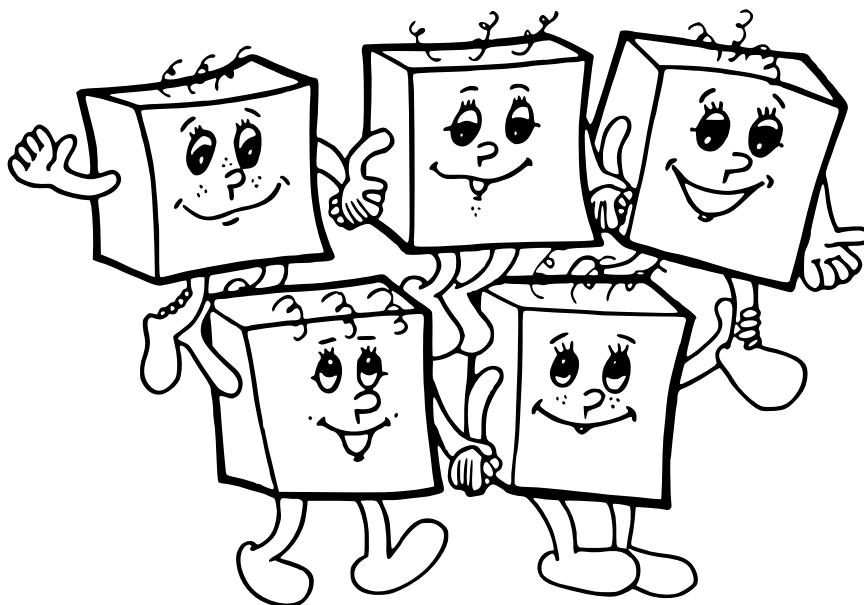


OLYMPIÁDA V INFORMATIKE NA STREDNÝCH ŠKOLÁCH



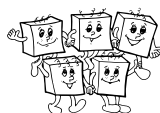
dvadsiaty šiesty ročník
školský rok 2010/11

zadania domáceho kola
kategórie A a B

- **Olympiáda v informatike** je od školského roku 2006/07 samostatnou súťažou. Predchádzajúcich 21 ročníkov tejto súťaže prebiehalo pod názvom **Matematická olympiáda, kategória P** (programovanie).
- Oficiálnu **webstránku** súťaže nájdete na <http://oi.sk/>.

NOVINKY v tomto ročníku:

- Riešenia domáceho kola sa odovzdávajú len v elektronickej podobe.
- Praktické úlohy kategórie B je možné riešiť v ľubovoľnom programovacom jazyku.



Informácie a pravidlá

Pre koho je súťaž určená?

Kategória B má dve kolá: domáce a krajské.

Do kategórie B sa smú zapojiť len tí žiaci základných a stredných škôl, ktorí ešte ani v tomto, ani v nasledujúcom školskom roku nebudú končiť strednú školu.

Kategória A má tri kolá: domáce, krajské a celoštátne.

Do kategórie A sa môžu zapojiť všetci žiaci (základných aj) stredných škôl.

Najlepší riešitelia kategórie A majú šancu reprezentovať Slovensko na medzinárodných súťažiach.

Priebeh súťaže

Za každú úlohu domáceho kola sa dá získať od 0 do 10 bodov. Na základe bodov domáceho kola stanoví Slovenská komisia OI (SK OI) pre každú kategóriu bodovú hranicu potrebnú na postup do **krajského kola**. Očakávame, že táto hranica bude približne rovná **tretine maximálneho počtu bodov**.

V krajskom kole riešitelia riešia štyri teoretické úlohy, ktoré môžu tematicky nadväzovať na úlohy domáceho kola. V kategórii B súťaž týmto kolom končí. V kategórii A prebehne po opravení riešení koordinácia bodovacích škál, spoja sa výsledkové listiny do jednej celoštátnej a podľa nej je približne najlepších 30 riešiteľov pozvaných do **celoštátneho kola**.

V celoštátnom kole účastníci prvý deň riešia tri teoretické úlohy, druhý deň dve praktické úlohy. Najlepší riešitelia sú vyhlásení za víťazov. Približne desať najlepších riešiteľov následne SK OI pozve na týždňové výberové sústreďenie. Podľa jeho výsledkov SK OI vyberie družstvá pre Medzinárodnú olympiádu v informatike (IOI) a Stredoeurópsku olympiádu v informatike (CEOI).

Odvzdávanie riešení domáceho kola

Všetky úlohy je **nutné odovzdať prostredníctvom webového rozhrania** na stránkach olympiády (<http://oi.sk/>) najneskôr v deň uvedený v zadaniach príslušnej kategórie.

Ako majú vyzeráť riešenia úloh?

V praktických úlohách je vašou úlohou vytvoriť program, ktorý bude riešiť zadanú úlohu. Program musí byť v prvom rade korektný a funkčný, v druhom rade sa snažte aby bol čo najefektívnejší.

V kategórii B môžete použiť ľubovoľný programovací jazyk.

V kategórii A musíte písať riešenia v jazyku Pascal, C, alebo C++, Odovzdaný program bude automaticky otestovaný na viacerých vopred pripravených testovacích vstupoch. Podľa toho, na koľko z nich dá správnu odpoveď, vám budú pridelené body. Výsledok testovania sa dozviete krátko po odovzdaní. Ak váš program nezíska plný počet bodov, budete ho môcť vylepšiť a odovzdať znova, až do uplynutia termínu na odovzdávanie.

Presný popis, ako majú vyzeráť riešenia praktických úloh (napr. realizáciu vstupu a výstupu), nájdete na webstránke, kde ich budete odovzdávať.

Ak nie je v zadaní povedané ináč, riešenia teoretických úloh musia v prvom rade obsahovať **podrobný slovný popis použitého algoritmu, zdôvodnenie jeho správnosti** a diskusiu o efektívite zvoleného riešenia (t. j. posúdenie časových a pamäťových nárokov programu). Na záver riešenia uveďte program napísaný v jazyku Pascal, C alebo C++. Ak používate v programe netriviálne algoritmy alebo dátové štruktúry (napr. rôzne súčasti STL v C++), súčasťou popisu algoritmu musí byť dostatočný popis ich implementácie.

Usporiadateľ súťaže

Olympiádu v informatike (OI) vyhlasuje *Ministerstvo školstva SR* v spolupráci so *Slovenskou informatickou spoločnosťou* a *Slovenskou komisiou Olympiády v informatike*. Súťaž organizuje *Slovenská komisia OI* a v jednotlivých krajoch ju riadia *krajské komisie OI*. Na jednotlivých školách ju zaisťujú učitelia informatiky. Celoštátne kolo OI, tlač materiálov a ich distribúciu po organizačnej stránke zabezpečuje *IUVENTA* v tesnej súčinnosti so Slovenskou komisiou OI.



