nájdeme, môžeme ho vypísať a jeho priamym nasledovníkom zmenšíme počet predchodcov. Takto postupujeme, pokým nevypíšeme všetkých vojakov, alebo nezistíme, že úloha nemá riešenie.

**815.** Riešenie je priamočiare. Skontrolujeme, kto má viac celých boxov. Vhodné je použiť také zakódovanie farieb, aby sme ľahko vedeli zistiť, či sa nedoplnený box dá doplniť na jednofarebný. Napríklad biely bude 1, čierny bude 5 a nezafarbené políčko bude 0.

**821.** Je nešikovné najprv vyrábať celú krivku (prípadne ju uložiť do poľa) a potom vypísať len želanú časť. Celá krivka môže byť omnoho väčšia než hľadaná časť. Lepšie je pre každý

bod  $(x, y)$  patriaci do skúmaného obdĺžnika určiť, či obsahuje \* alebo medzeru. Uvedomte si, že krivka stupňa  $n$  sa skladá zo štyroch kriviek stupňa  $n-1$  a z troch \*, ktoré ich spájajú. Stačí teda rekurzívne určovať, do ktorej zo štyroch kriviek menšieho stupňa padne bod  $(x, y)$ , prípadne či nepadne do oblastí medzi krivkami. Treba pritom dávať pozor, lebo ako vidno zo zadania, krivky menšieho stupňa môžu byť otočené. Krivka stupňa 1 je \*. Celý postup sa dá ešte zefektívniť, keď si všimneme, že body, ktorých obe súradnice sú párne, sú medzery a body s oboma súradnicami nepárnymi sú \*. Skúste si dokázať prečo.

**822.** a)  $O(n^2)$ ; priamočiare riešenie je postupne kontrolovať body a zistiť, či všetky ostatné ležia v jednej polrovine. b)  $O(n \log n)$ ; body utriedime podľa uhlu, ktorý zviera ich spojnica s bodom  $[0, 0]$  s  $x$ -ovou osou. Ak je medzi niektorými dvoma po sebe idúcimi bodmi uhol väčší než  $180^\circ$ , hľadaná priamka existuje, inak nie. c)  $O(n)$ ; zistíte maximálne a minimálne  $x$ -ové súradnice bodov s kladnou a zápornou  $y$ -ovou súradnicou a rozmyslite si, čo pre ne musí platiť, aby úloha mala riešenie.

**823.** Zrátame minimálne vzdialenosti medzi všetkými dvojicami miest (pozri úlohu 523), treba na to  $O(n^3)$  operácií. A teraz stačí zrátať pre každé mesto súčet vzdialeností do ostatných a vybrať mesto, kde je tento súčet minimálny. Takže celkovo potrebujeme na realizáciu  $O(n^3)$  operácií.

**824.** Ukazuje sa, že treba postupovať podľa pravidiel, maximalizovať hmotnosť a počet. Toto pravidlo bezchybne funguje pri pribaľovaní do balíkov hmotností 8, 7, 6, 5, 3, 2 a 1kg, ale nie pre 4kg balík. V tomto prípade môžeme vyskúšať dve možnosti: pridať 2+2kg, alebo 3+1kg a zobrať lepšiu. Pokúste sa nájsť príklad, v ktorom všeobecné pravidlo neplatí (napríklad 4kg, 3kg, 3kg, 2kg, 2kg, 2kg sa dajú zabaliť do dvoch balíkov). Šikovnejšie je v prípade nepárneho počtu 4kg výrobkov jeden nechať, zabaliť najprv všetky 3kg výrobky a zabaliť ho až potom.

**825.** Pozor na mnoho okrajových podmienok. V prípade, že nebol vytvorený box, je situácia korektná, ak sa počet ťahov bieleho a čierneho líši najviac o jedna. Ak boli ale vytvorené boxy, podľa pravidiel ten, kto svojou čiarou vytvoril box, dostal ťah navyše. Ale pozor, ak sa mu podarilo vytvoriť jednou čiarou boxy dva, rovnako dostal iba jeden extra ťah. Podľa pravidiel hra končí aj vtedy, keď sa už nedá vytvoriť žiaden box. Ak v takejto situácii vykonáme ťah, dostaneme nekorektnú situáciu. No a teraz už neostáva nič iné, iba metódou spätných návratov postupne generovať jednotlivé priebehy hry.

**831.** Problém union-find. a) Najjednoduchšie je pamätať si v poli pre mesto  $i$  na tomto indexe číslo časti, kam dané mesto patrí. Na začiatku predpokladáme, že sú všetky mestá izolované. Postupne, ako čítame dvojice, ktoré ostali spojené, zistíme, či sú v rovnakých častiach. Ak nie sú, časti spojíme (jednu z nich prečísľujeme). Pretože nevieme, ktoré a koľko miest patrí do ktorej časti, vždy musíme skontrolovať všetky mestá. Na spojenie  $n$  oblastí potrebujeme v najhoršom  $O(n^2)$  operácií. b) Ak si pamätáme, ktoré mestá a kam patria, dosiahneme lepší výsledok. Stačí si pre každú časť udržiavať zoznam miest, ktoré sem patria a pre každé mesto si pamätať, do ktorej oblasti patrí. Teraz keď spájame dve oblasti do jednej, nemusíme mestá hľadať, ale stačí len príslušné zoznamy spojiť a prepísať číslo skupiny pripojeným mestám. Šikovne sa to dá urobiť, keď využijeme veľkosti skupín a pripájať budeme vždy skupinu s menším počtom miest. Kolkokrát potom môže jedno mesto zmeniť skupinu? (Aká veľká bude skupina po spojení?). Na  $n$  spojení treba  $O(n \log n)$  operácií. Ešte lepšie riešenie sa dá nájsť v [3.8 v 25, 4.6–8 v 1, 4.5 v 30, 22 v 6].

**832.** a) Najjednoduchšie a najmenej šikovné je skontrolovať všetky štvorice dier, tie určujú nejaký štvoruholník. Spomedzi nich treba vybrať najväčší neobsahujúci ďalšie diery. Všetkých štvoríc je  $O(k^4)$ , teda celkovo toto riešenie vyžaduje  $O(k^5)$  operácií. b) Lepšie riešenie vyžadujúce v najhoršom prípade len  $O(k^3)$  operácií. Najprv utriedime diery podľa rastúcej  $x$ -ovej súradnice. Prvé priblíženie k hľadanému obdĺžniku zoberieme najväčší obdĺžnik bez

dier, ktorého ľavá strana je na ľavom okraji dosky. Ten vieme nájsť ľahko. Budeme si pamätať, akú môže mať hľadaný obdĺžnik najväčšiu a najmenšiu y-ovú súradnicu niektorého vrchola, označíme si ich  $h$  a  $d$ . Postupne preberáme usporiadané diery a vždy zoberieme do úvahy obdĺžnik, ktorý má na tejto diere pravú stranu a rekurzívne nájdené najväčšie obdĺžniky s minimálnou a maximálnou y-ovou súradnicou, rovnakou ako má spracovávaná diera (ktoré diery vôbec nemusíme spracovávať?). Ešte treba prezrieť obdĺžniky s ľavou stranou od  $i$ -tej diery a pravou stranou po  $j$ -tu dieru. Tu postupujeme podobne ako pri hľadaní obdĺžnika so stranou na kraji dosky, ale netreba skúmať rekurzívne obe časti, ale stačí upraviť  $h$  alebo  $d$ , podľa toho, či je spracovávaná diera pod  $h$  alebo nad  $d$ .

**833.** a) Nie backtracking! b) Vtipné riešenie je použiť pomocné pole veľkosti  $m \times n$  a uložiť doň postupne pre každé písmeno návrhu  $A$  indexy jeho výskytov v návrhu  $B$ . Potom nájsť v tomto poli najdlhšiu rastúcu podpostupnosť (pozri úlohu 522), to budú indexy písmen najdlhšieho spoločného návrhu. c) Označme  $L_{i,j}$  dĺžku najdlhšieho spoločného návrhu z častí  $A[1..i]$  a  $B[1..j]$ . Chceme nájsť  $L_{m,n}$ . Ako určiť hodnoty  $L_{i,j}$ ?  $L_{i,0} = L_{0,j} = 0$ , keď  $0 \leq i \leq n$  a  $0 \leq j \leq m$ .  $L_{i,j}$  určíme na základe  $L_{i-1,j}$ ,  $L_{i,j-1}$  a  $L_{i-1,j-1}$ . Ak  $A[i] = B[j]$ , tak  $L_{i,j} = L_{i-1,j-1} + 1$  a v prípade  $A[i] \neq B[j]$ ,  $L_{i,j} = \max\{L_{i-1,j}, L_{i,j-1}\}$ . Ešte treba znaky najdlhšieho spoločného návrhu. Tie určíme od posledného tak, že prezeráme hodnoty poľa  $L$  a všimame si, kde hodnota  $L$  klesá.

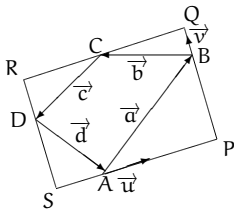
**834.** Hľadanie najpočetnejšieho párenia v bipartitnom grafe, pozri v [34].

**835.** Uvedomte si, že namiesto štvoríc  $x_i, y_i, u_i, v_i$  stačí uvažovať dvojice  $p_i, q_i$ , pričom prvá zložka je kód  $x_i, y_i$  a druhá zložka je kód  $u_i, v_i$ . Treba vlastne zistiť, či počnúc niektorou dvojicou sa dajú postupne použiť ďalšie dvojice tak, aby sa v nich vyskytli všetky štáty, vtedy môže vypuknúť svetová vojna. Ak začneme s dvojicou  $s, t$ , môžeme použiť dvojice  $t, *$ . Namiesto  $*$  môžeme dosadiť hocíčo. Dá sa to napríklad tak, že začneme z ľubovoľnej dvojice  $p_1, q_1$ , pokračujeme výberom dvojíc, pokiaľ sa to dá, a ak sme neprešli ešte všetky štáty, pokúsime sa nájsť dvojicu  $*, p_1$  a pokračujeme ako predtým. Ak použijeme všetky dvojice a neskončili sme, svetová vojna nebude.

**841.** Zrejme trasa bude prechádzať všetkými stanovišťami, lebo by sme ju inak vedeli predĺžiť a z rovnakého dôvodu bude na jednom stanovišti cieľ aj štart. Všimnite si, že najvzdialenejšie stanovišťa nemusia byť na najdlhšej trase: súradnice stanovišť napríklad  $[-100, 0]$ ,  $[0, 100]$ ,  $[0, 100]$ ,  $[0, 90]$ . Táto úloha patrí medzi tzv. ťažké a má aj svoje meno v literatúre: úloha obchodného cestujúceho. Riešenie spočíva len v postupnom prezretí všetkých možných trás, ktorých je  $n!$  (generovanie permutácií je v úlohe 543).

**842.** Minimálny počet presunov je  $2^{n+2} - 5$ . Najprv vyriešte jednoduchšiu úlohu, keď sú rovnaké disky nerozlíšiteľné. Vtedy treba  $2^{n+1} - 2$  presunov (je to dvakrát viac ako v originálnom hlavolame ale len s  $n$  diskami). Uvedomte si, že keď presuniete vežu s nerozlíšiteľnými rovnakými diskami, tieto sú v opačnom poradí. Takže po dvoch presunutíach sú opäť v pôvodnom usporiadaní.

**843.** Vzhľadom na to, že body máme zadané proti smeru chodu hodinových ručičiek, môžeme použiť napríklad časť Grahamovho algoritmu na vylúčenie bodov, ktoré do konvexného obalu nepatria, pozri úlohu 413.



**845.** Riešenie od Ladislava Kisa. Obdĺžnik s minimálnym obsahom sa dá nájsť, len treba použiť trochu matematiky. Každý opísaný štvoruholník, ktorý je kandidátom na minimálny, sa iste bude dotýkať na každej svojej strane aspoň jedného z daných vrcholov (inak by sa dal zmenšiť). Pre jednoduchosť si predstavme, že máme zadane iba štyri vrcholy  $A, B, C, D$ . Zrátame obsah  $S$  pravouholníka  $PQRS$  v závislosti od vektora  $\vec{u}$ .

Platí, že  $S = |SP||PQ|$ , kde  $|SP| = |AS| + |AP|$  a  $|AP| = (\vec{a} \cdot \vec{u})/\|\vec{u}\|$  a analogicky aj  $|AS|$ ,  $|BQ|$  a  $|BP|$ . Prípád  $\vec{u} = (0, z)$  skontrolujeme osobitne. Keď položíme  $\vec{u} = (1, z)$ ,  $\vec{r} = \vec{a} + \vec{d} = (r_1, r_2)$  a  $\vec{s} = \vec{a} + \vec{b} = (s_1, s_2)$ , dosadíme do vzťahu pre  $S$  a roznásobíme, dostaneme  $S(z)$  vzhľadom na parametre  $\vec{s}$  a  $\vec{r}$ , ktoré sú však konštantné. Zaujímá nás, kde má  $S(z)$  maximum. Zrátame  $S'(z) = 0$  a dostaneme  $z_1 = -q + \sqrt{q^2 + 1}$  a  $z_2 = -q - \sqrt{q^2 + 1}$ , kde  $q = (r_1 s_2 + r_2 s_1)/(r_2 s_2 - r_1 s_1)$ . Dve riešenia  $z_1$  a  $z_2$  predstavujú navzájom kolmé vektory a v jednom z nich nadobúda  $S(z)$  maximum.

Ako toto všetko využijeme, keď je bodov viac než 4? Zoberme niektorý vrchol  $v_i$ , nech leží na jednej zo strán nejakého opísaného obdĺžnika. Je zrejme, že zvyšné tri strany k nemu vieme ľahko určiť, keď si vyberieme smer tejto strany. Ten nemôže byť ľubovoľný, ale iba z intervalu, ktorý určujú strany k susedným vrcholom  $v_{i-1}$  a  $v_{i+1}$ . V oboch krajných prípadoch sú naše štyri body známe. Interval určený týmito krajnými polohami môže byť ešte rozdelený na podintervaly, kde v každom sú už iba tie isté štyri vrcholy ležiace na stranách opísaného obdĺžnika (intervaly sa dajú nájsť na  $O(n)$  operácií). Teraz môžeme použiť metódu pre štyri body a zmerať, ako by mala byť strana otočená, aby bol opísaný obdĺžnik minimálny, ak vyjde smer mimo aktuálny interval, vďaka tomu, že funkcia je medzi oboma extrémami monotónna, stačí preveriť hraničné smery intervalu a zobrať menší obdĺžnik. Toto musíme urobiť pre všetky intervaly.

**911.** Uvedomte si, že  $2^m - 1 = 2^0 + 2^1 + \dots + 2^{m-1}$ . Teda v prípade  $m = n$  je to ľahké. Ak  $m < n$ , treba niektoré sčítance rozpisovať podľa vzťahu  $2^x = 2^{x-1} + 2^{x-1}$ , pre  $x > 0$ . Zoberte vhodný násobok  $2^m - 1$ . Najelegantnejšie je zobrať  $2^{n-m}(2^m - 1)$ , po roznásobení dostaneme  $2^{n-m} + 2^{n-m+1} + \dots + 2^{n-1}$ . Keď rozpíšeme prvý sčítanec  $2^{n-m}$ , dostaneme práve  $n$  sčítancov  $2^0 + 2^1 + \dots + 2^{n-m-1} + 2^0 + 2^{n-m+1} + \dots + 2^{n-1}$ .

**912.** a) Obaľovanie ostrovov. Každý rok obalíme všetky ostrovy vrstvou políčok, ktoré susedia s morom. Jednotlivé vrstvy očísľujeme podľa roku, kedy sme ich vytvorili. Keď sa vo vrstve, ktorú práve pripisujeme, nájde políčko susediace s políčkom inej vrstvy, tak ak sa čísla vrstiev líšia o jedna, našli sme najkratší most. Ale pozor, ak sú čísla vrstiev rovnaké, musíme pripísať všetky vrstvy v tom roku, či sa nenájde kratší most. Most vyznačíme tak, že sa stačí pohybovať vždy na políčko vrstvy z predchádzajúceho roku, až pokým nena-razíme na pôvodný ostrov. Aby bolo toto riešenie šikovné, musíte pri obaľovaní prezerateľ len políčka pobrežia, nie celú mapu. Políčka pobrežia si ukladajte do frontu. b) Najkratší most môže mať najviac jednu pravouhlú zákrutu. Viac lomený najkratší most môžeme vždy upraviť do takéhoto tvaru. Ak by sa to nedalo, mohla nám prekážať iba pevnina, ale to je spor s tým, že most je najkratší. To sa dá využiť tak, že pre každé políčko mora spočítame jeho vzdialenosť od pevniny na severe, východe a západe. Z toho už ľahko určíme najkratší most. Skúste si to naprogramovať len s použitím pomocného poľa veľkosti  $[1.. \min\{m, n\}]$ . Pozor, treba rozlišovať ostrovy!

**913.** Plocha plechu  $P$  sa rovná  $\sum_{i=1}^k x_i y_i$ . Pozor, nestačí určiť  $m$  a  $n$  také, že  $P = mn$ , ale treba aj overiť, či sa na takomto plechu koláčiky mohli piecť. Kontrola rozloženia koláčikov na plechu sa dá vykonať postupným prezretím všetkých prípustných rozložení (backtracking). Pri rozkladaní si stačí pamätať iba obrys už rozložených koláčikov.

**914.** Pozri 814.

**915.** Robiť úplnú syntaktickú analýzu je zbytočné, pretože program na vstupe je syntakticky správny. Lepšie je realizovať iba príkazy *NEXT*, *PUT*, *GET* procedúrami a premenné *TOP* a *INP* funkciami. Ak sa uspokojíme s Turbo Pascalom, môžeme ich umiestniť do unitu. Všeobecnejšie riešenie ich vsunie medzi hlavičku programu a prvý **begin**. Fronta sa dá realizovať pomocou pevnej dĺžky alebo spájaným zoznamom. Zamyslite sa nad výhodami a nevýhodami takýchto implementácií.

**921.** a) Prvky poľa usporiadame nerastúco, zložitosť závisí od použitého algoritmu triedenia. b) Analogicky ako keď vytvárame haldu (špeciálny prípad, keď  $k = 2$ ), takéto riešenie

vyžaduje  $O(n)$  operácií.

**922.** a) Prejdeme postupne celú mapu a kontrolujeme každé políčko pevniny, či nesusedí s morom. b) Šikovnejšie je obísť ostrov po mori alebo po pobreží. Pozor na ostrov, ktorý má na mape políčka pevniny susediace navzájom iba rohom. Je dobré okolo mapy použiť zarážku pozostávajúcu z políčok predstavujúcich more, aby sme nemuseli testovať, či sme nevyšli z mapy. Ostrov môže byť až po hranice mapy.

**923.** a) Nešikovné riešenie je použiť pole a „zakresliť“ si doň cestu Banta a potom spočítať políčka vo vnútri. Nevieme totiž vopred veľkosť parcely. b) Iné riešenie môžeme dostať, keď sa budeme snažiť doplniť Bantovu cestu pridávaním obdĺžnikov na obdĺžnik. Plochy obdĺžnikov sa dajú ľahko zrátať. Úpravy sa dajú robiť tak, že budeme hľadať v zápise Santových povelov isté „vzory“ a tieto redukovať. Napríklad postupnosti 12, 21 a 30 môžeme smelo vypustiť. Vzor 13x23y1 môžeme nahradiť s 3y13x, kde  $x$  a  $y$  sú počty krokov. Nakreslite si to a skúste porozmýšľať nad ďalšími redukciami. c) Pozorný riešiteľ si isto všimol, že ak si celú plochu rozdelíme na pásy, pričom všetky body, kde Banto menil smer cesty, budú na hraniciach pásov, tak si vlastne stačí iba všímať, akú má Banto  $y$ -ovú súradnicu a či ide vľavo alebo vpravo.  $y$ -ová súradnica je rovnobežná s pásmi. Teraz využijeme to, že Santo sa vrátil odkiaľ vyšiel a neprekrížil svoju cestu. (Všimnite si, ako vyzerá hociktorý pás. Najnižšia a najvyššia cesta sú opačne orientované, rovnako ako všetky susedné cesty v páse.) Keď Santo ide vľavo  $x$  krokov, od plochy odpočítame  $xy$  a keď ide vpravo, zasa pripočítame  $xy$ ,  $y$  je  $y$ -ová súradnica. Ak Santo išiel opačným smerom, ako sme predpokladali, tak nám vyšla plocha záporná, stačí urobiť absolútnu hodnotu (nakreslite si to).

**924.** Pozri 813.

**925.** a)  $O(m^2n^2(m+n))$ ; priamočiare riešenie, skontrolujeme všetky obdĺžniky tak, že pre všetky prípustné pravé horné rohy vyberieme všetky ľavé dolné rohy a preveríme, či ich obvod je nakreslený. b)  $O(m^2n^2)$ ; zlepšenie dosiahneme za cenu pomocnej pamäti, v ktorej si budeme pamätať pre každé pole hracej dosky, aká súvislá čiara je z neho nakreslená smerom hore a vľavo, čo vieme zistiť pri načítaní, tak ušetríme kontrolu obvodu obdĺžnikov. c)  $O(nm^2)$ ; riešenie od Maťa Ondrušku, v každom riadku spočítame, koľko obdĺžnikov má na ňom svoju spodnú stranu. Budeme postupovať zvrchu. Postupne pre každý riadok vyplníme pole  $c$ , rozmerov  $m \times m$ , kde  $c[i,j]$  bude počet tých obdĺžnikov, ktorých spodná hrana je presne od  $i$ -teho po  $j$ -ty stĺpec v danom riadku. V ďalšom riadku určíme  $c[i,j]$  na základe hodnoty v predchádzajúcom riadku s tým, že tu bude nula, ak nebola nakreslená čiara v  $i$ -tom alebo  $j$ -tom stĺpci, inak prenesieme hodnotu s predchádzajúceho riadku. Počet obdĺžnikov je súčet hodnôt poľa  $c$  v každom riadku.

**931.** Riešenie je náročné, pozri 39

**932.** Skúste viac metód a vyberte najvhodnejšiu pre daný vstup. Napríklad použite bity na uloženie hodnoty 1 a 0; zapamätajte si len pozície 1; uložte iba počty po sebe idúcich 1 a 0 (vyberte najvhodnejší smer); uložte si iba počty po sebe idúcich 0 a 1 „skopírujte“.

**933.** Upravte do želaného tvaru ľubovoľné dva susedné riadky. Indiáni zajatca umučili iba v prípade, ak ho chytil jediný indián a zapísal riadok 1 3 2, prípadne jeho nejakú rotáciu.

**934.** Backtracking.

**935.** Riešenie časti a) Označme  $KO$  ako  $P_1$  a  $P_2$ , vyriešime všeobecnejšiu úlohu, nebudeme predpokladať, že  $P_1$  a  $P_2$  sú disjunktné.

1. Nájdeme bod  $p$ , ktorý je vnútorným bodom  $P_1$ ,
2. zistíme, či  $p$  je vnútorný v  $P_2$ , ak nie, pokračuj bodom 4,
3.  $p$  je vnútorný v  $P_2$ . Vrcholy  $P_1$  aj  $P_2$  zlúčime do utriedenej postupnosti podľa uhla od bodu  $p$  a Grahamovým algoritmom (413) určíme  $KO$ ,
4.  $p$  nie je vnútorný v  $P_2$ .  $P_1$  celý leží v uhle  $\phi$  s vrcholom v  $p$ . Uhol  $\phi$  je určený bodmi  $u$  a  $v$  z  $P_2$ . Vrcholy, ktoré ležia v uhle  $\phi$  sú v  $P_1$ , vypustíme. Oba zoznamy zlúčime do



utriedenej postupnosti podľa uhla od bodu  $p$ ,

5. použijeme Grahamov algoritmus (413).