Zadania kategórie B

Túto kategóriu môžu riešiť len žiaci, ktorí ani v tomto, ani v nasledujúcom školskom roku nematurujú. Jej riešenia je potrebné odovzdať na stránke <http://oi.sk/> najneskôr **29. 11. 2010**.

B-I-1 Hľadanie mín

Hru „Hľadanie mín“ asi nemusíme nijak špeciálne predstavovať. Na niektorých políčkach obdĺžnikového hracieho plánu sa nachádzajú míny. Úlohou hráča je postupne odkryť všetky políčka, ktoré neobsahujú mínu. Aby hráč nemusel len náhodne tipovať, hra mu pomáha: vždy, keď odkryje prázdne políčko, dozvie sa, s koľkými mínami toto políčko (stranou alebo rohom) susedí.



Súťažná úloha

Napište program, ktorý načíta rozmery hracieho plánu, počet mín a ich rozmiestnenie a vypočíta, ako by vyzerala úspešne ukončená hra na danom pláne.

Formát vstupu

V prvom riadku každého vstupného súboru sú dve celé čísla: počet riadkov R a počet stĺpcov S hracieho plánu. Riadky sú očíslované zhora dole od 1 po R , stĺpce zľava doprava od 1 po S .

V druhom riadku je jedno celé číslo N udávajúce počet mín. Nasleduje N riadkov. V i -tom z nich sú dve celé čísla r_i a s_i , udávajúce riadok a stĺpec jedného z políčok, ktoré obsahujú mínu.

Formát výstupu

Váš program by mal vyrobiť textový súbor, v ktorom bude R riadkov a v každom z nich S znakov. Každý znak predstavuje jedno políčko hracieho plánu. Ak je na danom políčku mína, vypíšte znak '*' (hviezdička). Ak dané prázdne políčko susedí s 1 až 8 mínami, vypíšte príslušnú cifru. Ak dané prázdne políčko nesusedí so žiadnou mínou, vypíšte znak '.' (bodka).

Príklad

vstup

```
9 9
10
1 8
1 9
3 7
6 4
7 7
7 8
8 1
8 2
8 4
9 2
```

výstup

```
.....1**
.....1232
.....1*1.
.....111.
..111....
..1*11221
223221**1
**3*11221
3*311....
```

Tento príklad vstupu a výstupu zodpovedá obrázku na vrchu zadania.

Odvzdávanie riešení

Toto je praktická úloha. Napište **v ľubovoľnom programovacom jazyku** program, ktorý ju rieši. Zo stránky <http://oi.sk/> stiahnite ZIP archív obsahujúci 5 testovacích vstupov, nazvaných 1.txt až 5.txt. Vytvorte k čo najviac vstupom správne výstupy a uložte ich do súborov sol1.txt až sol5.txt. Odovzdajte ZIP archív obsahujúci zdrojový kód vášho programu a tieto výstupné súbory. Za každý správny výstupný súbor získate 2 body.



B-I-2 Starosta

Móricka už omrzelo, že sú v jeho rodnej dedine chodníky plné odpadkov, obecný policajt je nezvestný, psy starej Košáričky terorizujú celú ulicu a navyše stále prší. Rozhodol sa, že bude kandidovať na starostu a všetko to zmení k lepšiemu.

Lenže byť na dedine starostom, to nie je len tak. Na to je najlepšie poznať všetkých dedinčanov. A keď náhodou starosta nejakého človeka Č nepozná, tak nutne musí poznať aspoň niekoho, kto človeka Č pozná. (Inými slovami, starosta musí mať s človekom Č spoločného známeho.)

Móricko si teraz nie je istý, či už môže za starostu kandidovať. Napíšte program, ktorý mu to zistí.

Súťažná úloha

Daný je zoznam všetkých dvojíc dedinčanov, ktorí sa navzájom poznajú. Zistite, či už Móricko môže byť starostom. Ak nie, nájdite všetkých ľudí, ktorých nepozná ani Móricko, ani nik, koho Móricko pozná.

Formát vstupu

V prvom riadku vstupného súboru sú dve celé čísla D a Z ($1 \leq D \leq 1\,000\,000$, $0 \leq Z \leq 5\,000\,000$). D je počet dedinčanov, Z je počet známostí. Dedinčanov si očísľujeme od 1 po D , pričom číslo 1 dostane Móricko.

Každý zo zvyšných Z riadkov obsahuje dve čísla od 1 po D – čísla dvoch dedinčanov, ktorí sa navzájom poznajú.

Formát výstupu

Ak Móricko môže byť starostom, vypíšte jeden riadok a v ňom reťazec „starosta“. Ak nie, vypíšte čísla všetkých dedinčanov, kvôli ktorým Móricko byť starostom nemôže. Čísla vypíšte v ľubovoľnom poradí, každé práve raz.

Príklady

vstup

```
4 3
1 3
1 4
4 2
```

výstup

```
starosta
```

Móricko pozná dedinčanov 3 a 4. Dedinčana 2 síce nepozná, ale pozná dedinčana 4 a ten pozná dedinčana 2. Takže je všetko v poriadku a Móricko môže byť starostom.

vstup

```
6 5
1 2
1 3
2 3
2 5
4 5
```

výstup

```
6
4
```

Móricko nemôže byť starostom. Dedinčanov 4 a 6 treba vypísať na výstup, keďže Móricko nepozná ani ich, ani žiadneho ich známeho.

Odvzdávanie riešení

Toto je praktická úloha. Napíšte v ľubovoľnom programovacom jazyku program, ktorý ju rieši.

Zo stránky <http://oi.sk/> stiahnite ZIP archív obsahujúci 5 testovacích vstupov, nazvaných 1.txt až 5.txt. Vytvorte k čo najviac vstupom správne výstupy a uložte ich do súborov sol1.txt až sol5.txt.

Odvzdajte ZIP archív obsahujúci zdrojový kód vášho programu a tieto výstupné súbory.

Za každý správny výstupný súbor získate 2 body.



B-I-3 Poriadok na polici

V knižnici Neviditeľnej univerzity je polica a na tej polici je šesť obrovských a strašne ťažkých zväzkov magickej encyklopédie. Tieto zväzky žijú vlastným životom a každú noc sa škodoradostne preusporiadajú.

Chudák knihovník ich potom musí každé ráno preusporiadať do správneho poradia. Keďže sú však strašne ťažké, jediné, čo zvláda, je vymeniť dva susedné zväzky. A navyše po každej takej výmene musí 10 minút oddychovať.

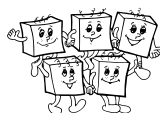
Včera napríklad boli zväzky ráno v poradí 1, 4, 2, 3, 5, 6. Keď ich chcel knihovník usporiadať, najskôr vymenil zväzky 4 a 2 (čím dostal poradie 1, 2, 4, 3, 5, 6), potom si oddýchol a potom vymenil zväzky 4 a 3 (čím dostal správne poradie 1, 2, 3, 4, 5, 6).

Súťažná úloha

- (3 body) Dnes sú zväzky v poradí 4, 2, 1, 6, 5, 3. Poradte knihovníkovi, ako ich najrýchlejšie usporiada.
- (3 body) Zdôvodnite, prečo je vaše riešenie podúlohy a) optimálne. Inými slovami, zdôvodnite, že neexistuje postup, ktorý by zväzky usporiadal pomocou menej výmen ako ten váš.
- (4 body) Budúci mesiac vyjde nové vydanie magickej encyklopédie. To už nebude mať 6 zväzkov, ale N .
Dokážte, že bez ohľadu na to, ako sa v noci zväzky prehádzu, knihovníkovi bude určite stačiť spraviť nanaajvyš $N(N - 1)/2$ výmen na to, aby ich usporiadal.

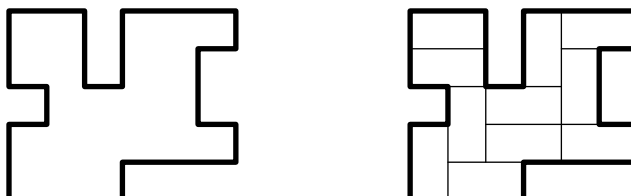
Odovzdávanie riešení

Toto je teoretická úloha. Pomocou webového rozhrania odovzdajte súbor vo formáte PDF, obsahujúci riešenie, spĺňajúce požiadavky uvedené v pravidlách.



B-I-4 Dláždenie

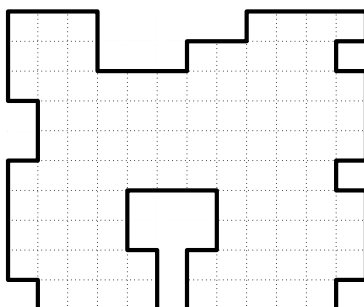
Korina je architektka. Poslednou dobou sa pustila do navrhovania moderných bytov. Keďže si kúpila štvorcový zošit, rozhodla sa, že každá miestnosť, ktorú navrhne, bude mať pôdorys tvorený niekoľkými jednotkovými štvorcami. Podlahu v miestnosti sa Korina rozhodla vydláždiť parketami rozmerov presne 2×1 štvorciek.



Na obrázku vľavo vidíte príklad jednej takejto miestnosti, na obrázku vpravo je jedno možné vydláždenie.

Súťažná úloha

- (2 body) Korina rýchlo zistila, že ak navrhne miestnosť tvorenú nepárnym počtom štvorcíkov, vydláždiť sa jej ju nepodari. Ukážte jej, že niekedy ani párný počet štvorcíkov nepomôže. Nájdite miestnosť tvorenú presne 10 štvorcíkmi, ktorá sa nebude dať vydláždiť.
- (3 body) Na nasledujúcom obrázku je ďalšia z miestností, ktoré Korina navrhla. Nech sa snaží, ako chce, nedarí sa jej ju vydláždiť. Pomôžte jej, alebo dokážte, že sa táto miestnosť vydláždiť nedá.



- (2 body) Pri niektorých miestnostiach má Korina na výber viacero spôsobov, ako ju vydláždiť. Navrhnite takú miestnosť, kde bude mať Korina na výber presne 3 možné dláždenia.
- (3 body) Navrhnite takú miestnosť, kde bude mať Korina na výber presne 16 možných dláždení.

Všetky vami navrhnuté miestnosti musia byť súvislé (t. j. „držať pokope“) a nesmú vo svojom vnútri obsahovať diery.

Odvzdávanie riešení

Toto je teoretická úloha. Pomocou webového rozhrania odovzdajte súbor vo formáte PDF, obsahujúci riešenie, spĺňajúce požiadavky uvedené v pravidlách.



Zadania kategórie A

Riešenia kategórie A je potrebné odovzdať na stránke <http://oi.sk/> najneskôr **15. 11. 2010**.

A-I-1 Indiana a poklad

Po dlhej a nebezpečnej ceste plnej dobrodružstiev sa pred Indianom konečne rozprestrel pohľad na starodávne pohrebisko aztéckych kráľov. Ležal tam hrob pri hrobe pozdĺž dlhej cesty. Všetky boli navlas rovnaké, líšili sa len výškou náhrobných kameňov. Pri prvom kameni bol nasledujúci nápis:

*Ak múdry si, hneď pochopíš, kde poklad môj má svoju skrýš.
Nepomôže však nič po tom čo zastaneš nad zlým hrobom!*

Výklad týchto veršov je jednoduchý: ak otvoríte hrob s pokladom, tak tento poklad získate; v opačnom prípade nezískate nič a pravdepodobne vás niečo rozmliaždi. „To nie je príliš lákavá perspektíva,“ pomyslel si Indiana. V tej chvíli si však spomenul na starý nápis, ktorý kedysi videl v aztéckom múzeu. A zrazu bolo Indianovi jasné, kde sa poklad nachádza.

Múdry je ten, kto volí z prostredných, ak nakoniec najväčší vyberie.

Súťažná úloha

Nech K je nepárne prirodzené číslo. *Medián* postupnosti dĺžky K je jej prostredný prvok podľa veľkosti. Teda keby sme danú postupnosť utriedili, medián by bol ten prvok, ktorý by skončil na pozícii číslo $(K + 1)/2$.

Daná je postupnosť N celých kladných čísel v_1, v_2, \dots, v_N a nepárne celé číslo K . Vašou úlohou je nájsť tú súvislú K -prvkovú podpostupnosť tejto postupnosti, ktorej medián je najväčší, a vypísať veľkosť tohto mediánu.

Napríklad pre postupnosť výšok hrobov $(4, 5, 1, 3, 2, 4)$ a $K = 3$ máme na výber štyri súvislé podpostupnosti: $(4, 5, 1)$, $(5, 1, 3)$, $(1, 3, 2)$ a $(3, 2, 4)$. Ich mediány sú 4, 3, 2 a 3. Hľadáme najväčší z nich, teda číslo 4.