Iné algoritmy na určenie KO sú v riešení úlohy 413.

**941.** Aby sa tanec podaril musí byť tanečníkov párny počet a viac než 7. Vtedy stačí vedieť ubrať s niektorého kraja dvoch až pokým nám neostane osem tanečníkov a pre tých je postup v zadaní.

**942.** pozri úlohu 831.

**943.** Metóda Divide et impera: Utriedme si brlôžky napríklad podľa  $x$ -ových súradníc. Rozdelíme ich na ľavú a pravú polovicu. V každej polovici zistíme najmenšiu vzdialenosť dvojice brlôžkov. Minimum z týchto dvoch čísel nech je  $m$ . Potom buď  $m$  je vzdialenosť dvoch najbližších brlôžkov, alebo je najbližšia dvojica rozdelená, jeden brlôžok je v ľavej, druhý v pravej polovici. Majme v obidvoch poloviciach utriedené brlôžky podľa  $y$ -ovej súradnice. Teraz nám stačí pre každý brloh z ľavej polovice, ktorý je od deliacej čiary vzdialený menej ako  $m$  nájsť najbližší brloh z pravej polovice. Stačí sa zaujímať iba o brlohy, ktoré sú bližšie ako  $m$ , a tých bude pre každý ľavý brloh max. 4. Preto to vieme spraviť v lineárnom čase. Ešte k triedeniu podľa  $y$ : hlavná rekurzívna funkcia, počítajúca z miním oboch polovic minimum celku, môže mať podobný vedľajší efekt ako triedenie zlučovaním: spojí ľavú a pravú polovicu, ktoré už boli utriedené podľa  $y$  do jedného utriedeného poľa. Tak budeme mať tento úsek poľa utriedený pre ďalšie spájanie.

**944.** Stačí testovať len vzdialenosti v každom ľavom ostrom rohu (takom, ktorý vyčnieva do chodby) od protiľahlej zvislej a protiľahlej vodorovnej steny a od všetkých ostrých rohov pravej steny, ležiacich medzi týmito stenami. Ako ostré rohy treba uvažovať aj koniec a začiatok steny.

**945.** Skúsme na to ísť „dynamicky“. Vždy, keď načítame jeden riadok zvislých čiar, aktualizujeme pole  $H$ . V poli  $H$  nech je pre každý bod dĺžka zvislej čiary, vedúcej hore od tohto bodu. Body štvorky, v ktorých sa vodorovná palička napája na zvislé, nazvime koleno (ľavý) a uzol (pravý). Počet všetkých štvoriek, ktoré majú spoločné koleno a uzol je zrejme  $H_{k,y} \times H_{u,y} \times d$ , kde  $H_{k,y}$  a  $H_{u,y}$  označujú dĺžky paličiek nad daným kolonom a uzlom a  $d$  je dĺžka paličky pod uzlom. (prirodzene, za predpokladu, že koleno a uzol sú spojené vodorovnou čiarou) Číslo uzla nech je súčet  $\sum_k H_{k,y} \times H_{u,y}$  cez všetky kolená príslušiace danému uzlu. Ten vieme spočítať lineárne pre všetky uzly v riadku, za predpokladu, že máme správne spočítané pole  $H$ . Správime to pri načítavaní riadku vodorovných čiar. V poli  $S$  nech je pre každý bod súčet čísel uzlov na čiare nad týmto bodom. Stačí si pamätať jeden riadok. Tieto čísla znamenajú počet štvoriek, ktorých nožička končí práve v danom bode. Stačí teda vypočítať  $\sum_{x=1}^n \sum_y^m S_{x,y}$  a je to. V praxi nasledujúci riadok polí  $H$  a  $S$  závisí iba od predchádzajúceho a od príslušného riadku vstupu, preto si stačí z obidvoch polí pamätať iba po jednom riadku.

**1011.** Existuje lepšie riešenie ako backtracking! Stačí si uvedomiť, že ak si budeme pamätať najväčší možný objem, ktorý možno vypíť z radu dlhého  $k$  a končiaceho jednou zo štyroch možností vypitia, prípadne nevypitia posledných dvoch fliaš. Musíme si vypočítať takúto tabuľku pre  $2 \leq k \leq n$ . Maximálny objem vypitých fliaš bude v poslednom riadku. Teraz ešte stačí od konca zistiť, ktoré fľaše treba vypíť. Ideme po tých, kde sa dosahuje maximum pre  $n, n-1, \dots, 1$ .

**1012.** Nech je hodnota políčka  $A_{i,j}$  rovná veľkosti najväčšieho chrámu, ktorý by tu mal svoj najsevernejší bod. Vtedy už úlohu ľahko vyriešime. Ako určiť hodnotu  $A_{i,j}$ ? Veľkosť chrámu s najsevernejším bodom  $(i,j)$  priamo závisí od veľkostí najväčších chrámov, ktoré majú najsevernejšie body v políčkach  $(i-1, j-1)$ ,  $(i, j-1)$ ,  $(i+1, j-1)$ . Teda  $A_{i,j} = \min\{A_{i-1,j-1}, A_{i,j-1}, A_{i+1,j-1}\} + 1$ .

**1013.** a)  $O(n)$ ; postupujeme ako pri zlučovaní utriedených postupností a počítame si, koľkých sme už zlúčili b)  $O(\log n)$ ; idea binárneho vyhľadávania. V každej postupnosti si

pamätáme rovnako dlhé úseky, v ktorých sa môže nachádzať hľadaný žiak. Vždy porovnáme „stredných“ z oboch úsekov, ak sú rovnako vysokí, skončili sme. Ak je napríklad z prvého úseku menší, ďalej berieme prvú polovicu druhého úseku a druhú polovicu prvého úseku.

**1014.** Tento príklad je veľmi komplikovaný, obsahuje vo svojom riešení mnoho podúloh, ktoré by mohli byť samostatnými príkladmi. Každú ulicu treba prejsť. Ak by sa nám podarilo každú prejsť iba raz, máme riešenie. To vieme, ak sa vo všetkých križovatkách zbíha párny počet ciest. Ak sú v meste aj nepárne križovatky, pokúsime sa medzi nimi doplniť pomocné cesty a dostaneme úlohu 643. Cesty musíme doplniť tak, aby sme celkovú dĺžku ulíc mesta zväčšili minimálne. Na to ale potrebujeme vedieť dĺžky najkratších ciest medzi nepárnymi križovatkami – úloha 523. Doplnenie ciest je v skutočnosti zdvojenie ulíc nachádzajúcich sa na týchto najkratších cestách. Uvedomte si, že potrebujeme vybrať také cesty, ktoré budú mať v súčte minimálnu dĺžku, takže žiadne dve nepôjdu po rovnakej ulici, lebo by sme ju mohli vypustiť a mali by sme kratšiu cestu. Všetky nepárne križovatky musíme pospájať, riešenie je napríklad v 8.D v [34]. Riešenie celej úlohy je v 9.B v [34] a úloha sa tu nazýva úloha čínskeho poštára.

**1015.** Riešenie založené na backtrackingu považujeme za veľmi nešikovné. Rozumné je riešenie vyžadujúce  $O(n^2)$  operácií. Predpokladajme, že poznáme dĺžky najkratších okružných ciest pre všetky  $j$ ,  $2 \leq j \leq k$  a pokúsme sa nájsť dĺžku najkratšej okružnej cesty pre  $k+1$  miest. Napíšme si postupnosť miest v tejto ceste:  $1, \dots, i, k+1, k, k-1, \dots, i+1, \dots, 1$ . Všimnite si, že pre nejaké  $i$  táto cesta obsahuje ako svoju podcestu skoro celú najkratšiu okružnú cestu cez  $i$  miest. Tiež si uvedomte, v akom poradí sme vypísali mestá. Návod, ako nájsť dĺžku najkratšej okružnej cesty, je priamočiarý:  $D_{k+1} = \min_{2 \leq i+1 \leq k} \{D_{i+1} - |i, i+1| + |i, k+1| + \sum_{j=i+1}^k |j, j+1|\}$ ,  $D_1 = 0$ ,  $D_2 = 2|1, 2|$ . Rozmyslite si, ako sa dá určiť, ktoré mestá ležia na ceste tam a ktoré na ceste späť a v akom poradí. Asi na to budete potrebovať informáciu, pre ktoré  $i+1$  sa dosiahlo minimum pre každé  $1 < k+1 \leq n$ .

**1021.** a)  $O(n^2)$ ; odpovede si zapisujeme do matice  $A$  rozmerov  $n \times n$ ,  $A_{i,j} = 1$ , ak  $i$  pozná  $j$ . Hľadáme teda riadok  $k$  obsahujúci jednotku na pozícii  $A_{k,k}$  a  $k$ -ty stĺpec musí obsahovať iba samé 1. b)  $O(n)$ ; všimnite si, že môžeme využiť obe z možných odpovedí. Pri každej vieme vylúčiť jedného kandidáta na dôležitú osobu. Teda po  $n-1$  otázkach máme jediného kandidáta. Stačí skontrolovať, či nikoho nepozná a či jeho všetci poznajú. Ak to urobíte šikovne, budete musieť položiť ešte  $n+k-3 + [k=1]$  otázok, kde  $k$  je číslo kandidáta.

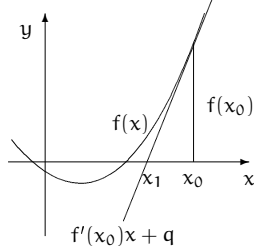
**1022.** Zistenie priemeru binárneho stromu (*priemer grafu* je dĺžka najdlhšej z najkratších ciest medzi nejakými dvoma vrcholmi). Úloha sa dá riešiť tak, že do každého bodu popísanej cesty pridáme iba raz. Nestačí rátať iba hĺbky jednotlivých fjordov, ale treba si všímať aj ich priemer. Aby sme určili priemer fjordu zloženého z dvoch podfjordov, zoberieme maximum zo súčtu hĺbok oboch podfjordov a priemerov oboch podfjordov. Hĺbka fjordu je maximum z hĺbok jeho podfjordov zväčšené o dĺžku fjordu.

**1023.** Jednou z možností je použiť princíp zapojenia a vypojenia. Spočítame plochy všetkých kruhov, odpočítame od nich plochy prienikov všetkých dvojíc, pripočítame prieniky trojíc atď. Nevýhoda je, že ak sú skoro všetky kruhy na kope, takýto algoritmus môže byť až exponenciálny. Lepšia možnosť je spočítať všetky priesečníky kružníc, každým viesť jednu severojižnú priamku. Tieto priamky nám rozdelia plochu na niekoľko pásov, v ktorých sa kružnice nepretínajú. V každom páse ostane niekoľko kruhových odsekov (pozor, niektoré ležia vnútri iných), ktorých plochu nie je problém zrátať.

**1024.** Predpokladajme spojitosť ihriska. Z ľubovoľného miesta vyšleme po hranách vlnu, ktorá sa šíri po každej hrane najviac jedenkrát (prehľadávanie do šírky). Keď do dvoch miest  $X, Y$  príde vlna z dvoch rôznych smerov, tak osmička na ihrisku existuje (môže sa stať, že  $X=Y$ , ale smery sú iné). Algoritmus je lineárny od počtu ciest.

**1025.** Kľúčovým miestom riešenia je výber vhodnej reprezentácie výsledku činnosti Santa a Banta (graf). Treba zistiť, či je graf súvislý. Dá sa v čase lineárnom od počtu hrán.

**1031.** Po každom presadnutí zmenšíme počet nesprávne sediacych. Teda na voľnú stoličku si presadne vždy niekto, kto nesedí na dobrom mieste. Skúste si dokázať, že takýto algoritmus je správny. Viete to spraviť tak, že každého čakajúceho skontrolujete iba jeden raz?



**1032.** a) Koľko je vojakov v prvých s štvorcach?  $S(s) = \sum_{i=1}^s i^2$ . Takže vojak s číslom id patrí do štvorca veľkosti  $\min_s \{S(s) - id \geq 0\}$ . V programe použite len sčítanie a porovnanie! b) (Newtonova<sup>31</sup> iteračná metóda) [2. kap. v 36] Označme  $f(x) = S(x) - id$ . Keď nájdeme riešenie,  $\hat{x} > 0$ , rovnice  $f(x) = 0$ ,  $\lceil \hat{x} \rceil$  je strana hľadaného štvorca. Prvé približenie  $\hat{x}$  položíme  $x_0 = id$ , nasledujúce počítame podľa vzťahu  $x_{i+1} = x_i - f(x)/f'(x)$ , čo nás vždy vďaka vhodnej voľbe  $x_0$  a konvexnosti  $f(x)$  dovedie k riešeniu, pravda len približnému. Pozrite si obrázok.

**1033.** a) Slová utriedime, použiť treba algoritmus, ktorý vykoná približne  $n \log n$  operácií. Spomedzi utriedených slov už ľahko nájdeme najčastejšie sa vyskytujúce slovo. b) Hašovanie [41, 42, 23]. Každému slovu nájdeme pomocou hašovacej funkcie jednoznačne miesto v tabuľke slov,  $T[1..S]$ . Hašovacia funkcia môže mať napríklad tvar  $C \bmod k$ , kde  $C$  je 4 bytové číslo zostavené z kódov prvých štyroch znakov slova a  $k$  je číslo nesúdeliteľné s  $S$ . Rovnaké slová budú v tabuľke na tom istom mieste, takže si stačí pamätať, koľko ich je. Môže sa stať, že na rovnaké miesto sa zobrazí viac rôznych slov, tieto je najjednoduchšie zrefaziť do zoznamu a príslušnému slovu zvýšiť počet výskytov. V priemernom prípade treba na nájdanie slova len konštantne veľa pokusov (max. 2), dokonca i keď je tabuľka plná [6.4 v 23]. c) Binárny vyhľadávací strom, [4.4.3 v 42]. V každom vrchole je uložené slovo a počet jeho výskytov. V ľavom podstrome sú všetky slová menšie a v pravom zasa väčšie než je slovo uložené v koreni.

**1034.** a) Hľadáme zhodné skupiny, ktorých začiatky sú od seba vzdialené o  $k$  znakov (môžu sa aj prekrývať),  $0 < k < \text{Dĺzka} - \text{Maxd}$ , kde  $\text{Maxd}$  je dĺžka doteraz najdlhšieho opakujúceho sa úseku. Zisťujeme, či  $T[i] = T[i + k]$ ,  $0 < i < \text{Dĺzka} - k$ . Uvedomte si, že ak sú začiatky najdlhších skupín vzdialené o  $k$ , tak ich vždy nájdeme. Očividne na to potrebujeme približne  $\text{Dĺzka}^2$  operácií. b) Ak sa môžu opakujúce sa úseky prekrývať, existuje riešenie využívajúce pozičné stromy, ktoré vyžaduje len  $O(\text{Dĺzka})$  operácií [9.5 v 1].

**1035.** Máme vlastne nájsť kružnicu s minimálnym polomerom, vnútri ktorej budú ležať všetky vrcholy  $n$ -uholníka. Jej stred bude zrejme hľadaný bod. Vyskúšame metódu fuciaho balónika. Nájdeme nejakú kružnicu, vnútri ktorej sa nachádzajú všetky vrcholy (napríklad opíšeme  $n$ -uholníku obdĺžnik so stranami rovnobežnými so súradnicovými osami a jemu opíšeme kružnicu) a postupne ju budeme zmenšovať, tak, aby vždy boli všetky vrcholy vnútri. Necháme jej stred  $S_1$  na mieste a polomer zmenšíme tak, aby prechádzala bodom  $X$  najvzdialenejším od stredu. Ďalej ju trochu vyfučíme tak, aby stále prechádzala  $X$ , jej nový stred  $S_2$  bol na spojnici  $S_1$  a  $X$  a aby prechádzala ďalším vrcholom  $Y$ . Máme všetko pripravené, môžeme použiť vyfučovaciu procedúru: Nájdeme stred novej kružnice, ktorý leží na osi  $\overline{XY}$  a má tú vlastnosť, že všetky body ležia vnútri kružnice a jej polomer je minimálny.  $X$  a  $Y$  iste budú ležať na novej kružnici. Buď budú ležať na priemere, alebo na nej bude ležať ešte jeden bod  $Z$ . Ak  $X$  a  $Y$  tvoria priemer, skončili sme, lebo zrejme menšiu kružnicu nenájdeme. Ak  $\triangle XYZ$  je ostrouhlý, tiež sme zrejme našli najmenšiu možnú kružnicu. Ak je tupouhlý, iste existuje menšia, prechádzajúca tými dvoma vrcholmi, pri

<sup>31</sup> Isaac Newton, 1643-1727, anglický matematik, fyzik a astronóm.

ktorých nie je tupý uhol. Na tieto dva vrcholy opäť našijeme vyučovaciu procedúru. Je zrejmé, že ten „tupý“ vrchol už nikdy nebude na obvodě našej kružnice, preto ho môžeme úplne vyhodíť zo zoznamu vrcholov. Tým je dokázané, že algoritmus po najviac  $n - 2$  vyhodeniach vrcholu skončí, a preto je kvadratický.

**1041.** a) Bez delenia. Využijeme, že kombinačné číslo je celé. Každý činiteľ v čitateli aj menovateli rozložíme na prvočinitele, potom stačí len pripočítavať alebo odpočítavať exponenty k príslušným exponentom prvočísel vo výsledku. Nakoniec prvočísla vynásobíme, na to stačí vedieť násobiť malým číslom. Uvedomte si, že netreba každý činiteľ rozkladať na prvočinitele osobitne. Urobte si pomocné pole, v ktorom budete mať pre každý činiteľ informáciu, akým najmenším prvočíslom je deliteľný a aký je výsledok tohoto delenia. b) S delením, podľa [4.3.1 z 22].

**1042.** Najväčší ostrov, ktorý sa potopí za  $r$  rokov označme  $O_r$ . Je to štvorec, ktorého uhlopriečka v štvorcovej metrike má dĺžku  $2r+1$ . Úloha sa redukuje na určenie, aký najväčší  $O_r$  jednotlivé ostrovy obsahujú a veľkosť najmenšieho z nich je čas potopenia sa 1. ostrova. Ako určiť  $O_r$ ? Podobne ako v 1012, ak pre každé políčko určíme  $r$ , t.j. aký najväčší  $O_r$  nad týmto políčkom môže byť. Ak určíme hodnotu pre  $(i, j)$ , tak je to najmenšia hodnota z políčok  $(i-1, j-1)$ ,  $(i, j-1)$ ,  $(i+1, j-1)$ ,  $(i, j-2)$  zväčšená o jedna (nakreslite si to!). Všimnite si, že na to, aby sme určili hodnotu všetkých políčok mapy, stačí mapu raz prejsť po riadkoch zhora nadol.

**1043.** Úloha je viacnásobným hľadaním konvexného obalu (KO). Na určenie KO možno použiť napríklad Jarvisov algoritmus, ktorý je popísaný v riešení c) úlohy 413. Na určenie KO  $n$  bodov potrebujeme  $O(kn)$  operácií, kde  $k$  je počet vrcholov KO. V tomto prípade je každý spomedzi  $n$  bodov najviac v jednom KO, teda celková zložitosť algoritmu bude  $O(n^2)$ .

**1044.** Na riešenie možno použiť jednoduchú modifikáciu Floyd-Warshallovho algoritmu (pozri [34]), keď sčítanie nahradíme násobením.

**1045.** Ku každému človeku hľadáme rekurzívne všetkých, ktorých v konečnom dôsledku živí a pamätáme si to v poli. Keď niekto živí človeka, ktorý je už preskúmaný, využijeme zistené údaje. Pozor na cykly v grafe.

**z1011.** Priamo zo zadania vyplýva, že Čiapočky si môžu čiapočky vymeniť práve vtedy, ak  $C[i] = i$ ,  $1 \leq i \leq n$ .

**z1012.** Označme  $S_{i,j}$  súčet  $C_i + C_{i+1} + \dots + C_j$ , pre  $i \leq j$ . V prípade  $S_{i,j} < k$ , môžeme určiť  $S_{i,j+1} = S_{i,j} + C_{j+1}$ . Ak  $S_{i,j} > k$ , stačí určiť  $S_{i+1,j} = S_{i,j} - C_i$ . Ak nájdeme také  $i, j$ , že  $S_{i,j} = k$ , Mamojkov strach bol opodstatnený, inak nie.

**z1013.** Na poradií prvkov nezáleží, sú to maskované kombinácie  $k$  prvkov z  $n$ , takže Pišta môže batoh naplniť  $\binom{n}{k}$  spôsobmi. a) Do batohu môžeme dať buď  $k$ -tice, ale len z  $n-1$  tovarov (jeden tovar vôbec nepoužijeme), alebo  $k-1$ -tice z  $n-1$  tovarov a tiež  $n$ -tý tovar (jeden tovar vždy bude v batohu). Na základe tejto myšlienky už ľahko napíšeme rekurzívny program. b) Ak si tovary označíme číslami od 1 po  $n$ , rôzne náplne batohov môžeme lexikograficky usporiadať. Generovať takéto  $k$ -tice sa dá ľahko nerekurzívnym programom [1.7 v 27].

**z1014.** a) Najjednoduchšie je nechať zohrať 1. zápas borcov č.1 a č.2 a víťaza 1. zápasu zohrať 2. zápas s borcom č.3, atď., víťaza  $(n-2)$ -ého zápasu zohrať  $(n-1)$ -vý zápas s borcom č. $n$ . Analogicky určíme najslabšieho. Celkovo potrebujeme  $2n-2$  zápasov. b) Lepší spôsob je nechať zohrať zápasy najprv  $n$  div 2 rôznych dvojíc a potom spomedzi víťazov rovnakým spôsobom ako v a) vybrať najsilnejšieho a spomedzi porazených zasa najslabšieho. Ak je  $n$  párne, potrebujeme  $n/2 + (n/2 - 1) + (n/2 - 1) = (3/2)n - 2$  zápasov, ak je  $n$  nepárne,  $(n-1)/2 + ((n-1)/2 - 1) + ((n-1)/2 - 1) + 2 = (3/2)n - 3/2$  zápasov, alebo pre oba prípady  $\lceil (3/2)n - 2 \rceil$ .

**z1021.** Stačí si vybrať ešte neskontrolovanú Čiapočku a postupne prejsť cez všetky, ktoré sa takto držia. Všetky označíme ako skontrolované. Uvedomte si, že nakoniec musíme prísť opäť k Čiapočke, s ktorou sme začali (prečo?). Ak sme skontrolovali už všetky, sme hotoví, ak nie, nájdeme prvú neskontrolovanú a začneme odtiaľ.

**z1022.** Nešikovné je počítať vzdialenosti pre každé dve mestá osobitne. Spočítajte si vzdialenosti  $s_i$  z 1. mesta do mesta  $i$ . Vzdialenosť miest  $j$  a  $k$  je potom  $s_k - s_j$ , pre  $k > j$ .