Formát vstupu

Prvý riadok vstupu obsahuje dve celé čísla N a K – počet hrobov a dĺžku uvažovaných podpostupností. Platí $1 \leq K \leq N \leq 1\,000\,000$ a tiež $K \leq 10\,000$. Každý z nasledujúcich N riadkov obsahuje výšku jedného hrobu (v poradí v_1, v_2, \dots, v_N). Výšky sú prirodzené čísla menšie ako $1\,000\,000\,000$.

Vo vstupoch, za ktoré budú dokopy 4 body, bude $K < 100$.

Formát výstupu

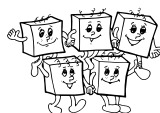
Vypíšte jeden riadok a v ňom jedno číslo, ktoré je rovné maximu všetkých mediánov súvislých úsekov dĺžky K v postupnosti čísel zadaných na vstupe.

Príklad

vstup	výstup
<pre>6 3 4 5 1 3 2 4</pre>	<pre>4</pre>

Odovzdávanie riešení

Toto je praktická úloha. Pomocou webového rozhrania odovzdajte **funkčný, odladený program**. Časový limit pre túto úlohu sú 4 sekundy, pamäťový limit je 256 MB.



A-I-2 Ešte raz poklad

Potom, čo Indiana našiel najväčší z prostredných hrobov, zistil, že nápis nebol úplne presný. Namiesto pokladu bol totiž pod kameňom len vchod do ďalšej chodby. Tá bola vydláždená štvorcovými dlaždicami a na stene chodby bol vyrytý tento nápis:

*Návštevník, čo nie je hlúpy,
práve raz na každú stúpi,
nie však ak je označená,
lebo lebka smrť znamená!*

Chodba mala na šírku presne 3 dlaždice a dlhá bola kam až oko dovidelo. Na niektorých dlaždiciach boli namalované lebky. (Pár z týchto dlaždíc zdobili aj skutočné lebky predchádzajúcich objaviteľov tajnej chodby.)

Indiana veľmi rýchlo nápis pochopil – musí na každú dlaždicu, okrem tých zakázaných, šliaپnúť práve raz, lebo len tak sa živý dostane k pokladu. V tom momente ale začal ľutovať, že má tak veľké nohy. Nech sa snažil, ako chcel, nikdy sa mu nepodarilo stúpiť len na jednu dlaždicu, ale vždy stúpil na dve susedné. To mu jeho úlohu samozrejme značne skomplikovalo.

Súťažná úloha

Daná je chodba dĺžky N , tvorí ju $3 \times N$ dlaždíc. Z týchto dlaždíc je K zakázaných, na tie Indiana nesmie stúpiť. Vašou úlohou je určiť, koľkými spôsobmi je možné pokryť túto chodbu stopami veľkosti 1×2 dlaždice tak, aby každá nezakázaná dlaždica bola pokrytá práve raz a žiadna zakázaná dlaždica nebola pokrytá. Keďže výsledné číslo môže byť veľmi veľké, stačí spočítať zvyšok, ktorý dáva po delení zadaným číslom L .

Formát vstupu

V prvom riadku sú zadané prirodzené čísla N , K a L . Môžete predpokladať, že platí $1 \leq N \leq 2\,500\,000$, $0 \leq K \leq \min(3N - 1, 1\,000\,000)$ a $1 \leq L \leq 1\,000\,000$.

Nasleduje K riadkov, v i -tom z nich je dvojica čísel r_i, s_i : súradnice i -tej zakázanej dlaždice. Pre každé i platí $1 \leq r_i \leq 3$ a $1 \leq s_i \leq N$ a navyše platí aj $s_1 \leq s_2 \leq \dots \leq s_K$.

V dvoch sadách testovacích vstupov bude platiť $N \leq 20$.

V ďalších dvoch sadách testovacích vstupov bude platiť, že počet dláždení neprekročí 500 000.

V ďalších dvoch sadách testovacích vstupov bude platiť $K = 0$.

Formát výstupu

Nech D je počet spôsobov ako chodbu pokryť. Vypíšte jediný riadok obsahujúci hodnotu $D \bmod L$ (t. j. zvyšok, ktorý dáva D po delení L). Všimnite si, že pre niektoré vstupy nemusí existovať žiadne riešenie, v takom prípade vypíšte nulu.

Príklad

vstup

```
5 3 13
3 1
1 2
2 4
```

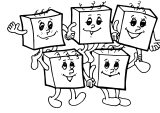
výstup

```
3
```

Pre tento vstup existujú presne 3 riešenia.

Odvzdávanie riešení

Toto je praktická úloha. Pomocou webového rozhrania odovzdajte **funkčný, odladený program**. Časový limit pre túto úlohu je 8 sekúnd, pamäťový limit je 256 MB.



A-I-3 Rieka

Za siedmimi horami sa nachádza rozsiahly prales, ktorým preteká dlhá rieka. Napriek svojej dĺžke nemá žiadne prítoky a jej tok sa nikde nerozvetvuje. V tomto pralese rastie veľa vzácnych druhov stromov, preto niet divu, že pri rieke ležia drevorubačské tábory. Vždy, keď drevorubači zotnú dosť stromov, postavia z nich plť a pošlú ju dole riekou.

Okrem drevorubačských táborov sa pri rieke nachádzajú aj píly. Zamestnanci každej píly číhajú pri rieke a keď zbadajú, že sa po nej pláva plť, vylovia ju a všetko drevo spotrebujú. Občas je píla zastavená kvôli údržbe, počas ktorej ignoruje plte a necháva ich plávať ďalej po rieke.

Zamestnancom píly prekáža, že nevedia, kedy očakávať dodávku dreva a tak zbytočne márnia čas pozorovaním rieky. Pomôžte im a napíšte program, ktorý im bude posilať upozornenia na blížiacu sa zásielku dreva.

Súťažná úloha

Rieka má dĺžku N kilometrov. Pozície bodov na rieke budú označené ich vzdialenosťou od prameňa. V každom z bodov $1, \dots, N$ sa nachádza buď drevorubačský tábor alebo píla. Umiestnenie píly na rieke je zadane na začiatku behu programu. Môžete predpokladať, že počet píly P je rádovo rovnaký ako dĺžka rieky N .

Váš program bude dostávať udalosti nasledujúcich typov: „píla v bode b začína údržbu“, „píla v bode b je opäť v prevádzke“ a „z tábora v bode b bola vyslaná plť“. Pre každú vyslanú plť váš program povie, v ktorej píle bude spracovaná. Môžete predpokladať, že rieka tečie naozaj rýchlo – medzi vyslaním plte a jej prijatím nestihnú v žiadnej píle spustiť ani ukončiť údržbu.

Poznámka: Existuje riešenie, ktoré na spracovanie každej udalosti potrebuje rádovo menej ako $\log N$ krokov.

Formát vstupu

Prvý riadok vstupu obsahuje tri prirodzené čísla N, P a M udávajúce dĺžku rieky, počet píly na nej a počet udalostí, ktoré nastanú ($1 \leq N \leq 100\,000, 1 \leq P \leq N$ a $1 \leq M \leq 1\,000\,000$). Druhý riadok obsahuje čísla n_1, \dots, n_P udávajúce, na ktorých kilometroch ležia píly. Môžete predpokladať, že $1 \leq n_1 \leq \dots \leq n_P \leq N$.

Zvyšných M riadkov popisuje udalosti v chronologickom poradí. Popis udalosti tvorí jedno z písmen Z, K a V (Začiatok údržby, Koniec údržby, Vyslaná plť), nasledované kilometrom, na ktorom táto udalosť nastala.

Formát výstupu

Pre každú vyslanú plť vypíšte jeden riadok a v ňom jedno celé číslo: vzdialenosť píly, ktorá spracuje túto plť, od prameňa rieky. Ak plť na rieke nezachytí žiadna píla, vypíšte 0.

Príklad

vstup

```
6 3 9
2 4 6
V 1
V 3
V 5
Z 2
V 1
K 2
V 1
Z 6
V 5
```

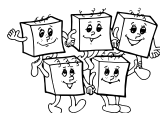
výstup

```
2
4
6
4
2
0
```

V poradí štvrtú plť zachytila až píla na 4. km, lebo na píle na 2. km vtedy prebiehala údržba. Posledná plť plávala len okolo poslednej píly a tá ju kvôli prebiehajúcej údržbe nezachytila.

Odovzdávanie riešení

Toto je teoretická úloha. Pomocou webového rozhrania odovzdajte súbor vo formáte PDF, obsahujúci riešenie, spĺňajúce požiadavky uvedené v pravidlách.

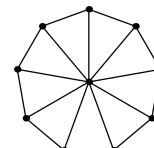


A-I-4 Grafový počítač

V tomto ročníku olympiády sa budeme v teoretickej úlohe stretávať so špeciálnym grafovým počítačom. V študijnom texte uvedenom za zadaním tejto úlohy je popísané, ako tento počítač funguje.

Súťažná úloha

- a) (5 bodov) Ruské kolo rádu n je graf, ktorý vznikne tak, že zoberieme kružnicu tvorenú n vrcholmi a pridáme jeden nový vrchol („stred kolesa“), ktorý bude spojený so všetkými n vrcholmi kružnice. Ruské kolo rádu n má teda $n + 1$ vrcholov a $2n$ hrán.



Napište funkciu pre grafový počítač, ktorá pre zadané n zostrojí ruské kolo rádu n . Na obrázku vpravo vidíte príklad pre $n = 9$.

- b) (5 bodov) Majme graf opisujúci cestnú sieť: vrcholy sú mestá, hrany cesty ohodnotené nezápornými vzdialenosťami v kilometroch. Cesty sú prepojené len v mestách, všetky ostatné križovatky sú mimoúrovňové. Zaujímá nás, akú najmenšiu vzdialenosť musíme prejsť, aby sme sa dostali z mesta x do mesta y . Inými slovami, máme v danom grafe nájsť cestu medzi x a y , pre ktorú bude celkový súčet váh použitých hrán najmenší možný.

Napište funkciu pre grafový počítač, ktorá dostane na vstupe graf cestnej siete a identifikátory dvoch rôznych vrcholov x , y a vráti vzdialenosť medzi týmito vrcholmi.

Odvzdávanie riešení

Toto je teoretická úloha. Pomocou webového rozhrania odovzdajte súbor vo formáte PDF, obsahujúci riešenie, spĺňajúce požiadavky uvedené v pravidlách.

Študijný text

Bežné počítače, ktoré máme všetci doma, všetko počítajú v číslach v dvojkovej sústave. Mnohým ľuďom to vyhovuje. Nie však železničným inžinierom v Tasmánii. Tí si jedného dňa uvedomili, že väčšina problémov, ktoré oni potrebujú riešiť, sa týka grafov. Preto si pre vlastnú potrebu navrhli *grafový počítač*, ktorý namiesto čísel pracuje priamo s grafmi. Vie s nimi robiť všetky bežné operácie, a to dokonca v konštantnom čase. V tomto ročníku Olympiády v informatike budete mať možnosť vyskúšať si, ako sa s takýmto počítačom pracuje.

Formálne, *graf* je usporiadaná dvojica (V, E) , kde V je konečná množina objektov, ktoré voláme *vrcholy*, a E je konečná množina obsahujúca niektoré neusporiadané dvojice vrcholov. Prvky E voláme *hrany*.

Neformálne si graf môžeme predstaviť ako železničnú sieť. Vrcholy sú jednotlivé zastávky, hrany sú priame úseky železničnej trate medzi dvojicami zastávok.

Upresnime ešte, že naše grafy nebudú nikdy obsahovať násobné hrany ani slučky. T. j., medzi každými dvomi vrcholmi bude viesť najviac jedna hrana a žiadna hrana nebude začínať a končiť v tom istom vrchole.

Grafy niekedy môžu byť *ohodnotené*: každému vrcholu, resp. každej hrane môžeme priradiť nejaké číslo. Ich význam môže byť v rôznych situáciách rôzny: raz to môže byť dĺžka koľajníc, inokedy cena cestovného lístka, a ešte inokedy môžeme pomocou čísel reprezentovať vlastnosti objektov: napríklad stanice, kde stoja rýchliky, môžu mať priradené číslo 1 a ostatné stanice číslo 2.

Reprezentácia grafu

Každý graf uložený v grafovom počítači má vrcholy očíslované prirodzenými číslami od 1 do ich počtu. Týmto číslam budeme hovoriť *identifikátory* vrcholov (skrátene: *id*).

Hranám identifikátory netreba, každá hrana je totiž jednoznačne určená pomocou *id* vrcholov, ktoré spája.

Každý vrchol má priradené svoje ohodnotenie, ktoré budeme volať *značka vrcholu*. Každá hrana má priradené svoje ohodnotenie, ktoré budeme volať *váha hrany*. Každé ohodnotenie je buď nezáporné celé číslo alebo špeciálna hodnota **undef** (nedefinované).



Programy budeme písať v bežnom programovacom jazyku, ktorý len rozšírime o niekoľko nových typov premenných a niekoľko nových funkcií. V tomto študijnom texte použijeme Pascal, ale pokojne môžete písať riešenia aj v iných jazykoch, do ktorých si nové súčasti doplníte analogicky.