**z1023.** Treba si uvedomiť, že stačí brať maškrtky usporiadané podľa jednotkovej ceny ( $c/v$ ) od najväčšej. Akonáhle už nemôžeme zobrať celú maškrtu, zoberieme takú jej časť, aby sme zapratali celú tašku. Takto možno dosiahnuť časovú zložitosť rovnú zložitosti triedenia (t.j. prinajlepšom  $O(n \log n)$ ). Zlepšenie na  $O(n)$  možno realizovať tak, že maškrtky rozdelíme na dve počtom podobné časti tak, aby každá maškrtka v jednej časti mala jednotkovú cenu vyššiu ako každá maškrtka z druhej časti (tzv. pivotizácia, pozri príklad z1042). Potom sa pozrieme, či možno zobrať všetky drahé maškrtky. Ak áno, tak ich vezmeme a zostanú nám iba lacné, inak sa zaoberáme iba drahými. Daný postup opakujeme, až kým nezostane jedna maškrtka a tú podľa potreby rozkrojíme.

**z1024.** Vždy treba mať akcie len jednej spoločnosti. Výhodnejšia je tá, ktorej pomer cien  $q_{i+1}/q_i$  je väčší (skúste dokázať),  $q_i$  je cena v  $i$ -ty deň. Majetok po  $n$  dňoch vzrastie najviac  $\prod_{i=1}^{n-1} \max\{\frac{w_{i+1}}{w_i}, \frac{t_{i+1}}{t_i}\}$  násobne. Pozor, vybrať lepšiu spoločnosť na základe  $q_{i+1} - q_i$  je zlé riešenie. Protipríkladom sú napríklad hodnoty  $t = 1, 2, 3$  a  $w = 1, 0.1, 1$ .

**z1031.** Ak vieme úlohu vyriešiť pre  $n-1$  a riešením sú vektory  $v_1, v_2, \dots, v_{2^{n-1}}$ , riešenie pre  $n$  dostaneme takto: Všetky vektory napíšeme dvakrát pod seba, raz v poradí 1. až  $2^{n-1}$ , a potom v zrkadlovom poradí  $2^{n-1}$ . až 1. K prvej polovici pripíšeme na koniec 0 a k druhej 1. Dostaneme teda  $v_1 0, v_2 0, \dots, v_{2^{n-1}} 0, v_{2^{n-1}} 1, \dots, v_2 1, v_1 1$ . Naprogramovať ho môžeme a) rekursívnym programom, v ktorom použijeme iba jedno pole `array[1..n]` of  $0..1$ ; b) keď si pozornejšie všimneme  $n$ -tice riešenia, pridáme na to, že vieme určiť, aká cifra je v  $r$ -tom riadku a  $s$ -tom stĺpci riešenia.  $s$ -tý stĺpec začína  $2^s$  nulami a potom sa striedajú skupiny jednotiek a núl dĺžky  $2^{s+1}$ . Keď riadky  $r$  číslujeme od 1 po  $2^n$  a stĺpce  $s$  od 1 po  $n$  (zprava doľava), v riadku  $r$  a stĺpci  $s$  je 1 práve vtedy, keď  $\lfloor (\frac{r-1-2^{s-1}}{2^s}) \rfloor \bmod 2 = 0$ .

**z1032.** Treba preskúšať všetky možné obsahy batohu. Najšikovnejšie je usporiadať tovary podľa objemu. Keď volíme jednotlivé naplnenia vieme, či má zmysel ešte do batohu pridávať. Takýmto spôsobom podstatne zmenšíme počet zbytočne kontrolovaných naplnení batohu. Pozor! Riešenie, v ktorom sa tovary do batohu vyberajú na základe maximálnosti ceny alebo podielu cena/objem alebo minimálnosti objemu, prípadne kombináciou predchádzajúcich spôsobov, nie je dobré. Nájdite protipríklady na jednotlivé spôsoby.

**z1033.** V skupine, ktorá kandiduje na najdlhšiu, musia byť najprv znaky A, potom znaky B a na koniec znaky C a pre ich počet musí platiť  $\text{počet}(A) \geq \text{počet}(B) \leq \text{počet}(C)$ . Teda rozhodujúci je počet znakov B v skupine.

**z1034.** Ak si označíme  $f[s, n]$  farbu Čiapočky v skupine  $s$ , ktorú drží Čiapočka číslo  $n$  ľavou rukou, po prechytení bude držať Čiapočku farby  $f[3-s, f[s, n]]$ . Všimnite si, že  $3-s$  je číslo druhej skupiny. Teraz stačí „ísť“ postupne po Čiapočkách, kým neprídeme k Čiapočke, od ktorej sme začali. Ak sme už prešli cez všetkých  $n$  Čiapočiek, sme hotoví, inak treba nájsť Čiapočku, cez ktorú sme ešte nešli. Takto postupne prejdeme cez všetky kruhy, ktoré po prechytení vznikli. a)  $n = 3$ , rozmyslite si prečo! b) Pre ľubovoľné  $n$  stačí, aby sa aspoň v jednej skupine točili všetky Čiapočky na mieste.

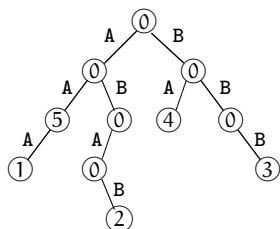
**z1041.** Táto úloha patrí medzi klasické programátorské úlohy. V riešení treba prekonať iba jednu prekážku: ako vytlačiť pomocou štandardných funkcií alebo procedúr tie riadky, ktoré obsahujú príkazy na vytlačenie riadkov programu (trochu krkolomné, ale tento cyklus

treba v riešení rozseknúť). Uvádzame jedno z možných riešení v Turbo Pascale:

```
var a : array[1..7] of string; i : integer;
begin
  a[1] := 'var a : array[1..7] of string; i : integer;';
  a[2] := 'begin';
  a[3] := '  writeln(a[1]); writeln a[2]);';
  a[4] := '  for i := 1 to 7 do';
  a[5] := '    writeln(#97:3, #91, i, #93, #58, #61, #39, a[i], #39, #59);';
  a[6] := '  for i := 3 to 7 do writeln(a[i]);';
  a[7] := 'end.';
  writeln(a[1]); writeln(a[2]);
  for i := 1 to 7 do
    writeln(#97:3, #91, i, #93, #58, #61, #39, a[i], #39, #59);
  for i := 3 to 7 do writeln(a[i])
end.
```

**z1042.** Uvedené preusporiadanie sa nazýva pivotizácia<sup>32</sup>. Najšikovnejšie je hľadať prvok  $A[j] \leq p$  a prvok  $A[k] \geq p$  také, že  $j < k$ , a potom ich vymeniť. Ak také prvky už neexistujú, sme hotoví.

**z1043.** a) Nešikovné riešenie je zapamätať si všetky vstupné slová v poli a potom zadané slovo s nimi porovnávať. b) Lepšie je použiť lexikografický strom. V našej úlohe to bude vlastne len binárny strom, ale nič nebráni tomu, aby sme ho rozšírili na viac písmen.



Vstupné slová uložíme do stromu nasledovne. Začneme v koreni, ak je prvé písmeno v slove A, posunieme sa v strome vľavo, ak je prvé písmeno B, posunieme sa vpravo. Prvé písmeno zo slova odstránime a z vrcholu, v ktorom sa nachádzame, pokračujeme so zvyškom vstupného slova. Keď vyčerpáme celé vstupné slovo, do vrcholu zapíšeme jeho poradové číslo. Je zrejmé, ako teraz kontrolujeme, či zadané slovo bolo vo vstupnej postup-

nosti, stačí sa podľa písmen slova posúvať. Keď vyčerpáme celé slovo a hodnota bude 0, alebo chceme ísť neexistujúcou vetvou, slovo sa vo vstupnej postupnosti nenachádzalo, inak je vo vrchole jeho poradové číslo. Uvedomte si, že pri takejto reprezentácii vykonáme počet porovnaní úmerný dĺžke zadaného slova, bez ohľadu na to, koľko slov bolo na vstupe. Na obrázku je príklad stromu pre vstupné dáta: 'AAA', 'ABAB', 'BBB', 'BA', 'AA', ''.

**z1044.** a) Riešenie sústavy lineárnych rovníc. Metódou neurčitých koeficientov spočítame koeficienty polynómu  $P_k(n) = a_{k+1}n^{k+1} + a_k n^k + \dots + a_1 n + a_0$ . Zvolíme si  $k+2$  hodnôt  $n$  a dostaneme sústavu  $k+2$  rovníc s  $k+2$  neznámymi.  $O(k^3)$ . b) Riešenie využívajúce matematiku – binomická veta. Platí, že  $(n+1)^{k+1} - n^{k+1} = (k+1)n^k + \binom{k+1}{2}n^{k-1} + \dots + 1$ . Označme  $S_k = \sum_{i=1}^n i^k$ . Keď sčítame

$$\begin{aligned} 2^{k+1} - 1^{k+1} &= (k+1)1^k + \binom{k+1}{2}1^{k-1} + \dots + 1 \\ 3^{k+1} - 2^{k+1} &= (k+1)2^k + \binom{k+1}{2}2^{k-1} + \dots + 1 \\ &\dots \\ (n+1)^{k+1} - n^{k+1} &= (k+1)n^k + \binom{k+1}{2}n^{k-1} + \dots + 1 \end{aligned}$$

<sup>32</sup> tvorí základ triediaceho algoritmu Quicksort, ktorý vymyslel Charles Antony Richard Hoare, Computer Journal 5, 1962, 10–15

dostaneme  $(n+1)^{k+1} - 1 = (k+1)S_k + \binom{k+1}{2}S_{k-1} + \dots + S_0$ , odkiaľ  $S_k = ((n+1)^{k+1} - 1 - \binom{k+1}{2}S_{k-1} - \dots - S_0)/(k+1)$ . Z poslednej rovnosti už ľahko  $S_k$  vypočítame, stačí si pamätať prvých  $k+1$  riadkov Pascalovho trojuholníka a koeficienty k predošlým vzorcov. Výpočty je vhodné robiť použitím zlomkovej aritmetiky.  $O(k^3)$ . c) Platí  $P_k(n) = \sum_{i=1}^n i^k$ , teda máme  $P_k(n) + (n+1)^k = P_k(n+1)$ , odkiaľ dostaneme  $(n+1)^k = P_k(n+1) - P_k(n)$ . Teraz obe strany rozpíšeme a porovnáme koeficienty pri rovnakých mocninách  $n$ . Dostaneme sústavu  $k+1$  rovníc,  $0 \leq j \leq k$ ,  $\binom{k}{k-j} = \sum_{i=j}^k a_{i+1} \binom{i+1}{i+1-j}$ . Sústava je trojuholníková, takže stačí postupne dosadzovať.  $O(k^2)$ .

**1111.** Veľmi pomôže jednoduché tvrdenie: Súčet arit všetkých operátorov P-výrazu je o 1 menší ako počet tokenov, (t.j. v našom prípade počet znakov). Jednoduchý dôkaz indukciou: pre  $0 \dots 9$  to platí, nech to platí pre  $v_1 \dots v_n$ , potom to platí aj pre  $Ov_1 \dots v_n$ , kde  $O$  je  $n$ -árny operátor. Najprv zistíme súčet arit všetkých operátorov a počet otáznikov. Musí platiť  $n = A + p \times a_o + 1$ , kde  $n$  je počet znakov,  $A$  je súčet arit známych operátorov,  $p$  počet otáznikov a  $a_o$  neznáma arita otáznika. Z toho vypočítame  $a_o$ . Ak nevyjde celočíselné, skončili sme. Inak treba (pri už známej arite ?) skontrolovať korektnosť P-výrazu, čo spravíme rekurzívnou funkciou presne podľa definície P-výrazu.

**1112.** Ako ináč, pomôcť môže jedine rekúzia. Počet vnorení nejakej procedúry bude znamenať veľkosť súčtu. Procedúra *Nacitaj* načíta do globálnej premennej číslo a zavolá procedúru *Minus1*. Tá zníži premennú o 1 a zavolá sama seba. Ak je premenná nulová, zavolá opäť *Nacitaj*. Pri návrate *Minus1* zvýši opäť premennú o 1. *Nacitaj*, keď načíta 0, vynuluje premennú a skončí. Vylepšenia: procedúry pre zapamätanie väčších čísel (*Minus10*, *Minus100*, ...). Nezabudnite na záporné čísla na vstupe.

**1113.** Trochu prestrelené zadanie. Nie je známe nejaké exaktné riešenie. Testovanie trojuholníkových alebo  $n$ -uholníkových nerovností funguje len ako nutná podmienka. Nádej poskytujú iteratívne riešenia. Jednoduché fyzikálne iteratívne riešenie: Rozmiestnime plte náhodne, namiesto palíc medzi plte natiahnime pružiny s danou tuhosťou a dĺžkou rovnou dĺžke palice. Každá plť nech má svoju hmotnosť, odpor prostredia nech je priamo úmerný rýchlosti plte. Simulujeme pohyb plťí. Ak sme vhodne zakomponovali odpor prostredia, po istom čase sa plte prestanú hýbať. Potom stačí skontrolovať, či tento rovnovážny stav vyhovuje zadaniu a či stačí vymeniť pružiny za palice.

**1114.** Usporiadajme si domy od najjužnejšieho po najsevernejší (podľa južnej steny). Východné a západné steny delia  $x$ -ovú os (rovnobežku) na pásiky. V každom tomto pásiku si stačí pamätať výšku doteraz najvyššieho domu. Postupne budeme pridávať domy od najjužnejšieho, t.j. v každom pásiku, do ktorého dom zasahuje, zistíme, či ho vidno, ak áno, zapamätáme si tento dom ako najvyšší v tomto pásiku a vieme, že tento dom určite vidno. Pásikov je úmerne  $n$ , preto zložitosť je  $O(n^2)$ .

**1115.** Najdôležitejšia časť riešenia je vhodné rozvrhnutie štruktúry jednotlivých súborov, procesu hodnotenia a procesu kalibrácie riešení. Potrebné informácie k tomu, ako zastaviť program po ubehnutí časového limitu nájdete v Techhelpe (interrupty 21, 01 a 1C).

**1121.** Najprv treba zistiť, či sa všetkým dá vyhovieť. Dá sa to, ak kamarát každého študenta chce bývať s ním a zároveň nie sú dvaja ľudia, ktorí chcú bývať s tým istým študentom. Sťahovanie. Najjednoduchšie je zobrať študenta a ak nebýva s kamarátom vyhodíť jeho spolubývajúceho a prisťahovať tam jeho kamaráta. Opakovať pre všetkých študentov. Aby sme nemuseli lineárne prezeráť pole *Izba*, je vhodné spraviť si pole, v ktorom bude pre každú izbu zoznam obyvateľov a udržiavať si ho počas výpočtu.

**1122.** dvojková sústava, výpis čísla Jedno z riešení je na tvrdo napísať procedúru na výpis čísla v dvojkovej sústave zhruba v štýle

```
if n and $4000 > 0 then write(1);
if n and $2000 > 0 then write(1) else if n > $2000 then write(0);
```

...alebo využiť silu rekurzcie (viď aj príklad 1112). Základom bude rekurzívna procedúra *Bin*. Ak je  $n$  párne, väčšie ako 1, vydolí ho 2, rekurzívne sa zavolá a vypíše 0. Pri návrate musí obnoviť pôvodný stav, t.j. vynásobiť  $n$  dvoma. Ak je nepárne, odpočíta 1, vydolí 2, rekurzívne sa zavolá, vypíše 1, vynásobiť  $n$  dvoma a pripočíta 1. Triviálne prípady  $n = 0, 1$  iba vypíšu 0 alebo 1. Po skončení *Bin* máme v  $n$  pôvodné číslo, na obrazovke je vypísané, môžeme ho veselo vypísať desiatkovo a skončiť.

**1123.** Backtracking. Ku každej pečiatke si môžeme pamätať zoznam pečiatok, ktoré ju potrebujú a počet pečiatok, ktoré treba na jej získanie. Ďalej máme zoznam pečiatok, ktoré vieme získať hneď. Z tohto zoznamu vyberieme jednu pečiatku, dáme ju do zoznamu riešenia, aktualizujeme zoznam hneď získateľných pečiatok a počty potrebných pečiatok pre každú pečiatku. Rekurzívne opakujeme, kým nezískame žiadanú pečiatku.

**1124.** Majme graf, v ktorom sú ilegáli vrcholmi a hrana medzi  $i$  a  $j$  je práve vtedy, ak  $i$  pozná  $j$ , alebo  $j$  pozná  $i$ . Artikuláciou v grafe nazveme taký vrchol, po ktorého vyňatí sa graf rozpadne na dve časti (to je vlastne naša dôležitá osoba). Úlohou je zistiť, či sa v tomto grafe nachádzajú artikulácie (pozri [3.B v 34]).

Použijeme jednoduché prehľadávanie do hĺbky, pričom si pri každom vrchole budeme pamätať dve čísla:  $p_i$  značí poradie vrcholu  $i$ , v ktorom sme ho prehľadávaním v grafe objavili a najmenšie  $r_i$  také, že sa z  $i$  do vrcholu s poradím  $r_i$  môžeme dostať cestou pozostávajúcou z niekoľkých hrán prechádzaných v smere prehľadávania a nasledovaných jednou spätnou hranou (to je taká, ktorou sme pri prehľadávaní neprešli do nového vrcholu, lebo sme tam už boli). Ak po skončení algoritmu pre niektorý vrchol platí, že existuje aspoň jeden jeho syn  $j$  taký, že  $r_i \geq p_j$ , potom je vrchol  $i$  artikuláciou. Artikuláciou je aj vrchol s poradím 1, pokiaľ sme pri prehľadávaní z neho vyrazili aspoň dvoma rôznymi smermi.

**1125.** Problémom je rozhodnúť pri konflikte, ak chce viac písmeniek ísť na to isté políčko. Asi treba do poľa vygenerovať pre všetky písmenká smery, kam sa chcú hýbať. Potom teba pre každé políčko, na ktoré chce ísť viac kandidátov, náhodne vybrať jedného. Ostatným treba oznámiť, že sa hýbať nebudú. Teraz máme už iba otvorené a uzavreté refaze, v ktorých sa písmenká budú hýbať. Stačí každú refaz posunúť a zapamätať si, že ju netreba druhý krát posúvať.

**1131.** Drevorubačský príklad. Najjednoduchšie je prejsť pre každé slovo všetky riadky, stĺpce aj uhlopriečky. Výhodné je používať čo najrýchlejšiu procedúru pre hľadanie výskytu podrefazca v refazci. Tá, aj keď to nie je triviálne, sa dá napísať lineárne. Jedno z riešení je použiť Pascalovský *Pos*, alebo si pozrieť, ako sa to robí (úloha 1222).

**1132.** Všetko treba schovať do rekurzcie. Ak nájdeme napríklad '(', rekurzívne skontrolujeme výraz za ňou a ak bol dobrý, nasleduje za ním buď ')', alebo niektorá z otvárajúcich zátvoriek '(', '<', '{'.

**1133.** Stupeň čísla zrejme nezávisí od poradia čífer, preto najmenšie číslo s daným stupňom bude mať cifry vzostupne usporiadané. Čísla, obsahujúce párnú cifru a 5 majú stupeň max. 2. Cifry 5,5 svedčia o stupni najviac 3. Ďalej vieme, že niektoré dvojice čísel sa dajú vymeniť za iné cifry alebo dvojice čífer (napríklad 2, 2 vymeníme za 4). Takto vieme veľmi obmedziť počet čísel, ktoré treba prezrieť.

**1134.** Ťažký príklad. Vraj existuje lineárne riešenie. Treba zistiť, či je nejaký graf rovinný. Nato existuje *Kuratowského*<sup>33</sup> veta: Graf je rovinný práve vtedy, ak neobsahuje podgraf izomorfný s  $K_{3,3}$  alebo  $K_5$  ( $K_5$  má 5 vrcholov pospájaných každý s každým,  $K_{3,3}$  má 6 vrcholov v 2 trojiciach, každý z jednej trojice je spojený s každým v druhej). Dva grafy sú homomorfné, ak po nahradení všetkých vrcholov stupňa 2 hranou, ktorá ich „obchádza“

<sup>33</sup> Kuratowski Kazimierz, 1896–1980, poľský matematik, práce o funkciách reálnej premennej, matematickej logike, teórii grafov, topológii a teórii množín.



a vyhodení všetkých vrcholov stupňa 1 z obidvoch grafov dostaneme izomorfné (rovnaké) grafy.

**1135.** Dosť tvrdé je používať špeciálny riadiaci súbor pre každý formát. Jednoduchšie je každý formát natvrdo „zadrôtovať“ do kódu programu. Výber formátu potom môžeme spraviť parametrami na príkazovom riadku.

**1141.** a) Asi najrýchlejšie bude vytvárať si strom, v ktorom budú v uzloch čísla na príslušnej pozícii v permutácii. Stačí sa týmto stromom náhodne pohybovať, vytvárať pre každú novú permutáciu jej vetvu a pamätať si, ktoré vetvy sú už plné, t.j. už sa tam nezmestí ďalšia permutácia. b) Ak pripustíme s nenulovou pravdepodobnosťou opakovanie permutácií, môžeme použiť program 6.1 z [38]:

```
for i := 2 to n do begin
  t := randominteger(1, i);
  v := a[t]; a[t] := a[i]; a[i] := v
end
```

Za predpokladu, že funkcia *randominteger* generuje rovnomerne rozložené náhodné čísla, je pri prechode cyklom každá s *i* permutácií rovnako pravdepodobná. Teda po skončení cyklu je rovnako pravdepodobná každá z  $2 \cdot 3 \cdot \dots \cdot n = n!$  permutácií.

**1142.** Majme číslo  $c = a_0 \cdot 10^0 + a_1 \cdot 10^1 + a_2 \cdot 10^2 + \dots + a_n \cdot 10^n$ . Na výpočet zvyšku  $c \bmod 7$  potrebujeme vypočítať zvyšky  $(a_i \cdot 10^i) \bmod 7$ . Zvyšky  $10^i \bmod 7$  nadobúdajú periodicky hodnoty 1, 3, 2, 6, 4, 5. Program sa nachádza v jednom z konečného počtu stavov určených doterajším zvyškom a zvyškom  $10^i$ . Tieto stavy je možné si zapamätať vo vetvení programu. Premennú použijeme na načítanie ďalšej cifry.

**1143.** *Anagram* nejakého slova je slovo, ktoré vznikne z pôvodného slova utriedením jeho písmen. Všetky prešmyčky jedného slova majú rovnaké anagramy. Slová utriedime v prvom rade podľa anagramov, ale pri rovnosti anagramov podľa poradia na vstupe. Skupiny treba vypisovať v správnom poradí – udržiavajte si informáciu o tom, kde sa pri triedení nachádza prvok z pôvodnej *i*-tej pozície.

**1144.** Dáta je vhodné uchovávať v dynamických štruktúrach – vyhľadávacie stromy.

**1145.** Definované makrá si pamätáme v tvare dvoch reťazcov: pôvodného a nahradeného. Keď pri čítaní vstupu narazí na lomítko nasledované nejakým výrazom, hľadáme vyhovujúce makro. Pri tom treba testovať, či vyhovuje tvar podvýrazov (predpokladáme jednoznačnosť priradenia podvýrazov). Pretože zadanie pripúšťa vnorenie makier, výsledok ukladáme do bufferu a ďalej spracovávame.