Nové typy premenných a konštant

- Typ **Graph** slúži na uloženie grafu. Do premennej typu **Graph** teda môžeme uložiť jeden graf, je jedno ako veľký.
Premenné typu **Graph** vieme priradzovať a vieme testovať rovnosť obsahu dvoch takýchto premenných. Obe tieto operácie počítač vykoná v konštantnom čase.
- Konštanta **EmptyG** typu **Graph** obsahuje prázdny graf, teda graf s 0 vrcholmi (a teda nutne aj 0 hranami).
- Typ **Value** slúži na uloženie jedného ohodnotenia. Premenná typu **Value** teda môže nadobudnúť ako hodnotu ľubovoľné nezáporné celé číslo alebo špeciálnu hodnotu **undef**. Až na **undef** fungujú operácie s týmto typom rovnako ako operácie s bežnými celočíselnými typmi premenných.

Operácie meniace štruktúru grafu

- Funkcia **AddV(G,z)** pridá do grafu **G** nový vrchol a priradí mu značku **z**. Do nového vrcholu nevedú žiadne hrany a jeho id je rovné novému počtu vrcholov. Funkcia **AddV(G,z)** vráti toto id ako svoju návratovú hodnotu.
- Procedúra **DelV(G,id)** zmaže z grafu **G** vrchol s identifikátorom **id** a všetky hrany, ktoré do tohto vrcholu viedli. Následne poposúva id vrcholov tak, aby nevznikla diera v ich číslovaní. Teda z vrcholu s číslom **id+1** sa stane **id**, z **id+2** sa stane **id+1**, atď.
- Procedúra **AddE(G,x,y,w)** vytvorí v grafe **G** hranu medzi vrcholmi s id **x** a **y** a priradí jej ohodnotenie **w**.
- Procedúra **DelE(G,x,y)** odstráni z grafu **G** hranu medzi vrcholmi s id **x** a **y**.
- Funkcia **Test(G,x,y)** vráti **true** ak sú vrcholy s id **x** a **y** v grafe **G** spojené hranou a **false** ak nie sú.

Všetky tieto funkcie a procedúry bežia v konštantnom čase. Je nutné ich volať so správnymi parametrami, inak nastane chyba a program bude ukončený. Napr. procedúru **DelE** je povolené volať len s id vrcholov, ktoré skutočne sú v danom grafe spojené hranou.

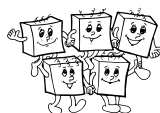
Operácie meniace značky vrcholov

- Funkcia **GetV(G,id)** vráti značku zadaného vrcholu.
- Procedúra **SetV(G,id,z)** nastaví značku zadaného vrcholu na zadanú hodnotu.
- Procedúra **SetAllV(G,z)** nastaví každému vrcholu v **G** značku na **z**.
- Procedúra **ReplaceV(G,zold,znew)** každému vrcholu v **G**, ktorý mal doteraz značku **zold**, zmení značku na **znew**.

Operácie meniace váhy hrán

Funkcia **GetE(G,x,y)** a procedúry **SetE(G,x,y,w)**, **SetAllE(G,w)** a **ReplaceE(G,wold,wnew)** sú definované rovnako ako funkcie pracujúce s vrcholmi. Hrana je popísaná id vrcholov **x** a **y**, ktoré spája.

Pre vaše pohodlie: Ak pri volaní **SetE(G,x,y,w)** hrana medzi **x** a **y** neexistuje, tak je do grafu pridaná a následne jej je priradená váha **w**.



Štatistické funkcie

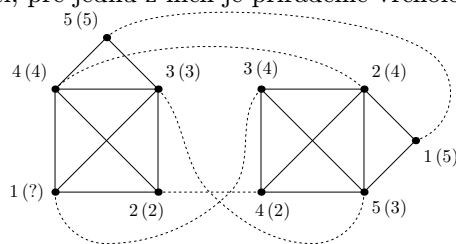
- Funkcia $\text{CountV}(G)$ vráti počet vrcholov v grafe.
- Funkcia $\text{SumV}(G)$ vráti súčet značiek všetkých vrcholov, pričom undef sa počíta ako 0. Pokiaľ graf nemá žiadne vrcholy, funkcia vráti 0.
- Analogicky definujeme funkcie $\text{CountE}(G)$ a $\text{SumE}(G)$ pre hrany a ich váhy.

Globálne operácie

- Funkcia $\text{Iso}(G, H, \text{veq}, \text{eeq})$ zistí, či sú grafy G a H *izomorfné* – t. j., či možno jednému z grafov prečíslovať vrcholy tak, aby sa zhodoval s druhým grafom. Dva grafy sú zhodné, pokiaľ majú rovnaké množiny vrcholov a hrán; navyše sa im musia zhodovať značky vrcholov a váhy hrán podľa toho, ako určujú parametre veq (pre vrcholy) a eeq (pre hrany). Tieto parametre môžu nadobúdať nasledujúce hodnoty:
 - **any**: Ľubovoľné dva vrcholy/hrany sa rovnajú (na ohodnotenie sa nehľadí).
 - **value**: Zodpovedajúce si vrcholy/hrany musia mať rovnaké ohodnotenie. Hodnotu undef ale považujeme za „žolíka“, ktorý sa rovná ľubovoľnej hodnote.
 - **value_strict**: Vrcholy/hrany musia mať presne rovnaké ohodnotenie, undef sa rovná len undef -u.
 - **value_defined**: Vrcholy/hrany musia mať rovnaké ohodnotenie, ale undef sa nerovná ničomu, ani undef -u. Teda akonáhle má nejaký vrchol/hrana ohodnotenie undef , nemôžeme ho priradiť žiadnemu vrcholu/hrane v druhom grafe.
 - **id**: Vrcholy musia mať rovnaké id (toto možno aplikovať len na vrcholy, lebo hrany nemajú id). Inými slovami, zakazujeme prečíslovať vrcholy, ale na ich ohodnotenie nehľadíme.
 - **none**: Žiadne dva vrcholy/hrany nie sú identické. (I keď to vyzerá neúčinne, túto možnosť použijeme v ďalších funkciách.)

Ako izomorfizmus funguje, vidíte na nasledujúcom obrázku. Plné čiary predstavujú hrany dvoch 5-vrcholových grafov. Čísla pred zátvorkami sú id vrcholov, v zátvorkách sú ich značky (otáznik označuje hodnotu undef). Všetky hrany majú váhu undef .

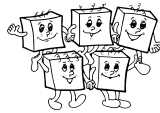
Volaním $\text{Iso}(G, H, \text{any}, \text{any})$ zistíme, či vieme zmeniť id vrcholov grafu G tak, aby sme dostali v G presne tú istú množinu vrcholov ako v H . Takýchto prečíslovaní môže vo všeobecnosti byť aj viac. V príklade na obrázku existujú dve možnosti, pre jednu z nich je priradenie vrcholov zobrazené čiarkovanými čiarami.



Ak navyše požadujeme, aby sa zhodovali aj značky vrcholov, resp. váhy hrán, dosiahneme to zmenením veq , resp. eeq , na inú hodnotu ako any . Grafy z nášho obrázku budú izomorfné, ak nastavíme veq na value alebo any a zároveň eeq na any , value alebo value_strict . (Čiarkované čiary predstavujú priradenie vrcholov, ktoré vo všetkých týchto prípadoch funguje.)

Pri ostatných kombináciách veq a eeq tieto dva grafy izomorfné nie sú.

- Funkcia $\text{Find}(G, H, \text{veq}, \text{eeq})$ nájde podgraf grafu G , ktorý je izomorfný s grafom H . (Podgraf je taký graf, ktorý možno získať z G odstránením niektorých vrcholov a hrán.) Návratovou hodnotou funkcie bude tento podgraf, pričom vrcholy budú očíslované podľa grafu H a ohodnotenie vrcholov a hrán bude pochádzať z

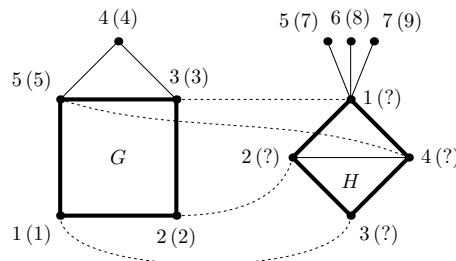


grafu G . Pokiaľ hľadaný podgraf neexistuje, funkcia vráti `EmptyG`. Parametre `veq` a `eeq` určujú rovnako ako pri funkcii `Iso`, ako sa správa izomorfizmus.

Pokiaľ existuje viacej izomorfných podgrafov, funkcia `Find` nájde najľahší z nich (t. j. ten, ktorý má najmenší súčet váh hrán, ako by ho spočítala funkcia `SumE`). Pokiaľ aj tak existuje viacej možností, `Find` vráti ľubovoľnú z nich.

- Funkcia `Common(G, H, veq, eeq)` nájde najväčší spoločný podgraf grafov G a H . Presnejšie, nájde graf, ktorý je izomorfný (podľa `veq` a `eeq`) s niektorým podgrafom G i niektorým podgrafom H . Zo všetkých možných riešení si navyše vyberie také, ktoré má najväčší možný počet vrcholov, a z takých potom to s najväčším počtom hrán. Pokiaľ aj tých je viacej, vyberie si ľubovoľné z nich.

Výsledný graf bude mať id vrcholov v rovnakom poradí, ako ho mal zodpovedajúci podgraf v G , len budú prečíslované od 1 po ich počet, aby v číslovaní neboli diery. Ohodnotenie vrcholov a hrán bude taktiež zdedené z grafu G .



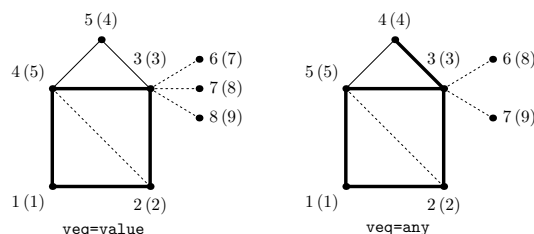
Na obrázku sú opäť plnými čiarami nakreslené dva grafy. Pri `veq=value` a `eeq=any` je ich najväčším spoločným podgrafom „štvorec“ obtiahnutý tučne. Čiarkované hrany ukazujú jedno možné priradenie vrcholov.

Ak zmeníme `veq` na `any`, pribudne do spoločného podgrafu ešte jedna nová hrana a jej koncový vrchol: v ľavom grafe to môže byť jedna z hrán 3-4 a 3-5, v pravom jedna z hrán 1-5, 1-6 a 1-7.

Najväčší spoločný podgraf nemusí byť určený jednoznačne. Napríklad vo vyššie popísanej situácii ním nemusí byť len zvýraznený štvorec. Rovnako veľký je aj podgraf, ktorý je v ľavom obrázku určený vrcholmi 3, 4, 5, 1 a v pravom 4, 1, 2, 3.

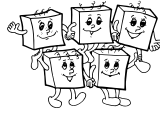
- Funkcia `Join(G, H, veq, eeq)` zlúči grafy G a H . Môžete si to predstaviť tak, že ich „zlepí za ich najväčší spoločný podgraf“. Urobí to tak, že najprv nájde najväčší spoločný podgraf (tak ako vo funkcii `Common`), potom k nemu doplní zostávajúce vrcholy grafu G a nakoniec vrcholy grafu H (id vrcholov výsledného grafu teda budú v tomto poradí). Hrany, váhy a značky pritom zdedí z oboch grafov, pričom pokiaľ sa nejaký vrchol alebo hrana vyskytujú v oboch grafoch, použije sa ohodnotenie z grafu G .

Ak by sme pre grafy z predchádzajúceho obrázku použili `Join`, mohli by sme dostať napr. nasledujúci výstup:



Tučnými čiarami je vyznačená spoločná časť (všimnite si rozdiely v id vrcholov), tenké neprerušované hrany pochádzajú z grafu G , čiarkované hrany sú z grafu H .

(Keďže aj pri `veq=value`, aj pri `veq=any` existovalo viac možností, ako vybrať najväčší spoločný podgraf, existuje aj viacero možností, ako bude vyzeráť výstup funkcie `Join` pre tieto dva grafy. Výstup na obrázku vľavo zodpovedá priradeniu z obrázku k funkcii `Common`, výstup na obrázku vpravo zodpovedá tomu istému priradeniu a navyše vrchol 4 vľavo je priradený vrcholu 5 vpravo.)

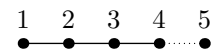


Všetky operácie predpokladajú, že dostanú korektný vstup – nie je teda napríklad povolené volať ich s id neexistujúceho vrcholu alebo upravovať grafovú premennú, do ktorej ste ešte nepriradili, a podobne. Všetky grafové operácie trvajú konštantný čas.