**1211.** Příklad možno riešiť napríklad rekurziou. Základná myšlienka spočíva v tom, že ak vieme vypísať zoznam pre *n*-rozmernú kocku, zoznam pre *n* + 1 rozmernú kocku bude vyzeráť takto: najprv vypíšeme zoznam pre *n*-rozmernú kocku a pred každé číslo vrcholu pridáme 0 a potom obrátený zoznam pre *n*-rozmernú kocku, pričom pred každé číslo vrcholu pridáme 1. Tento algoritmus možno ešte vylepšiť, dá sa totiž ukázať, že desiatkový zápis čísla *i*-teho vrcholu v zozname pri tomto postupe bude  $f(i) = i \text{ xor } (i \text{ div } 2)$ .

**1212.**

```
var x : string;
begin
  x := '{}''{}';
  if x = '}' then writeln('nie')
  else writeln('ano');
end.
```

**1213.** Tento príklad možno riešiť pomocou dynamického programovania. Stačí nám pre všetky možné „vyjedenia“ kompótov v *i*-tom riadku spočítať najväčší možný objem. To ale vieme urobiť rýchlo pomocou výsledku podobného výpočtu v predchádzajúcom riadku.

**1214.** Riešením príkladu je jednoduchý backtracking. Pre jeho urýchlenie uvažujte symetriu v úlohe a ďalšie podmienky, ktoré môžu vylúčiť existenciu riešenia na začiatku výpočtu (parita počtu políčok a pod.)

**1215.** Najmenší počet *Donwloadov* dosiahneme vtedy, keď z pamäti vyhodíme vždy to písmeno, ktoré sa vo zvyšku súboru vyskytuje najneskôr.

**1221.** Riešenie založíme na tvrdení: Nech máme postupnosť všetkých permutácií  $n$  prvkov takú, že každé dve susedné permutácie sa líšia výmenou dvoch prvkov - nazvime túto postupnosť  $X$ . Potom postupnosť permutácií  $n+1$  prvkov s požadovanou vlastnosťou môžeme z tejto vytvoriť tak, že z každej permutácie  $n$  prvkov vytvoríme  $n+1$  permutácií  $n+1$  prvkov vhodným vsunutím  $n+1$ -ho prvku.

**1222.** Zreťazíme dvakrát za sebou prvé slovo a dáme v ňom vyhľadávať druhé. Jediným problémom ostáva vyhľadávanie podreťazca v reťazci.

Používa sa metóda vyhľadávania podreťazcov (pattern matching). Pattern – vzorka je reťazec, ktorý hľadáme v druhom. Jej rýchlosť sa ocení ešte viac, keď dochádza k viacnásobnému vyhľadávaniu rovnakého patternu. Metóda spočíva v tom, že ku vzorke sa vytvorí takzvaná chybová funkcia  $f$ , ktorej definičným oborom sú všetky pozície písmen vo vzorke. Ku každému písmenu, ktoré je na nejakej pozícii, sa vypočíta hodnota, ktorá zachytáva situáciu, keď sa daná vzorka hľadá v nejakom slove a po túto pozíciu boli všetky písmená rovnaké, ale teraz sa už nezhodujú. Vtedy nejdeme hľadať nový začiatok vzorky v texte, ale funkcia vyjadruje, ktoré písmeno vzorky to ešte stále môže byť, keď nastane nezhoda na tejto pozícii, pričom nová pozícia by mala byť čo najväčšia. Formálne povedané, keď máme hodnotu  $s$ , ktorej zodpovedá podreťazec  $u$  vo vzorke od jej prvého po  $s$ -tého písmeno a podobne písmenu  $t$  zodpovedá reťazec  $v$ , potom  $f(s) = t$  práve vtedy, ak  $v$  je najdlhším vlastným prefixom  $u$ .

Po vytvorení funkcie už iba prechádzame základným reťazcom a hľadáme prvé písmená vzorky. Potom robíme porovnávanie jednotlivých písmen vzorky, pričom keď dôjde ku chybe, zmeníme iba pozíciu vo vzorke a pokúšame sa porovnávať ďalej. Chybová funkcia sa v niektorých prípadoch môže aplikovať viacnásobne.

Zložitosť tohoto algoritmu je súčet dĺžok obidvoch reťazcov, v našom prípade majú slová dĺžku  $2n$  a  $n$  (pre rôzne dĺžky slov sú slová automaticky rôzne), teda zložitosť je iba  $O(n)$ . Pozri [9 v 1, 34 v 6, 6.7 v 30, 19 v 37, 1.12 v 43].

**1223.** Nech pri poslednom spájaní spájame integrácie ABCD a EFG. Potom je zrejmé, že je úplne jedno, či najprv vytvoríme ABCD a potom EFG alebo najprv EFG a potom ABCD, alebo chvíľu spájame jedno a chvíľu druhé. Aby však bolo výsledné spájanie optimálne, musia byť aj jednotlivé časti (ABCD a EFG) pospájané optimálne.

To nás navádza na riešenie pomocou dynamického programovania. Vypočítame si optimálne spájanie pre každý úsek integráčov a ukladáme si medzivýsledky do poľa. Ak postupujeme od najmenších úsekov po najväčšie, máme už všetky potrebné informácie o menších úsekoch v poli. Zložitosť takéhoto algoritmu je  $O(n^3)$ .

**1224.** Úloha preformulovaná do teórie grafov: zistite, či existuje súvislý graf, ak sú dané stupne jeho vrcholov. Využijeme dve tvrdenia:

**Tvrdenie 1:** Postupnosť  $s_1, s_2, \dots, s_n$ , kde  $s_1 \leq s_2 \leq \dots \leq s_n \leq n$  je postupnosť stupňov vrcholov obyčajného grafu práve vtedy, keď postupnosť  $s_1, s_2, \dots, s_{n-s_n-1}, s_{n-s_n} - 1, \dots, s_{n-1} - 1$  je postupnosť vrcholov obyčajného grafu (mesto s najväčším počtom ciest  $c$ , bude napojené na  $c$  miest s najväčším počtom ciest).

**Tvrdenie 2:** Ak  $G$  je obyčajný graf s  $n$  vrcholmi a  $H$  hranami, ktorý neobsahuje izolované vrcholy a platí  $H \geq n - 1$ , potom existuje obyčajný graf  $G'$  taký, že jeho vrcholy majú rovnaké stupne ako vrcholy  $G$  a je súvislý.

Takže v riešení stačí overiť podmienky  $0 < s_i < n$  a  $H \geq n - 1$  a potom použiť vetu. Tu je výhodné zapamätať si postupnosť stupňov tak, aby ju nebolo treba triediť.

**1225.** Úlohou je napísať jednoduchý syntaktický analyzátor jazyka. Návod ako na to možno nájsť napríklad v [5.1-6 v 42].

**1231.** Označme si  $q(n, m)$  počet partícií čísla  $n$  so sčítancami menšími ako  $m$ . Toto číslo vieme ľahko spočítať pomocou rekurentného vzťahu  $q(n, m) = q(n, m - 1) + q(n - m, m)$ . Pripomeňme, že  $q(n, 1) = 1$  a  $q(n, m) = q(n, n)$  ak  $m \geq n$ , dodefinujeme  $q(n, 0) = 0$ .

Pomocou toho ale ľahko vieme určiť prvý sčítanec partície. Rovnakým spôsobom určíme aj ostatné sčítance rozkladu. Pri vhodnej realizácii má algoritmus zložitosť  $O(n^2)$ .

**1232.** Skúste využiť súčet všetkých kartičiek, prípadne vám bude možno stačiť operácia xor.

**1233.** Použite Dijkstrov algoritmus, úloha 521 alebo [34].

**1234.** Cez bod preložíme priamku rovnobežnú s osou  $x$  a spočítajme počet tých priesečníkov s hranami mnohoúhelníka, ktoré ležia napravo od nášho bodu. Práve ak je tento počet nepárny, bod leží v mnohoúhelníku. Dajte si pozor na priesečníky s vrcholmi mnohoúhelníka. Zložitosť algoritmu je  $O(n)$  pre jeden bod.

V prípade, že predpokladaný počet parašutistov je veľký, môžeme mnohoúhelník predspracovať v čase  $O(n^2)$  tak, aby sa v ňom dal počet priesečníkov pomocou binárneho vyhľadávania určiť v čase  $O(\log n)$ .

**1235.** V zásade je možné použiť dva možné prístupy: každé okno si pamätá svoj obsah, alebo každé okno si pamätá obsah, ktorý prekrylo.

**1241.** Najlepšia permutácia je taká, pri ktorej je najmenší spoločný násobok dĺžok jej cyklov maximálny. Stačí uvažovať cykly s dĺžkou, ktorá je mocninou prvočísla alebo 1, pričom dĺžky každých dvoch cyklov sú nesúdeliteľné. Dá sa použiť dynamické programovanie –  $O(np \log n)$ , kde  $p$  je počet prvočísel menších ako  $n$ .

**1242.** Pozor, podľa zadania postupnosť dĺžok ceruziek nemusí byť permutáciou čísel 1 až  $n$ . V správnom riešení stačí zistiť, či sa dajú usporiadať podľa dĺžky ceruzky osobitne na párných a nepárnych pozíciách. Na koniec stačí skontrolovať, či je výsledná postupnosť usporiadaná.

**1243.** V reči teórie grafov by sa dalo povedať, že ide o zistenie, či je daný graf bipartitný, t.j. či pre každú hranu (výstrel) platí, že vrcholy (ľudia) majú rôzne zafarbenie (rôznu stranícku príslušnosť). Našou úlohou je zistiť toto zafarbenie (viď. [34]).

**1244.** Pri generovaní možností išlo o čo najlepšie zapamätávanie objektov, ako aj o rýchlosť vytvárania nových objektov. Dalo sa použiť veľké množstvo zlepšení, napríklad vytvárať dieliky  $n$ -trisu podľa už vytvorených dielikov  $n - 1$ -trisu.

**1245.** Na získanie čo najväčšej rýchlosti výpočtových algoritmov bolo potrebné použiť rozumné a efektívne štruktúry uchovávané dané čísla. Napríklad používanie čísel v binárnom zápise, apod. Potom samotné operácie s číslami robíme presne tak, ako to robí štandardný počítač.

**1311.** Použite Floyd-Warshallov algoritmus (pozri [34]).

**1312.** Podobne ako v príklade 1132, všetko treba schovať do rekurzcie. Napríklad dve rekurzívne procedúry, každá spracúva jedno z písmen **A** a **B**. V hlavnom programe stačí v cykle vyvolávať tieto procedúry. V prípade, že narazíme na medzeru označujúcu koniec vstupu pred dočítaním príslušného počtu písmen **A** alebo **B**, vypíšeme **ne**. Po dočítaní celého vstupu vypíšeme **vrátili**.

**1313.** Táto úloha patrí medzi tzv. „ťažké“. Riešenie založené na backtrackingu. Zamyslite sa aké vylepšenia pri preberaní všetkých možností môžete použiť.

**1314.** Rozšírme polomer stĺpov o polomer valcočloveka a okolo stien „vybudujeme“ nové steny takisto vo vzdialenosti polomeru valcočloveka. Cez takúto pôvodnú miestnosť prejde

valcočlovek práve vtedy, ak bod prejde transformovanou miestnosťou. Bod prejde transformovanou miestnosťou práve ak neexistuje množina stĺpov, ktoré tvoria súvislý útvar a spájajú protilahlé steny. Dobrou realizáciou možno získať algoritmus o zložitosti  $O(p^2)$ , kde  $p$  je počet stĺpov.

**1315.** Roznásobte výrazy a jednotlivé členy jednoznačne zotriedte. Potom možno výrazy porovnávať priamo.

**1321.** a) Najjednoduchšie je pre každé kráľovstvo prejsť všetky ostrovy a zistiť či doň patria. Výhodné keď je málo ostrovov a kráľovstiev. b) Predvypočítame si počet ostrovov v štvorčekoch  $1 \times 1$ . Potom stačí prebehnúť tie štvorčeky, ktoré tvoria kráľovstvo. Treba vyriešiť zarátanie ostrovov, ktoré ležia na hraniciach štvorčekov. c) Keď je ostrovov aj kráľovstiev veľa, je šikovné spočítať  $p_{x,y}$  počet ostrovov v kráľovstvách s vrcholmi  $[0,0]$  a  $[x,y]$ . Kráľovstvo s vrcholmi  $[x_1, y_1]$  a  $x_2, y_2$  obsahuje aspoň  $p(x_2, y_2) + p(x_1, y_1) - p(x_1, y_2) - p(x_2, y_1)$  ostrovov. Nezarátali sme ostrovy ležiace na ľavej a hornej strane kráľovstva. Tie sa dajú zarátať ľahko rovnakou technikou, stačí si spočítať počet ostrovov v pásikoch s celočíselnými súradnicami.

**1322.** a) priamočiare ale nevhodné je použitie triediaceho algoritmu založeného na vzájomnom porovnávaní hodnôt prvkov. V najlepšom prípade dostaneme algoritmus vyžadujúci  $O(n \log n)$  operácií. b) Využitie triedenia založeného na inom princípe, napríklad radix sort

**for**  $i := 1$  **to**  $d$  **do**  
    utried' stabilným algoritmom podľa  $i$ -tej cifry

Jednotky označujeme ako 1. cifru, ďalší rád ako 2. cifru, atď. Triedené čísla majú najviac  $d$  cifier. Všimnite si, že každé číslo z intervalu 1 až  $n^2$  možno zapísať ako najviac dvojciferné číslo v číselnej sústave so základom  $n$ . Ak vieme  $n$  čísel utriediť použitím približne  $n$  operácií, sme hotoví. Vďaka rozsahu čísel to vieme napríklad využitím count sortu. Pre každú hodnotu si spočítame koľko je vo výslednom usporiadaní pred ňou prvkov, čo vieme zistiť na jeden prechod údajmi. Potom už iba stačí umiestniť prvky na ich správne miesto.

**1323.** Riešte backtrackingom. Pozor na chybné heuristiky (napríklad výber novej linky s najmenšou periódou).

**1324.** Máme dve možnosti, z nich musíme overiť, ktorá je najkratšia: 1. obídeme celý kruh; 2. začneme s otáčaním v smere hodinových ručičiek, potom sa vrátíme a to isté na druhej strane.

**1325.** Tento príklad bol zaujímavý. :-)

**1331.** Riešenie je založené na tvrdení, že Tomáš sa pri optimálnom zápise môže zapísať na najskôr končiacu prednášku. Algoritmus je teda jasný: zotriedime prednášky podľa toho, kedy končia, vyberieme z nich prvú a zrušíme tie, ktoré sa s ňou prekrývajú atď. Časová zložitosť pri dobrej realizácii je  $O(n \log n)$ .

**1332.** a) Najjednoduchšie je postupne skúšať, či daný vzťah platí v sústave so základom 2, 3, ... Nevýhodou je, že ak by sme nemali obmedzenie na veľkosť cifier, takýto program by v prípade, že úloha nemá riešenie neskončil. b) Lepšie je takéto riešenie. Nech  $A = a_n z^n + a_{n-1} z^{n-1} + \dots + a_0$ ,  $B = b_m z^m + b_{m-1} z^{m-1} + \dots + b_0$  a  $C = c_l z^l + c_{l-1} z^{l-1} + \dots + c_0$ . Ak  $a_i + b_i = c_i$ , výraz platí vo všetkých sústavách so základom  $\geq z$ , kde  $z = 1 + \max_{0 \leq k \leq l} \{c_k\}$ . V prípade, že  $a_i + b_i \neq c_i$ . Nech  $i$  je najmenšie také, potom platí  $z = a_i + b_i - c_i$ . Treba ešte overiť, či  $(a_i + b_i) \bmod z = c_i$ .

**1333.** Úlohou je nájsť medián – to možno v čase  $O(n)$  (pozri napríklad [43]). Pozor si treba dať na (a špeciálne ošetriť) tie náleziská, ktoré sú na tej istej  $x$ -ovej súradnici.

**1334.** Dobré riešenie sa zakladá na tejto myšlienke: zakódujeme čísla do postupností písmen a čísel, kde písmená označujú ešte neurčené cifry a číslice už zistené. Napríklad ak u

čísla 123432 máme zistené, že 3 sa vytáča ako 5 a ostatné cifry sú neznáme, zakódujeme ho postupnosťou ab5c5b. Potom vyberieme na preskúšavanie taký tvar, u ktorého možno predpokladať najvyššiu pravdepodobnosť „narazenia“ na správne číslo (nezabudnite, že jednej šablóny môže zodpovedať aj viac čísel zo zoznamu).

**1335.** Tento príklad bol zaujímavý. :-)

**1341.** Použijete triedenie dvojcestným zlučováním (pozri [42]).

**1342.** a) Nešikovné riešenie je použiť postupné presúvanie druhých dielov  $b_i$ ,  $i = 1, 2, \dots, n$ , na svoje miesto. Vyžaduje to  $O(n^2)$  operácií. Rovnako náročné je aj riešenie využívajúce výmenu párných prvých dielov z nepárnych druhými dielmi, čím sa dostaneme prvé a druhé diely rovnakých kníh k sebe. Dvojice nebudú ešte všetky na svojich miestach. Zvyšné knihy vymieňame tak, aby sme zmenšovali počet dvojíc, ktoré nie sú na svojom mieste. Treba rozlišovať prípady, keď je dvojíc párny a nepárny počet.

b) Lepšie riešenie je využitím techniky rozdeľ a panuj. Knihy  $a_{p+1}, \dots, a_n, b_1, \dots, b_p$ , kde  $p = n \div 2$ , posunieme cyklicky o  $p$  miest vpravo, pozri úlohu 123 (ak je  $n$  párne, obe časti stačí vzájomne vymeniť). Dostali sme dve časti, medzi ktorými sa už knihy vymieňať nebudú. Každú z častí rovnakým spôsobom usporiadame. Posunutie  $n$  kníh vieme realizovať  $O(n)$  operáciami, takže celkovo vystačíme s  $O(n \log n)$  operáciami.

c) Postupujeme zľava doprava a vymieňame knihu, na ktorej sa nachádzame za knihu, ktorá na to miesto patrí. Problém je v tom, že po niekoľkých krokoch nie je jasné, ako určiť, kde sa nachádza kniha, ktorá na aktuálne miesto patrí. Našťastie sa to však dá vypočítať pomerne jednoduchou funkciou (*Supermiesto*):

$$\begin{aligned} \text{Miesto}(2i) &= i \\ \text{Miesto}(2i + 1) &= \text{Miesto}(i + 1) \\ \text{Supermiesto}(i) &= \text{Miesto}(i) + n, \text{ ak } \text{Miesto}(i) + n \geq i \\ \text{Supermiesto}(i) &= \text{Supermiesto}(\text{Miesto}(i) + n), \text{ inak} \end{aligned}$$

Túto funkciu možno realizovať bez rekurzie (a bez pomocnej pamäti). Pri dobrej realizácii výsledná časová zložitosť bude asi  $O(n \log n)$ .

**1343.** Možno riešiť pomocou dynamického programovania. Pre každé písmeno vety a pre každý jeho výskyt na náramku vieme z toho istého údajá pre predchádzajúce písmeno a všetky jeho umiestnenia na náramku spočítať minimálny počet otočení.

**1344.** Úlohu prevedieme na úlohu teórie grafov: nech vrcholy sú políčka miestnosti a hranami sú spojené tie vrcholy, ktoré sú spojené kachličkou vo výslednom uložení. Dostávame takzvaný bipartitný graf a našou úlohou je zistiť, či v ňom existuje párovanie (pozri [34]).

**1345.** Jediným problémom pri riešení je uvádzanie zlomkov do základného tvaru – na to použijete Euklidov algoritmus. Vnútorne v rámci operácií je potrebné zlomky uchovávať s dvojnásobnou presnosťou (napríklad ak sa bežne uchováajú v premenných typu **integer**, treba použiť na medzivýsledok **longint**).

**1411.** Riešenie so zložitnosťou  $O(n^2)$  je pomerne jednoduché. Stačí si uvedomiť, že v každej triangulácii mnohoúhelníka  $v_0, v_1, \dots, v_{n-1}$  musí existovať vrchol  $v_i$ , že trojuholník  $v_{i-1}, v_i, v_{i+1}$  patrí do triangulácie.  $v_i$  je dobrý vrchol, ak je jemu prislúchajúci vrchol konvexný a vnútri trojuholníka  $v_{i-1}, v_i, v_{i+1}$  neleží žiadny iný vrchol mnohoúhelníka. Toto vieme overiť lineárnym prebehnutím všetkých vrcholov. Ak vnútri niekto leží, pokračujeme v overovaní vrchola  $v_{i+1}$ . Ak ho môžeme odrezať, urobíme tak. V tomto prípade pokračujeme vrcholom  $v_{i-1}$ , lebo by sa mohlo stať, že uhol pri ňom sa stal konvexným. Pri takejto stratégii máme zaručené, že budeme testovať najviac  $2n$  vrcholov. V [4] je uvedený algoritmus zložitosti  $O(n \log n)$ .

**1412.** Ak sú dĺžky článkov reálne čísla, najjednoduchšie je asi napísať backtracking. Ak sú však celočíselné, zaväta to dynamickým programovaním. Poskladaná dážďovka s  $k$

článkami sa dá popísať trojicou čísel  $a, b, c$  – súradnica najľavejšieho bodu, konca dažďovky a jej najpravejšieho bodu ( $a \leq 0 \leq c$ ;  $a \leq b \leq c$ ). Nech pre všetky  $t[k, a, b]$  obsahuje najmenšie  $c$  také, že existuje poskladanie k článkov dažďovky popísaným spôsobom. Potom nie je problém vypočítať  $t[k+1, a, b]$  pre všetky  $a, b$ . Má zmysel uvažovať  $|a|, |b| \leq 2 * d$  kde  $d$  je dĺžka najdlhšieho úseku dažďovky. Stačí si pamätať  $t[k, a, b]$  len pre posledné dve hodnoty  $k$ .

**1413.** Prevedme si úlohu na problém z teórie grafov: stromy a rieky sú vrcholy, hrana medzi dvomi vrcholmi vedie práve vtedy, ak sa medzi nimi nedá pretlačiť klavír, (stromy, ktorých rozdiel  $x$ -ových súradníc je menší ako  $x$ -ový rozmer klavíra a zároveň rozdiel  $y$ -ových je menší ako  $y$ -ový rozmer klavíra; rieka a strom, ktorých rozdiel  $y$ -ových súradníc je menší ako  $y$  klavíra). Tomáš môže prejsť cez les práve vtedy, ak neexistuje v takomto grafe cesta od severnej rieky k južnej.

**1414.** Do stredu každej hrany (t.j. mosta) dajme nový vrchol (dôležité miesto). Tým sa nám hrana rozpadne na dve menšie. Teraz nám vlastne stačí zistiť, či existuje nejaký nový vrchol, ktorého odobratím by sa nám graf rozpadol. Vrcholy, ktorých odobratím graf rozpadne, sa nazývajú *artikulácie* a dajú sa hľadať modifikovaným prehľadávaním do hĺbky. Viac informácií nájdete v [3.B v 34].

**1415.** Každý počítač akoby prehľadával celú sieť na vlastnú päsť do šírky. Zostavuje si postupnosť zoznamov počítačov, vzdialených  $l$  prerútovaní ( $l = 0, 1, \dots, \text{diam}(G)$ ). Pre  $l = 0$  je v ňom len sám. Vždy, keď má zoznam pre nejaké  $l$  hotový, pošle ho všetkým susedom a očakáva ich zoznamy. Z týchto zoznamov skonštruuje svoj zoznam pre  $l+1$ . Opakuje, dokiaľ sa dozvedá nové mená, potom vie, že pozná všetkých v sieti a skončí. Pri implementácii treba dávať pozor na to, že kým od nejakého suseda ešte neprišiel zoznam pre  $l$ , od iného mohol prísť zoznam pre  $l+1$  – treba ho odložiť pre neskoršie použitie.