Príklad 1: Tvorba cesty

Ukážeme, ako vytvoriť cestu dĺžky n . To je graf s $n + 1$ vrcholmi a n hranami, v ktorom je každý vrchol spojený hranou s nasledujúcim. Určíte by sme cestu mohli vytvárať postupne, napríklad takto:

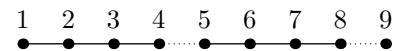
```
function cesta(n: Integer): Graph;
var i, posledny, novy: Integer;
    G: Graph;
begin
  G := EmptyG;
  posledny := AddV(G, 0);
  for i := 1 to n do begin
    novy := AddV(G, 0);
    AddE(G, posledny, novy, undef);
    posledny := novy;
  end;
  cesta := G;
end;
```



Začínáme s jediným vrcholom (má id 1) a potom n -krát pridáme nový vrchol a hranu doň. (Vrcholom dávame značky 0, hranám nedefinované váhy, čo sa bude hodiť neskôr.) Celý postup teda trvá lineárne dlho a vytvorí cestu začínajúcu vo vrchole s id 1 a končiacu vo vrchole s id $n + 1$.

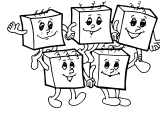
Ten istý graf však vieme zostrojiť aj rýchlejšie. Predstavme si na chvíľku, že už máme v G zostrojenú časť cesty, povedzme tvorenú k vrcholmi. Pomocou `Join(G,G,none,none)` vytvoríme nový graf, ktorý obsahuje dve kópie tejto cesty: jednu s id $1, \dots, k$, druhú s id $k + 1, \dots, 2k$. Do toho stačí následne pridať hranu z k do $k + 1$ a máme cestu dĺžky $2k$. Tento prístup využijeme v nasledujúcom (rekurzívnom) riešení úlohy:

```
function cesta(n: Integer): Graph;
var vysledok: Graph;
    polovica: Integer;
begin
  if n = 0 then begin { cesta dlzky 0 je len jeden vrchol }
    vysledok := EmptyG;
    AddV(vysledok, 0);
  end else begin
    { rekurzivne vytvorime cestu polovicnej dlzky }
    polovica := (n-1) div 2;
    vysledok := cesta(polovica);
    { vyrobime 2 kopie a spojime ich }
    vysledok := Join(vysledok, vysledok, none, none);
    AddE(vysledok, polovica+1, polovica+2, undef);
    { ked polovica nevysla celociselna, pridame este hranu }
    if n mod 2 = 0 then begin
      AddV(vysledok, 0);
      AddE(vysledok, n, n+1, undef);
    end;
  end;
  vysledok := vysledok;
end;
```



Pri každom rekurzívnom volaní sa n zmenší aspoň na polovicu, časová zložitosť tohto riešenia je teda $O(\log n)$.

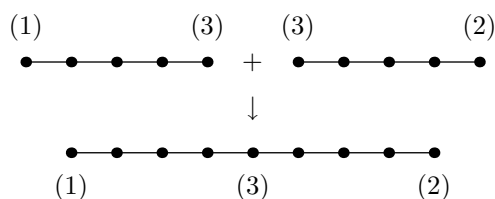
Ukážeme ešte jedno riešenie, tentoraz založené na spájaní ciest za vrchol. Budeme vytvárať cesty, ktorých začiatkový vrchol bude mať značku 1, koncový vrchol značku 2 a všetky ostatné vrcholy značku `undef`. Keď chceme dve cesty spojiť do jednej, preznačíme koncový vrchol prvej a začiatkový vrchol druhej na 3 a zavoláme



Join s `veq=value_defined`. Tým spôsobíme, že sa vrcholy označené trojkou stotožnia a vznikne cesta dvojnásobnej dĺžky. (Všimnite si, že keby sme namiesto `value_defined` použili `value` alebo `value_strict`, stotožnili by sa aj vnútorné vrcholy ciest, čo nechceme.) Potom ešte odstránime pomocnú značku 3 a prepíšeme ju na `undef`. Program tentokrát pre jednoduchosť napíšeme len pre $n = 2^k$:



```
function cesta(n: Integer): Graph;
var G, T1, T2: Graph;
begin
  if n = 1 then begin
    G := EmptyG;
    AddV(G, 1);
    AddV(G, 2);
    AddE(G, 1, 2, undef);
  end else begin
    T1 := cesta(n div 2);
    T2 := T1;
    ReplaceV(T1, 2, 3);
    ReplaceV(T2, 1, 3);
    G := Join(t1, t2, value_defined, any);
    ReplaceV(G, 3, undef);
  end;
  cesta := G;
end;
```



Časová zložitosť tohto riešenia je opäť logaritmická od n .

Príklad 2: Obchodný cestujúci

Všetci poznáme prešibaných obchodníkov. Predávajú ktovie čo a najradšej by boli, keby ich po predaji už kupujúci nikdy nenašiel. Predstavme si takého obchodníka. Teraz sa nachádza v meste (t. j. vo vrchole) číslo 1. Chce prejsť celú krajinu (graf) po cestách (hranách) tak, aby navštívil každé mesto práve raz a potom sa vrátil domov. Navyše pri tom chce najazdiť čo najmenej. Inými slovami, celková váha použitých hrán by mala byť najmenšia možná.

Na obvyklom počítači pre tento problém nepoznáme riešenie v polynomiálnom čase. Pokiaľ ale máme k dispozícii grafový počítač, pôjde to veľmi efektívne.

Stačí totiž vyrobiť cyklus z n hrán a potom funkciou **Find** nájsť jeho najľahší výskyt v grafe predstavujúcom mapu krajiny. Cyklus vytvoríme tak, že podľa predchádzajúceho príkladu vytvoríme cestu s n vrcholmi a $n - 1$ hranami a potom spojíme hranou jej prvý vrchol s posledným. To vieme spraviť v logaritmickom čase. Následné volanie funkcie **Find** beží v konštantnom čase, a teda nám časovú zložitosť nepokazí.

```
function cestujuci(mapa: Graph): Graph;
var trasa: Graph;
begin
  trasa := cesta(CountV(mapa)-1);
  AddE(trasa, 1, CountV(mapa), undef);
  cestujuci := Find(mapa, trasa, any, any);
end;
```

Simulátor

Aby sme vám uľahčili ladenie programov, vytvorili sme simulátor grafového počítača v podobe knižnice pre FreePascal. Linku na jeho stiahnutie nájdete na webe Olympiády v informatike.

SLOVENSKÁ KOMISIA OLYMPIÁDY V INFORMATIKE
DVADSIATY ŠIESTY ROČNÍK OLYMPIÁDY V INFORMATIKE

Vydala IUVENTA s finančnou podporou Ministerstva školstva SR
Náklad: 400 výtlačkov
Zodpovedný redaktor: Michal Forišek
Sadzba programom L^AT_EX

© Slovenská komisia Olympiády v informatike, 2010