**1421.** Sú dva možné prístupy, oba so zložitou  $O(n^2)$ . Prvý prevádza úlohu na úlohu z teórie grafov, kde sa potom hľadajú mosty a artikulácie. Netreba zabudnúť na možný výskyt grafu v grafe (teda nesmieme zabudnúť na pôvodné zakreslenie v rovine), lebo to môže zmeniť výsledok. Druhý prístup rozoberieme trochu podrobnejšie. Vytvoríme „obdĺžnikovú sieť“ tak, že každú úsečku predĺžime až po okraj koláča. Potom stačí rekurzívne vyfarbiť okrajové časti koláča, pričom vyfarbujeme tie políčka, ktoré nie sú oddelené úsečkou. Diery zostanú nevyfarbené, pomocou ofarbovania zistíme ich počet.

**1422.** Backtracking. Rozdelíme na prípady, keď obe čísla  $u, v$  sú nenulovej dĺžky a keď aspoň jedno z nich je prázdnu postupnosťou. Predstavme si, že sa postupnosti generovali nasledovne  $w = w_1, w + 2, \dots, w_k$ , pričom  $w_k$  obsahuje  $x$  a  $w_{k-1}$  nie. V prvom prípade najskôr zistíme, či sa  $x$  nachádza vo  $w$ . Ak, áno, odpovedáme **ano**, inak začneme od postupnosti  $x$  späťne generovať tie úseky  $x'$  možných predchodcov  $w_{k-1}$ , z ktorých sa  $x$  mohlo vygenerovať. Úseky  $x'$  hľadáme najkratšie možné. Ich dĺžku vieme ľahko obmedziť podľa postupností  $u, v$ , napríklad určite bude nanejvýš tak dlhé ako  $x$ . Teraz rekurzívne robíme to isté so všetkými  $x'$  ako predtým s  $x$ . Aby sme sa nezacyklili, testujeme, či už také  $x'$  náhodou nebolo testované predtým. Na to by sa zišlo vedieť prezeráť zásobník v rekurzii, teda je jednoduchšie implementovať rekurziu pomocou vlastného zásobníka. Po neúspešnom skončení prehľadávania odpovedáme **nie**.

V prípade, že sú obe  $u, v$  prázdne, odpovedáme **ano** práve vtedy, keď sa  $x$  nachádza vo  $w$ . Nech je teda  $u$  prázdne a  $v$  neprázdne. Potom buď  $x$  je vo  $w$  a odpovedáme **ano** alebo tam  $x$  nie je. Označme  $p_w$  počet dvojok vo  $w$  a  $p_v$  vo  $v$ . Ak  $p_w = 0$ , odpovedáme **nie**. Ak  $p_w = 1$  a  $p_v \leq 2$ , odpovedáme podľa toho, či je  $x$  vo  $v$ . Ak  $p_w \geq 1$  a  $p_v \geq 2$ , naša odpoveď závisí iba od toho, či sa  $x$  nachádza v postupnosti  $v^n$  pre nejaké  $n$  ( $v^n$  je  $n$ -krát za sebou napísaná postupnosť  $v$ ). A na záver ak  $p_w \geq 1$  a  $p_v \leq 1$ , potom treba overiť, či  $x$  je z  $v^{p_w}$ .

**1423.** Lineárne riešenie je založené na Dirichletovom princípe (Pidgeon Hole Principle): Najväčšia vzdialenosť dvoch susedných vrabcov je aspoň  $\frac{\max - \min}{n-1}$ , kde  $\min$ , resp.  $\max$

je vzdialenosť najbližšieho, resp. najvzdialenejšieho vrabca od dediny. Ak by najväčšia vzdialenosť dvoch susedných vrabcov bola menšia, potom by boli najvzdialenejší a najbližší vrabec vzdialení menej ako  $\max - \min$ , čo je spor.

Drôt rozdelíme na neprekrývajúce sa úseky o veľkosti  $\frac{\max - \min}{n-1}$ . Pre každý úsek zistíme vrabca najviac napravo a najviac naľavo. Potom pre  $i$  od 1 po  $n-1$  porovnávame vzdialenosť najpravejšieho vrabca z  $i$ -teho úseku s najľavejším vrabcom z najbližšieho neprázdného  $j$ -teho úseku smerom napravo. Pokračujeme s  $i := j$  a pamätáme si nájdené maximum. Toto riešenie má časovú zložitosť  $O(n)$ .

**1424.** Vzorovým riešením je Ford-Fulkersonov algoritmus [6 v 34] na hľadanie maximálneho toku v sieti so zložitou  $O(nm)$ . Označme  $f_{ij}$  veľkosť toku medzi stanicami  $i$  a  $j$ , teda  $f_{ij}$  je 1, ak tam rum tečie, je 0 ak netečie a je  $-1$  ak tečie v opačnom smere. Pre každú dvojicu staníc  $i, j$ , ktoré sú spojené rúrou spočítame rezervu rúry, t.j. o koľko rumu viac sme schopní prepraviť rúrou z  $i$  do  $j$  oproti súčasnemu stavu. Rezerva je teda  $r_{ij} = 1 - f_{ij}$ .

Na začiatku pre každú rúru z  $i$  do  $j$  platí  $f_{ij} = 0$ . Začneme zo stanice  $s$  hľadať cestu do stanice  $t$  po rúrach s nenulovou rezervou. Ak nájdeme takúto rezervnú cestu, každej rúre na ceste zvýšime  $f_{ij}$  o 1, čím sa celkové množstvo prepraveného rumu zvýši o 1. Takto zväčšujeme tok dovtedy, kým ešte existuje nejaká rezervná cesta. Nájdenie rezervnej cesty trvá  $O(m)$  a veľkosť toku je nanajvýš  $n-1$ , celková časová zložitosť je teda  $O(nm^2)$ .

**1425.** Za šéfa zvolíme počítač, ktorého meno je posledné v abecede. Každý počítač  $A$  si v danom momente bude o počítači  $B$  myslieť, že je jeho „lokálny šéf“, ak je meno počítača  $B$  najďalej v abecede z počítačov známych počítaču  $A$ . Na začiatku je každý počítač sám sebe „lokálnym šéfom“. Právo posilať nové správy majú iba „lokálni šéfi“, cez ostatné počítače nezmenené správy len prechádzajú. Posilajú sa len tri typy správ –  $\hat{S}$ +meno oznamuje meno „lokálneho šéfa“,  $U$ +meno uznáva meno za svojho nového „lokálneho šéfa“ a  $K$ +meno oznamuje koniec – určil sa šéf.

Každý počítač si okrem svojho mena pamätá meno svojho „lokálneho šéfa“. Na začiatku každý počítač pošle po oboch linkách správu typu  $\hat{S}$  so svojím menom. Keď nejakému počítaču príde správa typu  $\hat{S}$  s menom, ktoré je menšie ako meno jeho „lokálneho šéfa“, túto správu ignoruje. Ak je toto meno väčšie, aktualizuje svojho „lokálneho šéfa“ a po tej istej linke pošle späť správu typu  $U$ . Najzaujímavejší je prípad, keď sa toto meno rovná menu „lokálneho šéfa“. Vtedy ak správa prišla z toho istého smeru ako ostatné správy od tohto šéfa, len ju pošleme ďalej. Ak ale prišla z opačného smeru, uvedené meno je šéfom pre celú sieť. Potom pošleme správy typu  $K$  po oboch linkách. Počet prenesených správ za predpokladu približne rovnako rýchlych liniek je  $O(n \log n)$ , v priaznivom prípade  $O(n)$ .

**1431.** Keď zotriedime priesečníky priamok s kružnicou polárne podľa počiatku súradnicovej sústavy (t.j. podľa stredu kruhu), dostávame pomerne jednoduché kritérium pretínania sa priamok: pre priesečníky  $A, B$  priamky  $p$  a  $C, D$  priamky  $q$ , pričom bez ujmy na všeobecnosti nech je  $A < B, C < D$  a  $A < C$ , sa priamky  $p, q$  pretínajú práve vtedy, ak je  $C < B < D$ . Stačí teda, ak budeme postupne prechádzať obvod kružnice. Keď narazíme na počiatočný bod úsečky, tak ju zaradíme do našej vyhľadávacej štruktúry, keď narazíme na jej koncový bod, tak ju vyberieme. Pri vkladaní úsečky stačí potom určiť počet vložených úsečiek s menším koncovým bodom – keďže tieto úsečky sa už nachádzajú v štruktúre, ich koncový bod je väčší ako počiatočný bod vkladanej úsečky. Ide teda o všetky úsečky s menším počiatočným bodom pretínajúce danú úsečku. Keď prejdeme celý obvod kružnice, budú určené počty týchto priesečníkov pre všetky úsečky. Ich súčet však zároveň zahŕňa všetky priesečníky priamok vnútri kruhu. Pri použití vhodnej vyhľadávacej dátovej štruktúry napríklad vyvážený binárny strom a pri efektívnej implementácii sa jedna operácia vykoná v čase  $O(\log n)$ . Celková časová zložitosť je teda  $O(n \log n)$ .

**1432.** Vzorový program po načítaní vstupu najprv skontroluje, či je riešenie zjavne nemožné, čo je v prípade ak je trojuholníkov iný počet, ako počet vrcholov mínus 2, alebo ak

je súčet obsahov trojuholníkov iný ako obsah mnohouholníka, alebo ak existuje trojuholník, pre ktorého jednu stranu neexistuje príslušná dĺžka uhlopriečky.

Ak riešenie nevyhlúčil postupuje rekurzívne – na kraji mnohouholníka nájde trojuholník, aký má v zozname (že taký musí existovať, ak sa to dá, si iste dokážete sami). Skúsi ho teda odrezať a rekurzívne sa zavolá. Mnohouholník sa trojuholníkmi dá pokryť, ak nakoniec zostane z mnohouholníka trojuholník zhodný s tým, čo ostal v zozname. Ak sa tak nestane po vyskúšaní všetkých možností, trojuholníky sa na plech poukladať nedajú.

**1433.** Ak požadujeme, aby sme nepoužili viac, ako dvojnásobok minimálneho počtu krabíc, tak stačí, aby program vymyslel riešenie, v ktorom budú všetky krabice naplnené aspoň do polovice (dôkaz pozri nižšie, problémy ešte môžu nastať s poslednou krabicou). Vzorové riešenie pracuje veľmi jednoducho – zo vstupu číta objemy vecí a postupne ich ukladá do krabice, kým sa tam vojdú. Keď nejaká vec do krabice nevojde, tak sa krabica uzavrie, otvorí sa nová a pokračuje sa v balení. Ak však objem veci, ktorá nevošla do krabice bol väčší ako  $\frac{s}{2}$ , otvorená krabica sa neuzavrie, ale vec s nadpolovičným objemom sa zabalí do osobitnej krabice, tá sa uzavrie a pokračuje sa v balení do krabice, ktorá bola otvorená predtým.

Treba dokázať, že počet krabíc nájdeného riešenia neprekročí dvojnásobok minimálneho počtu krabíc, do ktorých je veci možné zabaliť. Stačí si všimnúť, že všetky krabice (okrem poslednej) budú zaplnené viac ako z polovice. Naraz program pracuje len s jednou otvorenou krabicou. Krabica sa uzavrie len vtedy, ak do nej nevojde vec s objemom nanajvyš  $\frac{s}{2}$ , t.j. objem veci v uzavretej krabici je vždy väčší ako  $\frac{s}{2}$ . Nech optimálne balenie potrebuje  $l$  krabíc a nami nájdené riešenie potrebuje  $k$  krabíc. V našom riešení sú v každej krabici veci s objemom viac ako  $\frac{s}{2}$  a teda ich celkový objem je viac ako  $\frac{s(k-1)}{2}$ . V optimálnom balení sa do krabice zmestí najviac  $s$ , a preto  $l > \frac{k-1}{2}$ .  $l$  je však celé číslo, čiže  $l$  je aspoň najmenšie celé číslo väčšie ako  $(k-1)/2$ , čo v matematike zapíšeme ako  $l \geq \lfloor \frac{k-1}{2} \rfloor + 1$ . Ak je  $k$  párne, dostávame  $l \geq \lfloor \frac{k-1}{2} \rfloor + 1 = \frac{k-2}{2} + 1 = \frac{k}{2}$ , ak je  $k$  nepárne, máme  $l \geq \lfloor \frac{k-1}{2} \rfloor + 1 = \frac{k-1}{2} + 1 = \frac{k+1}{2}$ . Teda v oboch prípadoch  $k \leq 2l$ , čo sme chceli dokázať.

**1434.** Máme nájsť ofarbenie planárneho grafu piatimi farbami tak, aby žiadne dva vrcholy spojené hranou nemali rovnakú farbu. Takže: pri načítavaní si spravme (iste neprázdny, dá sa dokázať) zoznam všetkých vrcholov stupňa nepresahujúceho 5. Tento zoznam budeme udržiavať tak, aby v ňom boli vždy všetky nespracované vrcholy stupňa najviac 5, avšak nebudeme požadovať, aby sa každý vrchol vyskytoval iba raz, alebo aby tam neboli iné vrcholy.

Spracovať vrchol stupňa menšieho ako 5 znamená: zapamätať všetkých jeho susedov, vymazať ho z grafu (ak pri mazaní hrán klesne nejakému vrcholu stupeň na 5, treba ho zaradiť), rekurzívne spracovať zvyšok grafu, nájsť farbu, nevyskytujúcu sa u jeho susedov a ofarbiť ho touto farbou.

Pri spracovávaní vrchola v stupňa 5 musíme dosiahnuť, aby dvaja z jeho susedov mali rovnakú farbu. Iste existujú susedia  $v_i$  a  $v_j$ , ktorí nie sú spojení hranou. Zmažeme vrchol  $v$  a vrcholy  $i, j$  skontražujeme. Takýto skontražovaný graf bude opäť planárny. Rekurzívne ho ofarbíme,  $v_{ij}$  rozbijeme naspäť na  $i$  a  $j$  a vidíme, že vrchol  $v$  susedí s najviac štyrmi rôznymi farbami. Keď je zoznam nespracovaných vrcholov stupňa nanajvyš 5 prázdny, buď  $G$  nebol rovinný, alebo sme už spracovali (t.j. ofarbili) všetky vrcholy.

**1435.** Oba krajné počítače začnú číslovanie. Sebe priradia jednotku, susedovi pošlú dvojku a každý ďalší počítač pošle susedovi, od ktorého číslo nedostal, číslo o jedna väčšie. Takto číslovania postupujú z oboch strán, až sa v istom okamihu stretnú. Napríklad pri siedmich počítačoch to môže vyzeráť napríklad takto: 1 2 3 4 3 2 1. Vidíme, že jedna strana od miesta stretnutia by si čísla mohla ponechať, ale druhú ešte musíme prečíslovať. Teda keď počítač dostane druhé číslo, porovná ho s prvým. Ak je menšie alebo rovné, pošle ďalej nulu a ako



svoje definitívne číslo si priradí prvé číslo. Jedným smerom teda bude postupovať nula, ktorá je vždy menšia ako číslo daného počítača. Táto strana si takto ponechá pôvodné číslo. Ak bude druhé číslo väčšie ako prvé, počítač si nechá druhé číslo a ďalej pošle číslo o jedna väčšie ako druhé. Takto sa aj druhá strana postupne prečísľuje.

**1441.** Rozdeľuj a panuj v čase  $O(n \log n)$ . Rozdelíme uhly na dve rovnako veľké množiny  $A$  a  $B$ , rekurzívne spočítame prienik uhlov v množine  $A$  (konvexný útvar  $K_A$ ) a prienik uhlov v množine  $B$  (konvexný útvar  $K_B$ ) a potom spočítame prienik  $K_A$  a  $K_B$ .

Otázkou zostáva, ako spočítať prienik dvoch konvexných útvarov. Rozdelíme rovinu vodorovnými priamkami, ktoré prechádzajú vrcholmi konvexných útvarov, na pásy. V každom páse vieme spočítať prienik v konštantnom čase. Prienik dvoch konvexných útvarov (ak počet vrcholov je  $m$ ) dokážeme spočítať v čase  $O(m)$ .

**1442.** Dynamické programovanie. Počítame pre všetky podreťazce postupne od najkratších až po najdlhšie. Časová zložitosť  $O(n^3)$ , pamäťová  $O(n^2)$ .

**1443.** Hľadáme počet  $n + 1$ -prvkových postupností  $a'_0, \dots, a'_n$ , pre ktoré platí:

$$\begin{aligned} a'_0 &= 0 \\ \forall l, 0 \leq l \leq n : 0 \leq a'_l &\leq k \\ \forall l, 0 \leq l \leq n-1 : |a'_l - a'_{l+1}| &= 1 \end{aligned}$$

a navyše  $a'_n = 0$  ( $a'_i = k - a_i$ ). Označme  $p_{i,j}$  počet postupností  $\{a'_k\}_{k=0}^j$  dĺžky  $j + 1$  s uvedenými vlastnosťami, pre ktoré platí  $a_j = i$ . Z toho, že nasledujúci člen postupnosti sa od predchádzajúceho líši práve o jedna, ľahko vyrobíme rekurentný vzťah:

$$\begin{aligned} p_{0,0} &= p_{i,0} = 0 \quad \text{pre } 0 < i \\ p_{0,j+1} &= p_{1,j} \\ p_{i,j+1} &= p_{i-1,j} + p_{i+1,j} \quad \text{pre } 0 < i < k \\ p_{k,j+1} &= p_{k-1,j} \end{aligned}$$

Na základe tohoto vzťahu môžeme dynamickým programovaním vypočítať v čase  $O(nk)$  a s pamäťou  $O(k)$  hodnotu  $p_{0,n}$ , čo je počet kľúčikov.

Iné riešenie: Dá sa previesť na umocňovanie matíc rozmeru  $(k + 1) \times (k + 1)$  na  $n$ . Tak dostávame zložitosť  $O(F(k) \log n)$ , kde  $F(k)$  je zložitosť násobenie matíc rozmeru  $k \times k$ . Triviálne  $F(k) = O(k^3)$ , existujú však algoritmy so zložitostou  $F(k) = O(k^{2.49})$ . Výhodné pre „dlhé“ kľúčiky.

**1444.** Ide o počet oblastí v rovinnom grafe. Stačí použiť známu Eulerovu vetu:  $s + v = h + k + 1$ , kde  $v$  je počet vrcholov,  $h$  je počet hrán,  $k$  je počet komponentov súvislosti a  $s$  je počet oblastí.

**1445.** Algoritmus bude v podstate sekvenčný. Ide o klasickú úlohu hľadania artikulácie v grafe. Pozri napríklad príklad 1124.

**z1411.** Najjednoduchšie je napísať si procedúru  $Otoc(i, j : \text{integer})$ , ktorá otočí úsek poľa od  $i$  po  $j$ , t.j. vymení prvky  $a_i$  a  $a_j$ ,  $a_{i+1}$  a  $a_{j-1}$  atď. Potom celý problém vyrieši postupnosť príkazov  $Otoc(i, j)$ ;  $Otoc(j+1, k-1)$ ;  $Otoc(k, l)$ ;  $Otoc(i, l)$ ;

**z1412.** Najrozumnejšie je asi napísať funkciu, zisťujúcu počet dní od napríklad 1.1.1901. Pozri aj úlohu 232.

**z1413.** Treba vlastne zistiť, či je graf, ktorého vrcholy tvoria zámky a hrany cesty, súvislý. To môžeme robiť prehľadávaním do šírky či do hĺbky, viac viď. [citKuc; 34]. Iný prístup je, že komponenty súvislosti budeme tvoriť postupne počas čítania vstupu. Využijeme algoritmus Union/Find. V prípade, že získame jeden komponent súvislosti je odpoveď kladná a netreba ani dočítať celý vstup.

**z1414.** Treba prehľadávať všetky možné začiatky a diferencie. Niekoľko fint: postupnosť dĺžky  $n$  má diferenciu deliteľnú všetkými prvočíslami menšími ako  $n$ . Členy postupnosti sú (veľké) prvočísla, a teda nebudú deliteľné malými prvočíslami (napríklad  $\leq 20$ ). Z toho nám vyplývajú nejaké obmedzenia na možné prvé členy postupnosti. Testovanie prvočíselnosti sa dá robiť pravdepodobnostne (napríklad Rabinov test) [33.8 v 6; 5 v 11; 10.6 v 25] a nakoniec, keď nájdeme kandidáta na postupnosť, overíme si, či sú všetky členy naozaj prvočísla. A najdlhšia nájdená postupnosť?

Zaciatok: 503213 počet: 14 diferencia: 4504500

**z1415.** Užitočné finty: Nič sa nestane, ak  $k \, dx \, (dy)$  pripočítame násobok  $2n \, (2m)$ . Môžeme ich teda upraviť tak, aby  $-n < dx < n \, (-m < dy < m)$ . Potom počas jedného skoku naraziť najviac do jednej severojužnej a jednej východozápadnej steny. Najjednoduchšie bolo skočiť bez ohľadu na steny a potom spätne zobraziť žabu zrkadlovo podľa tých stien, ktoré „preskočila“. Tieto operácie sa dajú robiť nezávisle pre  $x$ -ovú a  $y$ -ovú súradnicu.

**z1421.** Najprv upravme pole  $B$ : pre všetky  $i$  priradíme  $B[i] := B[i] - c$ . V poli  $B$  teraz treba nájsť súvislý úsek s najväčším súčtom (pozri úlohu 213). Náčrt riešenia so zložitou  $O(d)$ : Označme  $s_k$  súčet prvkov poľa  $B$  od 1 po  $k$ . Súčet prvkov od  $j$  po  $k$  teraz získame ako  $s_j - s_k$ . Tento súčet pre konkrétne  $k$  bude maximálny, keď  $s_{j-1}$  bude minimálne. Algoritmus bude pre každé  $k$  zisťovať maximálny súčet končiaci  $k$ -tým prvkom a z týchto čísel určí najväčšie. Pre dané  $k$  minimálnu hodnotu  $s_{j-1}$  zistíme v konštantnom čase z minimálnej hodnoty  $s_{i-1}$  pre číslo  $k-1$ , totiž buď  $j = i$  alebo  $j = k$ . Teda s inicializujeme v čase  $O(d)$  a samotný algoritmus beží tiež v čase  $O(d)$ .

**z1422.** Keby nebolo jazier, a všetky nuly by predstavovali more, stačilo by pre každé pevninové políčko zistiť, či hranou susedí s morom – potom by bolo určite na pobreží. Teda treba odlíšiť more od jazier. Na mape je more súvislý úsek z núl, ktorý obsahuje políčko  $(1, 1)$ . Jednoduchým rekurzívnym ofarbovacím algoritmom možno more označiť napríklad číslom 2, a to tak, že označíme políčko  $(1, 1)$  a rekurzívne sa zavoláme na všetkých jeho ôsmich susedov.

**z1423.** V podstate ide o hľadanie mediánu (stredného prvku) v postupnosti. Hoareho algoritmus podobný Quicksort-u hľadajúci  $k$ -ty najmenší prvok je založený na myšlienke preusporiadať pole tak, aby hľadanou susedí s morom – potom by bolo určite na pobreží. Teda treba odlíšiť more od jazier. Na mape je more súvislý úsek z núl, ktorý obsahuje políčko  $(1, 1)$ . Jednoduchým rekurzívnym ofarbovacím algoritmom možno more označiť napríklad číslom 2, a to tak, že označíme políčko  $(1, 1)$  a rekurzívne sa zavoláme na všetkých jeho ôsmich susedov.

Načrtujeme tiež myšlienku algoritmu s časovou zložitou  $O(n)$  aj v najhoršom prípade. Finta spočíva v lepšom výbere prvku, podľa ktorého rozdelujeme pole na dva menšie úseky. Pole najskôr rozdelíme na päťprvkové postupnosti a za pivota zvolíme medián postupnosti mediánov päťprvkových postupností (medián päťprvkovej postupnosti vieme určiť v konštantnom čase, na určenie mediánu mediánov použijeme tento istý algoritmus, ale už iba na  $n/5$  prvkov). Po preusporiadaní poľa podľa takto určeného pivota je ľahko vidno, že väčší z úsekov naľavo od pivota alebo napravo od pivota má najviac  $\frac{3}{4}n$  prvkov. Na tento úsek poľa znovu použijeme uvedený algoritmus. Ak  $T(n)$  označíme čas výpočtu nášho algoritmu, potom podľa vyššie uvedeného platí  $T(n) = c_1n + T(n/5) + T(3n/4)$ , z čoho vyplýva, že  $T(n) = O(n)$ .

**z1424.** Úloha sa nazýva aj Collatzov problém. Sú v podstate dva možné prístupy: backtracking alebo systematické „drevorubačské“ skúšanie možností. Pri backtrackingu začneme od 1 a snažíme sa postupne skúšať inverzné operácie k tým popísaným v zadaniach. Postupne získavame dlhšie a dlhšie postupnosti. Lepšie sa ale prekvapujúco javí postupné skúšanie všetkých nepárnych čísel, párne potom ľahko dostaneme  $-(2k+1)2^l$  má postupnosť  $d+i$ ,

pričom  $2k+1$  má postupnosť dĺžky  $d$ . Najlepší doterajší výsledok:  $n = 63728127$ , postupnosť má dĺžku 950 členov a  $\frac{p(n)}{c(n)} = 118.75$ .

**z1425.** Jednoduchá simulácia. Nemá zmysel v každom kroku prekresľovať celé vláčky, vždy len zmažeme koniec a nakreslíme nový začiatok (rušeň). Tak isto nemusíme posúvať celé pole, len si zaznačíme, na ktorom políčku trasy je práve rušeň. Pri posúvaní rušňa vždy otestujeme, či nenastala zrážka.

**z1431.** Existuje iba osem transformácií matice: Identita, otočenie o  $90^\circ$ , otočenie o  $180^\circ$ , otočenie o  $270^\circ$ , súmernosť podľa osi  $y$ , súmernosť podľa osi  $x$  a otočenie o  $90^\circ$ , súmernosť podľa osi  $x$  a otočenie o  $180^\circ$ , súmernosť podľa osi  $y$  a otočenie o  $270^\circ$ . Každá operácia zložená zo základných operácií zo zadania je ekvivalentná s niektorou z vymenovaných. Preto stačí skontrolovať iba každú z týchto ôsmich.

**z1432.** V hrách tohto typu uvažujeme spravidla nasledovne: Majme pole, do ktorého indexujeme jednojednoznačne podľa momentálneho stavu hry. Konkrétne – ak sú dve kôpky guľôčok veľkosti  $m$  a  $n$ , bude pole napríklad dvojrozmerné a indexovať sa bude hodnotami  $m, n$ : *pole*[ $m, n$ ]. V tomto poli budeme mať uloženú jednobitovú informáciu – či existuje z tohto stavu pre hráča, ktorý je na ťahu vyhrávajúca stratégia. Pole môžeme vybudovať takto: Označíme za prehrávajúce tie stavy, ktoré sú určené zadáním hry. Za vyhrávajúce potom označíme tie stavy, z ktorých sa dá pre súpera dosiahnuť prehrávajúci stav. Ďalej analogicky: prehrávajúci stav je taký, keď akýmkoľvek ťahom dosiahneme pre súpera vyhrávajúci stav, atď. Takto hravo zistíme, či je začiatočný stav vyhrávajúci. Vo väčšine prípadov však toto pole netreba, vyhrávacosť stavu sa dá zistiť aj jednoduchými matematickými vzorcami.

**z1433.** Predstavme si orientovaný graf – vrcholmi sú kone, hrany sú od koňa A ku B vtedy, ak sa niekto stavil, že A skončí pred B. Úlohou je nájsť také usporiadanie vrcholov, že zo žiadneho vrchola nevedie šípka do nejakého vrchola, ktorý je v poradí pred ním. Inými slovami: treba topologicky utriediť vrcholy orientovaného grafu, alebo povedať, že sa to nedá (pozri [34]).

**z1434.** Zadanie je jednoduchá tzv. monoalfabetická unilaterálna substitučná šifra, t.j. písmeno za písmeno podľa tabuľky. Šifra v druhej časti príkladu je posun o 25, 13, 22 a 24 znakov s periódou zámeny pochopiteľne 4. Na nájdenie týchto konštánt stačí použiť vhodnú heuristiku so zameraním na často opakujúce sa slová.

**z1435.** Jednoduchá simulácia. Na zistenie zacyklenia mohamedánov použijeme metódu dvoch bežcov: Námestie simulujeme v dvoch poliach, jedno sa za sekundu zmení raz, druhé dvakrát. Testujeme zhodnosť týchto polí. Ak sú po určitom čase obe polia rovnaké a stále sa niektorí otáčajú, znamená to, že sme našli cyklus. Zapamätáme si konfiguráciu námestia, aby sme pokračovali simuláciou do tohto istého stavu, označujúc si mohamedánov, ktorí sa aspoň raz pohnú. Potom sa už ľahko spočítajú označení, t.j. mohamedáni, ktorí sa nikdy neprestanú otáčať.

**1511.** Najlepšia bola implementácia tzv. hašovaním. Popis hašovania nájdete v [41]

**1512.** Pozri príklady 412 a 643.

**1513.** Použijeme modifikáciu Floyd-Warshallého algoritmu, tak aby zohľadňoval dobu priletu na letisko. Rátame pre všetky možné časy, ktorých je len  $24 \cdot 60$ . Takýto algoritmus má časovú zložitosť  $O(tn^3)$  kde  $t$  je počet minút v dni a  $n$  je počet liniek. V prípade nízkeho počtu lietadiel (riedkeho grafu) sa to dá vhodnou dátovou štruktúrou zoptimalizovať na  $O(d^2n^3)$  kde  $d$  je počet letov za deň.

**1514.** Táto úloha bola ťažká. Úloha sa dá neefektívne riešiť prehľadávaniami do šírky aj do hĺbky (kým nám stačí pamäť). Vzorovým riešením je algoritmus  $A^*$ , ktorý je nad rámec tejto zbierky. Aj nad váš. Aj nad náš. Tak to nechajme tak.

**1515.** a) Použijeme pomocnú funkciu, ktorá si bude posledné dve hodnoty odovzdávať ako parameter. b) Položíme si  $x = 0$  a to po jednej zvyšujeme, kým  $x^2 < n$ . Zlepšiť môžeme napríklad binárnym vyhľadávaním, prípadne lepším výpočtom druhej mocniny.

**1521.** Vytvoríme si prekladač (v podstate automat, ktorý v každom kroku prečíta znak zo vstupu a na základe znaku a zmeny stavu vypíše jeden znak). Ten dostane vstup jeden riadok stránky a bude vyznačovať tie miesta, na ktorých sa vo vstupe nachádzal ako podslovo niektorý riadok Tekilovho znaku (čísla z rozsahu 0 až b, kde 0 znamená žiadny). Tým dostaneme novú maticu obsahujúcu čísla 0..b. Utvoríme postupnosť  $d_0..d_{b-1}$ , kde  $d_i$  je najmenšie číslo, že  $z_i = z_{d_i-1}$ . Potom sa znak nachádza na stránke práve vtedy, keď postupnosť  $d_0..d_{b-1}$  je podreťazcom niektorého stĺpca novej matice.

**1522.** Netradične, napíšeme celý vzorák. **begin** *writeln*(6); **end**. Skúste si spraviť dôkaz.

**1523.** Príklad si môžeme premeniť na úlohu z teórie grafov. Nech vrchol v grafe reprezentuje štvoricu čísel, a to políčko  $(x,y)$  a vektor rýchlosti  $(v_x,v_y)$ , ktorou sme do políčka prišli. Ďalej nech vrchol  $i$  je spojený s vrcholom  $j$  práve vtedy, keď sa zo stavu vrchola  $i$  vieme dostať do stavu vrchola  $j$ . Úlohu teraz môžeme preformulovať na nájdenie najkratšej cesty v grafe z vrcholu  $(x_{start}, y_{start}, 0, 0)$  do ľubovoľného vrcholu s polohou cieľového políčka. Toto sa dá riešiť prehľadávaním (či už do šírky, alebo do hĺbky).

**1524.** Ak by boli všetky body v jednom riadku, môžeme tie nepotrebné jednoducho „vyhryznúť“. Napríklad pre body  $(1,1)$ ,  $(4,1)$ ,  $(6,1)$  ich orámujeme obdĺžnikom  $(0,0)$ ,  $(7,0)$ ,  $(7,2)$ ,  $(0,2)$  a postupne „vyhryzávame“ nežiadúce body, čím vznikne tento zoznam  $(0,0)$ ,  $(2,1)$ ,  $(3,1)$ ,  $(1,0)$ ,  $(5,1)$ ,  $(2,0)$ ,  $(7,1)$ ,  $(0,2)$ . Keby sme tento algoritmus potrebovali rozšíriť pre viac riadkov, budeme z vrcholu  $(7,1)$  pokračovať vo „vyhryzávaní“ druhého riadku.

**1525.**

1. potrebujeme pomocnú funkciu  $R$  s pomocným parametrom  $a$ , v ktorom sa vytvára obrátený zoznam. Potom funkciu  $Rev$  ľahko dostaneme,  $Rev(x) = R(x, 0)$ .  $R(0, a) = a$ ,  $R((u, v), a) = R(v, (u, a))$ .
2. využíva  $Rev$  z predchádzajúcej úlohy.  $Append(x, y) = R(Rev(x), y)$
3. môžeme spraviť ľubovoľný sort, ľahko implementovateľný je napríklad *Mergesort*. Potrebujeme urobiť funkcie *Msplit*, ktorá rozdelí zoznam na dve časti (pre jednoduchosť párne a nepárne pozície) a *Merge*, ktorá dva utriedené zoznamy spojí.

$$Mergesort(0) = 0,$$

$$Mergesort(u, 0) = u, 0,$$

$$Mergesort(u, v) = Merge(Mergesort(a), Mergesort(b), 0) \leftarrow a, b = Msplit(0, 0, u, v)$$

**1531.** Dá sa aj lepšie ako lineárne riešenie. Najefektívnejšie je použiť tri haldy. V jednej budú uložené ženy pred rozkvetom, v druhej po rozkvetu (podľa krásy). A v tej tretej budú opäť ženy pred rozkvetom, tentokrát usporiadané podľa toho, koľko im ostáva do rozkvetu. Takto vieme na silvestra pomocou tretej haldy ľahko nájsť ženy, ktoré treba presunúť z prvej haldy do druhej. Na operáciu EXITUS potrebujeme vedieť kde sa ktorá žena nachádza. Preto si ich ešte raz uložíme do nejakej efektívnej štruktúry, napríklad AVL-stromu alebo hašovacej tabuľky.

**1532.** a) Platí  $f_n = 3f_{n-1} - f_{n-2}$ . Toto už len stačilo lineárne vypočítať - pamätať si stále hodnoty dvoch predchádzajúcich krokov. b)  $f_n = Fib_{2n-2}$  a  $n$ -té Fibonacciho číslo sa dá vypočítať v logaritmickom čase pomocou známeho vzorca.

**1533.** Otázky si prepíšeme do tvaru  $s_{k-1} - s_l \leq -m$  alebo  $s_l - s_{k-1} \leq m$  podľa toho či hovoria **aspoň** alebo **najviac**. Vytvorme si graf, kde každú takúto nerovnicu premeníme na orientovanú hranu.  $s_a - s_b \leq c$  prepíšeme na hranu z  $a$  do  $b$  ohodnotenú číslom  $c$ . Teraz

už len pomocou Floyd-Warshallovho algoritmu zistíme či existuje záporný cyklus. Ak áno, šaman klamal.

**1534.** Preskúšanie všetkých možností, dalo sa však silne optimalizovať. Skúšame všetky natočenia darčiekov ( $3^n$ ) a pre každé natočenie polynomiálne zistíme ako darčeky preusporiadať – najdlhšia nerastúca podpostupnosť – pozri 522. Takto získavame riešenie v čase  $O(3^n n^2)$  alebo  $O(3^n n \log n)$ , ak použijeme inú metódu na zistenie preusporiadania darčiekov.

**1535.** Funkcia *member* sa dala triviálne implementovať v logaritmickom čase, horšie to bolo s *insert* a *delete*. Keďže sme si pri stromoch nemohli pamätať žiadnu informáciu navyše, nedala sa dosiahnuť lepšia ako lineárna zložitosť. Najlepšie je strom skonvertovať na zoznam, na tom previesť samotnú operáciu, a potom zoznam skonvertovať naspäť na strom.

**1541.** Rovinu si narežeme na pásy priamkami, ktoré prechádzajú cez všetky priesečníky kružníc. Takto nám vznikne nanajvýš  $n^2$  pásov. Každý pás obsahuje nanajvýš  $2n$  oblúkov, ktoré sa nepretínajú. Keď si pásy utriedime a takisto aj oblúky v pásoch, vieme použiť binárne vyhľadávanie, ktoré nájde správne políčko v čase  $O(\log n)$ . Predspracovanie urobíme použitím zametania, čo nám dá  $O(n^3)$ .

**1542.** Preskúname všetky možnosti polozenia papiera na darček - tých je  $4n^2$ . Ak máme nejaké uloženie papiera, tak prehľadávaním do hĺbky postupne darček obaľujeme, až zistíme či sa dá obaliť alebo nie. Toto prehľadávanie má zložitosť  $O(n^2)$  čiže celý program je  $O(n^4)$ .

**1543.** Úloha je vlastne hľadanie maximálneho toku zo začiatočných políčok von z mesta. Použijeme Ford-Fulkersonov algoritmus. Pozri 1424.

**1544.** Rastovú päťu si prepíšeme na graf, kde každému otlaku priradíme dva vrcholy. Jeden symbolizuje prelepenie otlaku horizontálne, druhý vertikálne. Ak nejaké prelepenie otlaku  $X$  implikuje prelepenie iného otlaku  $Y$  istým spôsobom, položíme hranu z príslušného vrcholu otlaku  $X$  do príslušného vrcholu otlaku  $Y$ . Z každej dvojice vrcholov teraz chceme ofarbiť práve jeden tak, aby farby vrcholov na hrane boli identické. Toto sa dá triviálne v čase  $O(n^2)$ , pre riešenie v čase  $O(n)$  si pozrite [3.E v 34].

**1545.** Výraz roznásobíme a rovnaké členy sčítame. Používame známe pravidlá na narábanie s mnohočlenmi. Ak nám nakoniec  $z \cdot a - b$  vyjde 0, oba výrazy sú ekvivalentné.

**z1511.** Každú súvislú oblasť ofarbiť a otestovať či to je znak  $Z$ . Najlepšie bolo spraviť priemet na os  $Y$  a spočítať koľko znakov je v tom-ktorom riadku. Očakávame v každom riadku jednu jednotku a v dvoch riadkoch aspoň tri. Na záver ešte otestovať či je to šikmá čiara. Program beží v čase úmernom počtu políčok.

**z1512.** Strom kreslíme rekurzívne. Nakreslíme rovnú čiaru, otočíme sa, menší podstrom, otočíme sa, menší podstrom a vrátime sa na začiatok. Dôležité je to, že po každej čiare prejdeme práve dvakrát a minimalizujeme otáčanie.

**z1513.** Existujú dva prístupy: Prehľadávanie do šírky, alebo do hĺbky. Prehľadávanie do šírky používa front, a do hĺbky používa zásobník. Oba postupy sú rovnocenné a majú časovú zložitosť rovnú počtu políčok. Pri prehľadávaní do hĺbky ofarbíme jedno políčko a rekurzívne sa zavoláme na našich neofarbených susedov. Pre optimalizácie ešte bolo potrebné neprekresľovať všetky okná v každom kroku.

**z1514.** Kratšiu stranu ihriska označíme ako  $\vec{k}$  a dlhšiu ako  $\vec{d}$ . Každý bod roviny vieme teraz zapísať ako  $A + x\vec{k} + y\vec{d}$  kde  $A$  je vrchol ihriska. Čísla  $x$  a  $y$  vypočítame tak, že si túto rovnicu rozložíme na  $x$ -ovú a  $y$ -ovú zložku, a vyriešime sústavu dvoch lineárnych rovníc. Teraz podľa čísel  $x$  a  $y$  jednoducho určíme kto je hráč a aký. Hráči majú  $x$  a  $y$  medzi 0 a 1. Hráči na jednej polovici majú  $x \leq 0.5$  a na druhej  $x \geq 0.5$ .

**z1515.** Existujú tri prípady podľa veľkosti mincí  $p$  a  $q$  ( $p \leq q$ ): a)  $p = 1$  Dajú sa zaplatiť všetky kombinácie. b)  $\text{nsd}(p, q) > 1$  Nedá sa zaplatiť nekonečne veľa kombinácií. c)

$\text{nsd}(p, q) = 1$  a  $p > 1$  Najväčšia nezaplátiteľná suma je  $pq - p - q$ . Najväčší spoločný deliteľ nájdeme známym Euklidovým algoritmom.

**z1521.** V prvom rade si treba uvedomiť, že stačí robiť iba slimákov otočených vpravo (vľavo otočení slimáci sú s nimi osovo súmerný). Taktiež oddelíme 1-mesačného slimáka ako osobitný prípad. Ostatní slimáci majú rozmery  $(4n - 1)(4n - 3)$  a môžeme si ho po stĺpcoch rozdeliť na tri časti: stĺpce 1 až  $2(n - 1) + 1$ , stĺpce  $2n$  až  $4(n - 1)$  a zvyšok. Pre jednotlivé časti sa dajú, zvlášť pre párne, zvlášť pre nepárne stĺpce odvodiť vzorčky pre výskyt \*. Potom stačí v dvoch cykloch prebehnúť celého slimáka a tlačíť obrázok po znakoch.

**z1522.** Bludisko stupňa stupňa 0 prejdeme triviálne ako chodbičku dĺžky  $d$ - príkaz *Rovno* ( $d$ ). Ak postupnosť príkazov pre bludisko stupňa  $n$  nazveme  $B(n)$ , príkaz *Vľavo* nazveme  $L$  a *Vpravo*  $R$ , tak postupnosť pre bludisko stupňa  $n + 1$  sa dá napísať takto:  $B(n); R; B(n); L; B(n); L; B(n); B(n); L; B(n); L; B(n); L; B(n); B(n)$ .

**z1523.** pozri 813.

**z1524.** Pre každé zvolené políčko  $P$  (jeho ľavý dolný roh má súradnice  $[x_0, y_0]$ , kvadrantom  $Q$  nazveme všetky body „vľavo a pod“), vieme popísať počet oblôh, ktoré sa na ňom nachádzajú, keď za každý ľavý dolný a pravý horný roh oblohy, ktorý sa nachádza v  $Q$ , pripočítame 1 a za každý ľavý horný a pravý dolný roh oblohy, ktorá sa nachádza v  $Q$  odrátame 1. Teda nám stačí pre všetky políčka chleba spočítať súčty v príslušných kvadrantoch. Ak označíme  $q_{x,y} = \sum_{0 \leq i \leq y} \sum_{0 \leq j \leq x} p[i, j]$  a  $s_{x,y} = \sum_{0 \leq i \leq x} p[i, y]$ , potom  $q_{x,y} = q_{x,y-1} + s_{x,y}$ , čo sme potrebovali.

**z1525.** Program je veľmi jednoduchý a využíva rekurzívne volania. Keď rozdelím nejaký interval  $(l, p)$  na polovicu, najskôr zaštipujem rovnakým algoritmom jeho ľavú časť  $(l, \frac{l+p}{2})$  a potom pravú  $(\frac{l+p}{2}, p)$ .

**z1531.** V každom políčku tabuľky si pamätáme koľko úloh môže marsochod vykonať ak začína tu. Pre políčko kde je stena tam položíme  $-1$ , pre prázdne políčko tam dáme maximum z políčov vpravo a dole, a pre políčko s úlohou tam dáme  $1 +$  maximum z políčov vpravo a dole. Takúto tabuľku vyplníme oddola a nakoniec si v začiatočnom políčku prečítame maximálny počet úloh, ktoré môže splniť. Pri rekonštrukcii cesty sa vyberieme na to políčko, ktoré má väčšiu hodnotu.

**z1532.** Kreslíme rekuziou podľa zadania. Treba dávať pozor na to, že veža veľkosti 1 vyzerá trochu inak ako ostatné.

**z1533.** Znak sa dá nakresliť jedným ťahom, ak vrcholov s nepárnym stupňom je buď 0 alebo 2. Ak sú dva musíme začať v jednom a skončiť v druhom, inak skončiť tam kde sme začali. Ideme zo začiatku a chodíme po hranách náhodne, kým neprídeme do nášho cieľa. Teraz na tejto našej ceste nájdeme všetky vrcholy z ktorých ešte vedú neoznačené hrany a spustíme sa rekurzívne hľadajúc cyklus z toho vrcholu do toho istého vrcholu. Keďže každú hranu prezrieme práve raz, celková časová zložitosť je úmerná počtu hrán.

**z1534.** Budík sa mohol zaseknúť iba tak, že dve dotýkajúce sa kolieska sa otáčajú opačne. Smer otáčania môžeme zistiť buď prehľadávaním do šírky alebo do hĺbky. Oba prístupy sú rovnako efektívne.

**z1535.** Diery si utriedime podľa vzdialenosti od začiatku. Potom ideme zľava doprava a položíme nový koberček na prvé nezakryté miesto. Takéto riešenie je optimálne. Triedenie je  $O(n \log n)$  a samotný algoritmus je  $O(n)$ .

**1611.** Použijeme dátovú štruktúru nazývanú intervalový strom. Koreň tohto stromu bude zodpovedať intervalu  $(1, 2^p) = n$ , vrchol zodpovedajúci intervalu  $a..b$  má synov, zodpovedajúcich intervalom  $(a, (a + b - 1)/2)$  a  $((a + b + 1)/2, b)$ . V každom vrchole si budeme udržiavať jednu hodnotu tak, aby výška políčka bola súčtom hodnôt intervalov, ktoré ho

obsahujú. Okrem toho si v každom vrchole budeme pamätať, koľko políčok z jeho intervalu má výšku aspoň 1. S takouto dátovou štruktúrou vieme všetky požadované operácie robiť v polylogaritmickom čase.

**1612.** Každú kružnicu môžeme posunúť tak, aby na jej okraji ležali aspoň 3 význačné útvary (strany obdĺžnika alebo body). Takýchto kruhov je  $O(n^3)$ , skontrolovať všetky z nich vedie k riešeniu v  $O(n^4)$ . Lepšie riešenie sa dá dosiahnuť, keď si uvedomíme, že najväčšia hľadaná kružnica sa buď dotýka steny, alebo má stred vo vrchole Voronoiovoho<sup>34</sup> diagramu daných bodov. Voronoiovo diagram sa dá nájsť v čase  $O(n \log n)$ , podrobnejšie viď [kap. 7 v 4.2.8 v 24].

**1613.** Riešenie sa skladá z dvoch fáz – v prvej zistíme použitím Floyd-Warshallovho algoritmu najkratšie cesty medzi mestami a v druhej metódou dynamického programovania nájdeme hľadaný počet mítingov. Na to ich najskôr utrieme podľa času konania a potom pre každý zrátame, koľko sa ich najviac stíha, ak je tento posledný.

**1614.** Dynamické programovanie cez všetky podreťazce. Keď vieme, koľko najmenej znakov potrebujeme na to, aby sme na palindróm doplnili všetky súvislé podreťazce dĺžky nanaajvýš  $k$ , vieme si túto hodnotu ľahko zrátať aj pre podreťazce dĺžky  $k+1$ . Keď má prvé a posledné písmeno rovnaké, potrebujeme rovnaký počet písmen ako na reťazec, ktorý dostaneme ich odstránením. Ak nie, buď doplníme prvé písmeno na koniec alebo posledné na začiatok a zvyšok reťazca doplníme na palindróm. Pre zvyšok reťazca však už v oboch prípadoch vieme, koľko znakov potrebujeme, takže len vyberieme lepšiu možnosť.

**1615.** Program hľadá cestu z  $[0,0]$  do  $[n-1, n-1]$  po nulových políčkach. V deterministickom programe by sme na to použili prehľadávanie do šírky.

**1621.** Hlavná myšlienka riešenia je predpočítať si dĺžky úsekov, ktoré je Kin schopný prejsť za jeden deň. Najprv pre každý kameň určíme, za aký najkratší čas sa k nemu vieme dostať z dediny. Určíte k nemu pôjdeme buď tak, že vyjdeme z dediny k nejakému kameňu a potom už ideme len jedným smerom. Teraz si všimnime, že kamene, dosiahnuteľné od jedného kameňa, tvoria v každom smere súvislý úsek. Preto nám stačí len ísť od kameňa, kým sa dá. Teraz si pre každý kameň spočítame, kam najďalej vieme od neho za jeden deň dôjsť v smere číslovania kameňov a kam proti smeru. Keby sme teraz vedeli, ktorým kameňom začína prvý deň a ktorým smerom ide, ľahko by sme našli optimálne riešenie. Zoberme totiž prvý kameň, ktorý nestihol navštíviť. Nejaký deň ho navštíviť musí a nič nestratíme tým, ak v ňom začne, ani tým, ak to bude na druhý deň. Túto úvahu môžeme ďalší deň zopakovať, až kým nedostaneme riešenie. Teraz len preskúšame všetky možnosti, ako mohol Kin ísť prvý deň. Výsledná časová zložitosť tohto algoritmu je  $O(n^2)$ , aj keď všetky jeho časti okrem poslednej sa dajú napísať v čase  $O(n)$ .

**1622.** Triviálne riešenie v čase  $O(n^3)$  je založené na myšlienke, že každú kružnicu, ležiacu vnútri mnohouholníka, môžeme posunúť tak, aby sa dotýkala dvoch jeho hrán. Lepšie riešenie je založené na nasledovnej myšlienke: Pre každú hranu mnohouholníka zoberme jej „zakázaný ovál“ – množinu všetkých miest, kde sa nemôže nachádzať stred kružnice, lebo by ju pretínala. Ak vyhovujúca kružnica existuje, tak určite existuje vyhovujúca kružnica, ktorej stred leží na vnútornej časti obvodu niektorého z oválov. Tá sa skladá z úsečky a dvoch polkružníc, ktoré nás zaujímajú len vtedy, ak je príslušný uhol mnohouholníka nekonvexný. Pre každý z týchto útvarov otestujeme, či na ňom existuje bod, ktorý neleží v žiadnom zo zvyšných oválov. Tento prístup vedie k riešeniu v čase  $O(n^2 \log n)$ . Riešenie v čase  $O(n^2)$  dosiahneme nasledovným postupom – udržujeme si množinu všetkých bodov, ktoré sú vnútri mnohouholníka a sú od neho vzdialené aspoň  $s$ . Na začiatku položíme  $s = 0$  a túto množinu tvorí celý mnohouholník. Teraz postupne s zväčšujeme (skokovito, nová

<sup>34</sup> Вороной Георгий Федосеевич, 1868–1908, ruský matematik. Venoval sa najmä teórii čísiel.

hodnota je taká, pri ktorej sa nám tvar hranice zmení), až kým nie je  $s \geq r$ . Ak je teraz množina neprázdna, kružnica existuje, ináč nie. Vraj existuje aj riešenie v čase  $O(n \log n)$ .

**1623.** Kto s kým chce tancovať si zaznačíme v bipartitnom grafe. Teraz chceme jeho hrany ofarbiť čo najmenším počtom farieb tak, aby hrany vychádzajúce z 1. vrchola mali navzájom rôzne farby. Určite potrebujeme aspoň toľko hrán, aký je najväčší stupeň vrchola. Dá sa ukázať, že toľkoto farieb naozaj stačí. Ofarbujeme postupne graf. Zoberme nejakú hranu, ktorej teraz nevieme nájsť vhodnú farbu. V každom z jej koncových vrcholov máme nejakú voľnú farbu, nech sú to farby  $a$  a  $b$ . Ak  $a = b$ , vyhrali sme, hranu ofarbíme touto farbou. Ak nie, chceme hranu z druhého vrchola, ktorá má farbu  $a$ , prefarbiť na  $b$ . Týmto nám v jej koncovom vrchole mohli vzniknúť dve hrany rovnakej farby, tak druhú z nich prefarbíme na  $a$  a pokračujeme. Rozmyslite si, že tento proces bude konečný. Teraz už máme v oboch vrcholoch voľnú tú istú farbu a môžeme našu hranu ofarbiť.

**1624.** Najprv si spočítame, aký by mal byť celkový počet 1 a 0 na poslednom bite čísel študentov, keby prišli na prednášku všetci (to vieme spraviť bez jedinej otázky). Potom sa spýtame na posledný bit každého čísla v zozname, a keďže vieme, koľko malo byť 1 (alebo 0), vieme povedať, či chýba 1 alebo 0. Čísla v zozname, ktoré sa nekončia chýbajúcim bitom, už nebudeme uvažovať. Zoznam sme skrátili na  $\lfloor \frac{n}{2} \rfloor$ . Takto iterujeme, pokiaľ neurčíme všetky bity. Položíme  $n + n/2 + n/4 + \dots + n/2^n < 2n$  otázok.

**1625.** Najprv si uvedomme, že Amália vie uhádnuť číslo z intervalu  $(1, n)$  v čase  $O(\log n)$ , lebo sa vieme pýtať na cifry v binárnom zápise. Vlastný algoritmus je priamočiary. Najskôr Amália každému mestu z papyrusu priradí nejaké mesto na mape. Kontrolujeme, aby rôznym mestám z papyrusu nepriradila to isté mesto mapy. Ešte ostáva overiť cesty z papyrusu. Každé ceste z papyrusu prislúcha nejaká postupnosť miest na mape pospájaná cestami. Postupnosti prislúchajúce rôznym cestám nesmú mať žiadne spoločné vnútorné mesto. Počiatočné a koncové mesto postupnosti vieme určiť na základe Amálinho priradenia, vnútorné mestá nám tiež napovie Amália. Treba však kontrolovať, či sú hádané mestá skutočne pospájané na mape cestami, aj to, či sa niektoré vnútorné mesto nevyskytuje vo viacerých postupnostiach.

**1631.** Použijeme Dijkstrov algoritmus. Keďže je však tento graf riedky (z každého vrcholu idú 4 hrany), pokúsime sa jeho zložitosť zlepšiť. Toto dosiahneme haldou, pomocou ktorej budeme vedieť rýchlo nájsť najbližší vrchol na spracovanie. Celková časová zložitosť takéhoto programu je potom  $O(mn \log m)$ .

**1632.** Skontrolujeme okrajové prípady (Roh, bod a stena). Riešenie bude vyzeráť tak že preložíme všetkými dvojicami bodov kružnicu s polomerom  $r$  a zrátajme koľko je v ktorej bodov. Takéto riešenie by však malo časovú zložitosť  $O(n^3)$ . Pre každý bod si preto polárne utriedime stredy kružníc, ktoré sa dotýkajú toho bodu a ešte nejakého iného. V takejto štruktúre už vieme polahky zrátať kde je najviac hrozienok. Celková časová zložitosť je  $O(n^2 \log n)$ .

**1633.** Dynamickým programovaním. Nech  $D[i, j]$  je minimálna dĺžka špagátu na pokrytie kolíkov  $i$  až  $j$ . Toto nájdeme tak že sa pokúsime tento mnohouholník rozbiť na dva segmenty, ktorých pokrytie už poznáme. Ak postupujeme od trojuholníkom k vyšším mnohouholníkom, táto podmienka bude splnená. Celková časová zložitosť je  $O(n^3)$ .

**1634.** a) Dynamické programovanie v čase  $O(kn \log n)$  - počítame najprv pre jedného pisára, pre dvoch, troch, atď. Jedno políčko tabuľky vieme vyplniť v logaritmickom čase, nakoľko môžeme použiť binárne vyhľadávanie. b) Odhadová metóda v čase  $O(n \log t)$ . Ak uhádneme čas  $t$ , za ktorý to pisári majú prepísať, vieme v lineárnom čase povedať, či to naozaj stihnú alebo nie. Tento čas môžeme odhadovať pomocou binárneho vyhľadávania. Ako keď hádate číslo a máte odpovede málo/veľa.

**1635.** Amália uhádne uloženie všetkých kúskov ( $O(k \log k)$ ) a my už len skontrolujeme, či



sa nejaké dva kusy neprekrývajú. ( $O(k^2)$ ). Použitím haldy a intervalového stromu sa dá aj v čase  $O(k \log k)$ .

**1641.** Vstup čítame po riadkoch. Naraz máme v pamäti vždy dva po sebe nasledujúce riadky. Súvislé úseky si ofarbujeme navzájom rôznymi farbami, keď sa nám niektoré úseky spoja, spojíme aj príslušné farby. Toto sa dá robiť triviálne použitím algoritmu Union-Find. Takéto riešenia majú podľa implementácie zložitosť  $O(mn \log^* n)$  až  $O(mn \log n)$ . Zložitosť  $O(mn)$  sa dá dosiahnuť drobnou fintou – keď máme v minulom riadku použité farby 1 až  $f_1$ , ofarbíme nový farbami  $f_1 + 1$  až  $f_2$ . Následne len zjednotíme nové farby, patriace do tej istej oblasti.

**1642.** Vzorové riešenie je backtracking, teda skúšanie všetkých možností. Keďže všetkých možností rozmiestnenia satelitov je nekonečne veľa, budeme skúšať len tie rozumné. Stačí si uvedomiť, že každú plochu môžeme posunúť tak, aby niektorý z bodov ležal na jej ľavej strane a niektorý na hornej.

**1643.** Úlohou bolo nájsť najširšiu kostru. Najširšia kostra je vlastne zároveň aj maximálna a vieme ju nájsť napríklad v čase  $O(m \log m)$  Kruskalovým algoritmom. Iná myšlienka riešenia v čase  $O(m \log m)$  je usporiadať si hrany podľa hrúbky a binárne vyhľadávať šírku hľadanej kostry (čo je najmenšia taká šírka, že keď zoberieme len hrany aspoň tak široké, budú tvoriť súvislý graf). Takéto riešenie sa dá pomocou niekoľkých ďalších netriviálnych myšlienok (hľadanie mediánu namiesto triedenia, algoritmus Union-Find a pod.) zlepšiť až na  $O(m)$ .

**1644.** Vstupný graf si rozbijeme na dva acyklické orientované grafy – graf zjazdoviek a graf vlekov. Oba topologicky zotriedime a následne metódou dynamického programovania nájdeme v prvom grafe medzi každými dvoma vrcholmi najdlhšiu a v druhom najkratšiu cestu. Z toho už vieme ľahko nájsť riešenie pôvodnej úlohy.

**1645.** Stačí overiť, či sa dá dostať z 1 do 2, z 2 do 3, ..., z  $n$  do 1. Ak sa z niektorého vrchola do ďalšieho dostať nedá, zjavne graf silne súvislý nie je a naopak ak sa to u všetkých dá, silne súvislý je. A ako overíme, či sa dá z  $i$  dostať do  $i + 1$ ? Jednoducho – Amálka nám hovorí cestu a my ju len kontrolujeme.

**z1611.** Union-find algoritmus. Pre každého človeka si pamätáme jeho šéfa, pričom šéf družstva má šéfa samého seba. Úplného šéfa niekoho hľadáme rekurzívne tak, že kontrolujeme šéfov, šéfov šéfov ... až kým neprideme k šéfovi družstva. Toho potom nastavíme ako šéfa všetkým ľuďom po ktorých sme k nemu došli (aby sme si nabudúce ušetrili robotu). Keď teraz jedno družstvo prehrá z druhým, tak obom zistíme ich šéfov a tomu slabšiemu šéfovi priradíme ako jeho šéfa toho víťazného. Takéto riešenie má časovú zložitosť  $O(\log^* n)$ .

**z1612.** Táto úloha sa dala riešiť iba backtrackingom, nakoľko je to NP-úplný problém.

**z1613.** Jednoduché rekurzívne riešenie. Ak chceme z tyče A na tyč B presunúť pomocou tyče C disky o výške  $n$ , tak urobíme toto: Presunieme z tyče A na tyč C pomocou tyče B disky o výške  $n - 1$ , presunieme jeden disk z A na B a nakoniec presunieme z C na B pomocou A disky o výške  $n - 1$ .

**z1614.** Použijeme prehľadávanie do hĺbky, pričom si pre každú stanicu pamätáme, z ktorej stanice sme do nej prvýkrát prišli (ak vôbec). Ak sa vieme dostať do nejakej stanice kde sme už boli, máme vyhraté.

**z1615.** Stačí začínať buď z prázdneho alebo plného hrnca, pričom nás nemusí trápiť to, či ho neskôr preplníme alebo vyprázdňime. Hľadáme riešenie úlohy  $n = xa + yb$ . Najväčší spoločný deliteľ ( $\text{nsd}(a, b)$ ) má podobnú vlastnosť, a naša úloha je teda riešiteľná vtedy keď je  $n$  násobkom  $\text{nsd}(a, b)$ . Ak je násobkom, tak potrebné koeficienty vieme vyrátať popri počítaní  $\text{nsd}$  známym algoritmom.

**z1621.** Byrokratov si uložíme do haldy. Halda je pole o veľkosti  $n$ , kde prvok na políčku s indexom  $x$  má dvoch synov, uložených v políčkach s indexami  $2x$  a  $2x + 1$ . Takisto je

každý prvok väčší ako jeho synovia. Vkladanie: Nový prvok zapíšeme na posledné políčko a pozrieme sa na jeho otca. Ak je otec menší tak ich vymeníme a pozrieme sa na otca jeho otca... Takto sa dostanem tak vysoko, ako je potrebné a vlastnosť haldy ostane zachovaná. Výber najväčšieho: Je to vždy prvé políčko. Zmažeme ho tak, že na jeho miesto presunieme posledný prvok a podobne ho presúvame nižšie a nižšie, až kým nám neplatí vlastnosť haldy. Takéto operácie majú zložitosť  $O(\log n)$  a celý algoritmus  $O(n \log n)$ .

**z1622.** Zitina metóda nie je správna. Úloha sa nazýva párovanie bipartitného grafu, a jeho riešenie nájdete napríklad v [34].

**z1623.** V tejto úlohe bolo potrebné ošetriť veľa okrajových prípadov (meče ktoré zväčšujú počet hláv draka, neúčinné meče ...) a potom vyriešiť Diofantovskú rovnicu. Takto sa dala úloha riešiť v čase potrebnom na nájdenie nsd, čo sa dá v  $O(\log n)$ .

**z1624.** Použijeme prehľadávanie do šírky, prípadne Dijkstrov algoritmus s tým, že všetky vzdialenosti medzi mestami sú rovnaké. Pre Dijkstrov algoritmus pozri úlohu 523.

**z1625.** Potrebujeme zrátať koľký je A aj B palindróm a potom nájsť palindróm číslo  $A+B$ . Koľko je palindrómov dĺžky  $x$ ? Dĺžky 1 a 2 ich je 9, dĺžky 3 a 4 ich je 90, dĺžky 5 a 6 je ich 900 ... Takto vieme koľko je palindrómov dĺžky menšej ako je naše číslo. Najmenší palindróm dĺžky  $x$  je 100...001. Nech je naše číslo  $x = 32544523$ . Palindrómy jeho dĺžky menšie ako on sú tvaru 1abccba1, 2abccba2 a potom tie 3abccba3 pre ktoré je abccba menšie ako 254452. Takto to vyrátame priamo úmerne od dĺžky čísla  $x$ . Obdobne aj inverzný proces.

**z1631.** Použijeme písmenkový binárny strom taký, že naľavo pôjde vetva Y a napravo Z. Takto bude časová zložitosť oboch operácií rovná dĺžke názvu pesničky. Viac o dátovej štruktúre strom si nájdete v literatúre. [6]

**z1632.** Škriatok VIL mal pravdu.

**z1633.** Uvedomte si aké variácie predchádzajú tú, čo chceme zakódovať. Potrebujeme si spočítať koľko je variácií A prvkov z B čísel. Toto si predpočítame a následne vieme ľahko určiť, koľko je variácií menších ako naše číslo. Stačí spočítať, koľko je variácií kratších ako naše číslo a potom tých, čo začínajú menšou prvou cifrou...

**z1634.** Použijeme upravený Dijkstrov algoritmus, ktorý zarátava cenu vrcholov a nie hrán. Pozri úlohu 524.

**z1635.** Pre všetky podniky sa snažíme znižovať dlhy cez daný podnik. Nájdeme jeho prvého dlžníka a prvého veriteľa a prevedieme dlh. Ak sme zmazali dlžníka tak nájdeme nového a naopak. Spracovanie jedného podniku trvá  $n$  krokov a pretože je  $n$  podnikov celková časová zložitosť je  $O(n^2)$ .

**1711.** Mestá a cesty reprezentujeme grafom  $G$ . Ku  $G$  zostrojíme graf  $G'$  takto: vrchol  $v_i$  grafu  $G$  rozdelíme na dva  $v_{iM}$  a  $v_{iG}$ . Medzi vrcholmi  $v_{xM}$  a  $v_{yG}$  povedie hrana, ak v pôvodnom grafe bola cesta medzi mestami  $x$  a  $y$ , pričom v meste  $x$  jej koniec strážili mušketeri a v meste  $y$  gardisti. Dĺžku hrany nezmeníme. Ďalej pridáme dve špeciálne hrany nulovej dĺžky, medzi  $v_{1G}$ ,  $v_{1M}$  a  $v_{nG}$ ,  $v_{nM}$ . Na  $G'$  aplikujeme Dijkstrov algoritmus, pričom nás zaujíma najkratšia cesta medzi  $v_{1G}$  a  $v_{nG}$ . Zložitosť je  $O(m)$ , kde  $m$  je počet hrán.

**1712.** Dynamickým programovaním.  $p[a, b, c]$  si označíme minimálnu cenu na nákup kombinácie  $a, b, c$ . Každú hodnotu si uložíme do poľa  $p$  s rozmermi  $A \times B \times C$ . Jedno políčko vieme vypočítať v čase lineárnom od  $n$  tak, že preskúšame kúpiť postupne všetky lístky a odkážeme sa do tabuľky na už vypočítané políčko  $p[a - p_a, b - p_b, c - p_c] + \text{cena}$ . Celková časová zložitosť algoritmu je  $O(ABCn)$ .

**1713.** Zjavne keď si povieme súradnice, kam chceme cvičencov dostať, vieme ich tam dostať tak, aby každý šiel vodorovne aj zvisle len jedným smerom – nemusia sa obchádzať. Ako určiť  $y$ -ovú súradnicu, na ktorú majú ísť? Ľahko nahliadneme, že súčet vzdialeností, ktoré

cvičenci prejdú, bude minimálny, ak za  $y$  zvolíme medián ich súradníc. Keby sme totiž zvolili inú  $y$ -ovú súradnicu (posunutú o  $d$ ), aspoň polovica ľudí by šla o  $d$  viac a najviac polovica by šla najviac o  $d$  menej, čiže výsledný súčet by sa nezmenšil. A ako určíme  $x$ -ovú súradnicu najľavejšieho z nich? Využijeme fakt, že určite existuje optimálne riešenie, v ktorom najľavejší cvičenec ide na najľavejšie políčko, atď. až najpravejší na najpravejšie. Keď si teraz predstavíme, že by sme najľavejšieho cvičenca posunuli o  $0$ , ďalšieho o  $1$ , atď. až najpravejšieho o  $n - 1$ , majú sa zbehnúť všetci na rovnakú  $x$ -ovú súradnicu, čo bude zároveň hľadaná  $x$ -ová súradnica najľavejšieho z nich. Tú nájdeme ako medián takto upravených súradníc. Takéto riešenie má kvôli triedeniu zložitost'  $O(n \log n)$ . Šikovným použitím Count sortu a ďalších netriviálnych myšlienok sa dá dosiahnuť zložitost'  $O(n)$ .

**1714.** Z výmenných kurzov si vybudujeme graf, pričom medzi menou  $x$  a  $y$  bude viesť hrana s dĺžkou  $-\log k_{x,y}$ . Našou úlohou je teraz nájsť záporný cyklus v grafe. Toto môžeme urobiť známym Floyd-Warshallovým algoritmom v čase  $O(n^3)$ .

**1715.** Najskôr si uvedomme, že nám stačí uvažovať súperove ťahy dopredu. Ak totiž niekedy potiahne dozadu, my potiahneme o toľko isto dopredu. Toto môže spraviť počas hry len konečne veľa krát. Uvažujme teraz hru NIM, v ktorej máme  $n$  kôpok a na  $i$ -tej kôpke je toľko zápaliek, koľko je voľných miest medzi panáčikmi v  $i$ -tom stĺpci. V jednom ťahu môžeme zobrať z jednej kôpky ľubovoľne veľa, kto zoberie poslednú zápalku, vyhráva. Tieto dve hry sú zjavne ekvivalentné a o tomto NIME sa dá ukázať, že prehrávajúce pozície sú práve tie, kde bitový xor počtov zápaliek na kôpkach je  $0$ . (Napríklad použitím Grundyho čísel alebo ukázaním, že z každej prehrávajúcej musíme ťahať do vyhrávajúcej a z každej vyhrávajúcej vieme ťahať do prehrávajúcej.) S využitím tejto myšlienky už vieme ľahko zistiť, či je zadaná pozícia vyhrávajúca a ak áno, nájsť ťah vedúci do prehrávajúcej.

**1721.** Za mesto, v ktorom sa majú stretnúť, vždy môžeme vyhlásiť to, do ktorého vchádza najviac teleportov. (Skúste si toto tvrdenie dokázať, napríklad sporom.) Samotný program je priamočiary.

**1722.** Pozri úlohu 223.

**1723.** Vrcholy do ktorých nič nevedie označme ako hlavy (je ich  $h$ ) a vrcholy z ktorých nič nevedie označme ako chvosty (je ich  $c$ ). Začneme z nejakej hlavy a ideme po zjazdovkách kým nevojdeme do nejakého chvostu. Tento chvost spojíme lanovkou s inou hlavou a pokračujeme v hľadaní z nej. Keď sme pospájali všetky hlavy a chvosty spojíme posledný chvost s prvou hlavou. V prípade že  $h \neq c$  tak zvyšné hlavy/chvosty spojíme s ľubovoľnými vrcholmi na už vytvorenom cykle. Takto dosiahneme minimálny možný počet lanoviek, čo je  $\max(h, c)$ . Vzorový program má časovú zložitost'  $O(m + n)$ .

**1724.** Aby bol pás najužší, musí jedna jeho strana obsahovať celú hranu  $n$ -uholníka a druhá aspoň jeden bod (skúste si dokázať). Na riešenie zo zložitost'ou  $O(n^2)$  stačí ku každej strane nájsť najvzdialenejší vrchol a potom vybrať z týchto vzdialeností najmenšiu. Existuje aj lineárne riešenie založené na myšlienke: Ak vrchol  $V_{m_1}$  je najvzdialenejší od strany  $V_0V_1$  a  $V_{m_2}$  je najvzdialenejší od strany  $V_1V_2$ , potom  $m_1 \geq m_2$ . Stačí teda nájsť najvzdialenejší bod od prvej hrany. Najvzdialenejší bod od ďalšej hrany potom stačí hľadať už len od tohto bodu. Najvzdialenejší bod hľadáme tak, že ideme, kým sa vzdialenosť od hrany zväčšuje. V okamihu, keď sa raz zmenší, tak predchádzajúci bod bol najvzdialenejší.

**1725.** Nech sú kamene na pozíciách  $a, b, c, d$ . Prehrávajúca pozícia je taká, kde  $b - a = d - c$ . Ak vyhrávame tak posunieme ten kameň, aby táto rovnosť platila. Program má konštantnú časovú zložitost'.

**1731.** Vstup si budeme reprezentovať ako orientovaný graf. Z vrcholu  $A$  do vrcholu  $B$  vedie hrana o hodnote  $k$  ak KSPÁk  $A$  dlhuje  $B$   $k$  kofôl. Pre všetky vrcholy  $A, B, C$  také že  $A \rightarrow B$ ,  $B \rightarrow C$  vykonáme takúto operáciu: od hrán  $A \rightarrow B$  a  $B \rightarrow C$  odpočítame  $\min(k_1, k_2)$  a pripočítame ho k  $A \rightarrow C$ . Celková zložitost' je  $O(n^2)$ .

**1732.** Ako posledný kúpime najdrahší lístok. Zoberme tabuľku  $D_{1,s}$ , kde  $D_i$  určuje na koľko stlačením dosiahneme sumu  $i$ . Na začiatku položíme  $D_0 = 0$ ,  $D_i = \infty$  pre všetky ostatné  $i$ . Hodnotu  $D_{j+1}$  vypočítame takto:  $D_{j+1} = \min_i (D_{j+1 - c \cdot n \cdot a_i} + 1)$ . Časová zložitosť je  $O(ns)$

**1733.** Vyberieme niektorú skrutku, a rozdelíme zvyšné matice  $i$  skrutky na menšie, väčšie a rovnaké. Na tieto dve kopy (väčšiu a menšiu) použijeme ten istý algoritmus, až kým neprídeme k triviálnemu prípadu.  $O(n \log n)$  operácií.

**1734.** Ak je tyčí viac ako 45 tak sa trojuholník určite dá nájsť (kvôli obmedzeniu na dĺžku tyče). Ak je ich menej, stačí ich utriediť a skontrolovať, či sa dá trojuholník zostrojiť z troch, ktoré nasledujú po sebe. Takýto algoritmus má zložitosť  $O(1)$  ak existuje obmedzenie a  $O(n \log n)$  ak neexistuje.

**1735.** a) Najprv vypijeme najväčšiu fľašu. Ofarbíme fľaše striedavo na čierne a biele. Teraz máme na výber medzi čiernou a bielou. Rozhodneme sa, či budeme piť čierne alebo biele fľaše a toho sa držíme až do konca hry. b) Označme si objemy fliaš ako  $f_1, f_2, \dots, f_n$  a  $B_{i,j}$  najväčšie možné množstvo nápoja aké sme schopný vypiť ak sme na ťahu  $a$  a na stole sú fľaše  $f_i, f_{i+1}, \dots, f_j$ .  $C_{i,j}$  bude označovať fľašu, ktorú máme vypiť. Zjavne  $B_{i,i} = f_i$ . Ďalej platí  $B_{i,j} = \sum_{k=i}^j f_k - \min(B_{i+1,j}, B_{i,j-1})$  a do  $C_{i,j}$  zapíšeme tú možnosť, kde súper vypije menej. Časová zložitosť je  $O(n^2)$

**1741.** Hľadané riešenie je rovné veľkosti minimálneho rezu. Ten je ale rovný veľkosti maximálneho toku. Stačí teda nájsť maximálny tok napríklad pomocou Ford-Fulkersonovho algoritmu (pozri úlohu 1424). Ako potom nájdeme hrany patriace do min rezu? Vyškrtáme z grafu všetky plné hrany t.j. také, ktoré sú pri maximálnom toku úplne vyťažené. Graf sa rozpadne na niekoľko častí. Všetky plné hrany (z tých vyškrtnutých) vedúce z vrcholov časti grafu obsahujúcej zdroj tvoria minimálny rez.

**1742.** Segmenty mora reprezentujeme grafom, ak sa ponorka môže pohnúť z jedného segmentu do druhého, sú spojené hranou. Na tomto grafe už iba pustíme prehľadávanie do šírky alebo do hĺbky z každého vrcholu nad hladinou (treba doplniť graf o výšku ponorky nad hladinu). Pri prehľadávaní nám ešte stačí pamätať si iba dva riadky (budeme ale potrebovať aspoň na výšku ponorky). Ako vybudovať graf? Ponorka sa dá pamätať ako začiatok a dĺžka neprázdnnej časti pre každý riadok. V mape si môžeme pre každé políčko predvypočítať počet voľných políčok napravo od neho. Potom pre každé políčko, zistenie či ľavý horný roh ponorky môže ležať na ňom ide v čase  $O(b)$  -pre každý riadok ponorky stačí pozrieť či napravo od segmentu  $i + \text{zac}_i$  je aspoň  $\text{dl}_i$  voľných políčok.

**1743.** Ide o netradičnú formuláciu geometrickej úlohy. Nech sa  $i$ -ty pretekár umiestnil na  $u_{i,1}$  prvých,  $u_{i,2}$  druhých a  $u_{i,3}$  tretích miestach. Potom musí platiť:

$$u_{i,1} * p_1 + u_{i,2} * p_2 + u_{i,3} * p_3 < u_{1,1} * p_1 + u_{1,2} * p_2 + u_{1,3} * p_3$$

Ďalej  $p_3 \geq p_2 \geq p_1 \geq 1$ . Dostávame sústavu niekoľkých lineárnych nerovníc o troch neznámych. Treba si uvedomiť, že  $p_1$  môžeme pevne zvoliť, napríklad 1. Riešiť sústavu lineárnych nerovníc o dvoch neznámych už nie je také ťažké. Treba postupne konštruovať prieniky polrovín, čo je v každom kroku konvexný „mnohouholník“ (niekedy nekonečný).

**1744.** Po predpocítaní čiastočných súčtov,  $s_i$  = súčet teplôt v prvých  $i$  dňoch, je jednoduché urobiť riešenie v čase  $O(n^2)$ . Platí ale, že existuje obdobie, ktorého dĺžka je menšia ako  $2k$  dní, a pritom dosahuje maximálny možný priemer. Tým vieme zložitosť zlepšiť na  $O(nk)$ . Existuje ale aj lineárne riešenie. Predstavíme si čiastočné súčty ako body  $S_i = (s_i, i)$  v rovine. Priemerná teplota obdobia  $a + 1$  až  $b$  je teda smernica priamky  $S_a S_b$ . Ako nájsť priamku s najväčšou smernicou? Načrtujeme len ideu. Budeme prechádzať zaradom po bodoch  $S_i$ , postupne si budeme vytvárať spodný konvexný obal bodov  $S_1 \dots S_i$ . V ňom existuje „zlomový vrchol“  $D_d$ , to je prvý taký bod, bod pre ktorý je uhol priamky  $B_d B_{d+1}$  väčší ako uhol zodpovedajúci doteraz najteplejšiemu obdobiu. Je zrejme, že väčší uhol

môžu zvierať iba body napravo od tohto vrcholu. Pri správnej implementácii jednotlivých úloh sa dá dosiahnuť zložitosť  $O(n)$ .

**1745.** Stačí dokázať, že prehrávajúce pozície sú tie, ktorých číslo mesta je deliteľné 4. Vyhrávajúca stratégia pre vyhrávajúceho hráča je ťahať vždy tak, aby protivník skončil v pozícii deliteľnej 4.

**z1711.** Najprv vypočítame posun  $p$  do 1. januára daného roku  $r$  oproti nejakému známemu roku  $r_1$ .  $P = (r - r_1) + (r - r_1) \bmod 4 - (r - r_1) \bmod 100 + (r - r_1) \bmod 400$ . Teraz už len pripočítame posun za počet dní od 1. januára do prvého dňa mesiaca (upravíme pre prípadný priestupný rok) a vypíšeme predpočítaný počet pracovných dní podľa dĺžky mesiaca a jeho prvého dňa.

**z1712.** Pozri úlohu 123.

**z1713.** Všimnime si, že keď hľadáme riešenie, pre  $n$  remeselníkov, dá sa zložiť z dvoch riešení pre  $n - 1$  remeselníkov. Najskôr vypíšeme kombináciu pre  $n - 1$ , pridáme  $n$ -tého a znovu vypíšeme pre  $n - 1$ , ale tentoraz v obrátenom poradí. To spravíme pomocou rekurzcie aj bez toho, aby sa do poľa zapisovali všetky kombinácie. Stačí nám pole, kde si pamätáme, koho máme v dome. Dá sa spraviť iné riešenie založené na zmene správneho remeselníka v správnom čase. Totiž  $i$ -ty remeselník sa strieda každý  $2^i$ -ty krok, vieme teda priamo z čísla kroku kto má prísť/odísť.

**z1714.** Tu vpodstate nie je čo dodať, algoritmus bol zadaný. Snáď len, že si treba dať pozor, kam má byť Jones nasmerovaný na začiatku a kedy skončí (nielen, keď dojde k východu!). Smer sa dá určiť aj bez rozoberania pozície Jonesa, na základe konštanty.

**z1715.** Za predpokladu, že je text korektný, môžeme príkazy HTML a podobné ignorovať. V cykle postupne načítavame vstupný súbor po slovách (slovo je aj HTML príkaz), pričom preskakujeme medzery (pozor, aby sme nestratili prvé písmená slov), ktoré si zapisujeme do pomocného reťazca. Po prekročení šírky stránky sa reťazec vypíše (či už centrovane, alebo nie). Pri načítaní príkazu P alebo CENTER, vypíšeme obsah reťazca a urobíme, čo nám príkaz káže. Pozor, príkazov CENTER môže byť aj viac vnorených.

**z1721.** Môžeme predpokladať, že existuje aspoň jedna cesta z knižnice (mesto 1) do hlavného stanu (mesto  $n$ ), lebo ak by nebola žiadna, určite by neexistovala ani najkratšia. Vezmeme teda všetky cesty z mesta 1 do mesta  $n$ , ktoré neobsahujú cykly (t.j. cez žiadny významný bod neprechádzajú dvakrát). Takýchto ciest je konečne veľa, a preto (ako hovorí známa veta), existuje medzi nimi najkratšia. Tým dostávame princíp vzorového riešenia – stačí overiť, či existuje nejaká cesta z mesta 1 do mesta  $n$ . Preto nás nemusia zaujímať ani dĺžky ciest a stačí použiť prehľadávanie do šírky či hĺbky so zložitou  $O(m + n)$ .

**z1722.** Aby sme nemuseli opakovane počítat mravcov na jednotlivých poschodiach (čím by sme dostali algoritmus so zložitou  $O(kn)$ ), použijeme pole  $A$ , pre ktoré bude platiť  $A[i] =$  celkový počet mravcov na poschodiach 1 až  $n$  (a navyše  $A[0] = 0$ ). Ak budeme potrebovať zrátať mravcov na poschodiach  $i$  až  $j$ , stačí nám potom vziať rozdiel  $A[j] - A[i - 1]$ . Takto dokážeme na jednu otázku odpovedať v čase  $O(1)$ , čiže celková zložitosť bude  $O(n + k)$ . Keďže pole  $A$  vieme vytvoriť už počas načítavania (lebo platí  $A[i + 1] = A[i] + z$ , kde  $z$  je počet mravcov na  $(i + 1)$ -om poschodí), použijeme skutočne len jedno pole veľkosti  $O(n)$ .

**z1723.** Zadanie úlohy priamo napovedá, že riešenie bude rekurzívne (dá sa použiť simulácia zásobníku namiesto rekurzcie). Keď potrebujeme vyhodnotiť výraz, sú dve možnosti – buď je to číslo, a vtedy sme skončili, alebo je to operátor. V takom prípade jednoducho dáme jednoducho vyhodnotiť argumenty (rekurzívne zavoláme vyhodnocovanie výrazu na zvyšok reťazca) a danú operáciu zrealizujeme. Časové aj pamäťové nároky sú  $O(n)$ .

**z1724.** Na simuláciu včiel je najjednoduchšie použiť zoznam ich polôh, v ktorom navyše na koniec pridáme polohu prvej včely, aby sme ju po presune nestratili a robili presuny správne. V jednom kroku vždy jednoducho prejdeme zoznam a včely posunieme. Priamou

implementáciou tohto postupu dostávame vzorové riešenie. Prémiová úloha – stačí si uvedomiť, že včely sú po každom kroku v rohoch štvorca, ktorý sa bude postupne zmenšovať a otáčať. Včely teda pôjdu po špirálach smerom do stredu.

**z1725.** Úlohou je napísať čo najefektívnejší triediaci algoritmus. Medzi najpomalšie patria Bubble-, Max- a Insert- Sort (so zložitostou  $O(n^2)$ ). Lepšie riešenie sa dosiahne použitím Dobosiewicz- alebo Shell- Sortu ( $O(n^{3/2})$ ). Najlepšie výsledky podáva Quick-, Heap- Sort ( $O(n \log n)$ ). Všetky tieto triedenia možno nájsť v literatúre. Týmito algoritmami sa zaoberá veľa kníh, informácie nájdete napríklad v [23] alebo [37]

**z1731.** Vstup načítame do grafu, ktorého vrcholmi sú ľudia a hranami sú vzťahy dieťa, rodič, manžel, manželka. Do grafu musíme doplniť všetky chýbajúce hrany, t.j. každému dieťaťu doplniť vzťah k druhému rodičovi – manželovi alebo manželke prvého rodiča. Najbližšiemu vzťahu zodpovedá najkratšia cesta v grafe, posluží prehľadávanie grafu do šírky.

**z1732.** Spočítame si uhol, pod akým vidíme prvý kilometer z Mt. Kopca a označíme ho ako potenciálnu koncovú zastávku. Postúpime na ďalší kilometer. Ak z neho kopec nevidíme (uhol je menší ako maximum za prejdenu časť), ideme ďalej. Ak kopec vidíme, zapamätáme si nové maximum uhla a skúsime či do tohoto kilometra nebude lanovka dlhšia.

**z1733.** Prehľadávaním (do šírky/hĺbky) vždy označíme jeden ostrov. A popri označovaní môžeme zároveň spočítavať kopce, uchováme si súradnice políčka, z ktorého sme označili ostrov s najviac kopcami.

**z1734.** Nealgoritmická, ale interaktívna úloha určená na rozvíjanie zručnosti.

**z1735.** Vytvoríme graf. Vrcholy budú HTML súbory. Orientovaná hrana vedie z A do B, ak v A je odvolávka na B. Prehľadáme graf z vrcholu zodpovedajúcom súboru „index.html“. Tie vrcholy (súbory), ktoré sme nanavštívili môžeme odmazáť. Na rýchle priradovanie názvu súboru k číslu vrcholu posluží písmenkový strom.

## Literatúra

- [1] Aho V. Alfred, Hopcroft E. John, Ullman D. Jeffrey, The Design and Analysis of Computer Algorithms, Addison-Wesley 1976. Ruský preklad: Ахо А., Хопкрофт Дж., Ульман Дж., Построение и анализ вычислительных алгоритмов, МИР, Москва 1979.
- [2] Barker Felix, Ross-Macdonald Malcolm, Castlereagh Duncan, The Search Begins, Aldus Books and Jupiter Books, London 1973. Slovenský preklad: Začiatky hľadania, Mladé letá, Bratislava, 1981.
- [3] Bentley Jon, Programming Pearls, Addison-Wesley, 1986. Slovenský preklad: Perly programovania, Alfa, Bratislava 1992.
- [4] de Berg Mark, van Kreveld Marc, Overmars Mark, Schwarzkopf Otfried, Computational Geometry, Algorithms and Applications, Springer-Verlag Berlin Heidelberg 1997.
- [5] Бородин А. И, Бугай А. С. Выдающиеся математики, Киев, Радянська школа, 1987.
- [6] Cormen H. Thomas, Leiserson E. Charles, Rivest L. Ronald, Introduction to Algorithms, The MIT Press, Cambridge, Massachusetts, McGraw-Hill 1990.
- [7] Dijkstra W. Edsger, A Discipline of Programming, Prentice-Hall, Inc., 1976. Ruský preklad: Э. Дейкстра, Дисциплина программирования, МИР, Москва, 1978.
- [8] Dijkstra W. Edsger, Feijen W. H. J., A Method of Programming, Addison-Wesley, Inc., 1988.
- [9] Льюдни А. К., Занимательный компьютер, *В Муре Науки*, 6/1985, s. 90–95. Pôvodne v *Scientific American* 4/1985.
- [10] Gonnet H. Gaston, Baeza-Yates Ricardo, Handbook of Algorithms and Data Structures, In Pascal and C, Addison-Wesley 1991.
- [11] Giblin Peter, Primes and Programming, An Introduction to Number Theory with Computing, Cambridge University Press, 1993.
- [12] Graham R. Lewis, Knuth E. Donald, Patashnik Oren, Concrete mathematics: a foundation for computer science, Addison-Wesley, 1994.
- [13] Granát Luděk, Sechovský Hynek, Počítačová grafika, KVT, SNTL, Praha, 1980.
- [14] Granát Luděk, Počítačová grafika – zobrazování úsečky, *Rozhledy matematicko-fyzikální* 190–196, č. 5, roč. 66, 87/88.
- [15] Granát Luděk, Počítačová grafika – ořezávání, *Rozhledy matematicko-fyzikální* 270–274, č. 7, roč. 66, 87/88.
- [16] Gries David, The Science of Programming, Springer Verlag, 1981. Ruský preklad: Д. Грис, Наука программирования, МИР, Москва, 1984.
- [17] Gruska Jozef, Foundation of Computing, International Thomson Computer Press, 1997.
- [18] Hopcroft E. John, Ullman D. Jeffrey, Formálne jazyky a automaty, Alfa, Bratislava, 1978.
- [19] Hvorecký Jozef, Hra s algoritmom, *Matematické Obzory* 31, 1–11, 1989.
- [20] Kemp Reiner, Fundamentals of the Average Case Analysis of Particular Algorithms, (B. G. Teubner) John Wiley & Sons
- [21] Knuth E. Donald, The art of computer programming, Volume 1: Fundamental Algorithms, Addison-Wesley 1968. Ruský preklad: Д. Кнут, Искусство программирования для ЭВМ, т. 1, Основные алгоритмы, МИР, Москва 1976.
- [22] Knuth E. Donald, The art of computer programming, Volume 2: Seminumerical Algorithms, Addison-Wesley 1981.

- [23] Knuth E. Donald, The art of computer programming, Volume 3: Sorting and Searching, Addison-Wesley 1973. Ruský preklad: Д. Кнут, Искусство программирования для ЭВМ, т. 3, Сортировка и поиск, МИР, Москва 1978.
- [24] Knuth E. Donald, The Stanford GeaphBase : A platform fro Combinatorial Computing, Addison-Wesley 1993.
- [25] Kučera Luděk, Kombinatorické Algoritmy, MS 18, SNTL, Praha, 1983.
- [26] Larson C. Loren, Metódy riešenia matematických problémov, Alfa, Bratislava, 1990.
- [27] Lipski Witold, Kombinatoryka dla programistów, Wydawnictwa Naukowo-Techniczne, Warszawa, 1982. Ruský preklad: В. Липский, Комбинаторика для программистов, МИР, Москва, 1988
- [28] Machačka Ivo, Paulů Jan, Programování v jazyku BASIC, SNTL, Praha 1986.
- [29] Maličský Peter, Zátvorkový problém, *Matematické Obzory* 33, 61–67, 1989.
- [30] Manber Udi, Introduction to Algorithms (a creative approach), Addison-Wesley, 1989.
- [31] Melhorn Kurt, Data Structures and Algorithms 3: Multi-dimensional Searching and Computational Geometry, Springer-Verlag, 1984.
- [32] Molnár Ludovít, Programovanie v jazyku Pascal, Alfa, Bratislava, 1987.
- [33] Оре О. (Ore Øystein), Приглашение в теорию чисел, Библиотечка Квант, выпуск 3, Наука, 1980.
- [34] Plesník Ján, Grafové Algoritmy, VEDA, 1983.
- [35] Preparata P. Franco, Shamos I. Michael, Computational Geometry (an Introduction), Text & Monographs in CS, Springer Verlag, 1985.
- [36] Riečan Beloslav, Príbehy o integráloch, SPN, 1988
- [37] Sedgewick Robert, Algorithms, druhé vydanie, Addison-Wesley 1988.
- [38] Sedgewick Robert, Flajolet Philippe, An introduction to the analysis of algorithms, Addison-Wesley 1996.
- [39] Suri Subhash, A Linear Time Algorithm for Minimum Link Paths inside a Simple Polygon, Computer Vision, Graphics, and Image procesing 35, 99–110 (1986).
- [40] Vít Pavel, Řetězové zlomky, Škola mladých matematiků, sv. 49, Praha, 1982.
- [41] Wiederman Jiří, Vyhledávání, MS 27, SNTL, Praha, 1991.
- [42] Wirth Niklaus, Algorithms + Data Structures = Programs, Prentice-Hall, Inc., 1975. Slovenský preklad: Algoritmy a štruktúry údajov, Alfa, Bratislava 1987.
- [43] Wirth Niklaus, Algorithms and data structures, Prentice-Hall, Inc., 1986. Ruský preklad: Н. Вирт, Алгоритмы и структуры данных, МИР, Москва 1989.





Michal Winczer a kolektív

**Zbierka úloh Korešpondenčného seminára z programovania (1983-2001)**

Vydala Fakulta Matematiky, fyziky a informatiky Univerzity Komenského,  
Mlynská Dolina, 842 48 Bratislava v júli 2001

Návrh obálky Dušan Bezák, Miroslav Dudík, Dano Štefankovič a Tomáš Vinař

Sadzbú programom  $\TeX$  vytvorili Rišo Nemec, Tomáš Vinař, Vlado Koutný a Michal Winczer

Konečné spracovanie dokumentu Dušan Bezák, Ivona Bezáková, Bronislava Brejová, Martin Pál, Vlado Koutný

Vytlačila OKAT PLUS, spol. s r.o. Bratislava

248 strán, 3. rozšírené vydanie, náklad 200 výtlačkov

Neprešlo jazykovou úpravou

