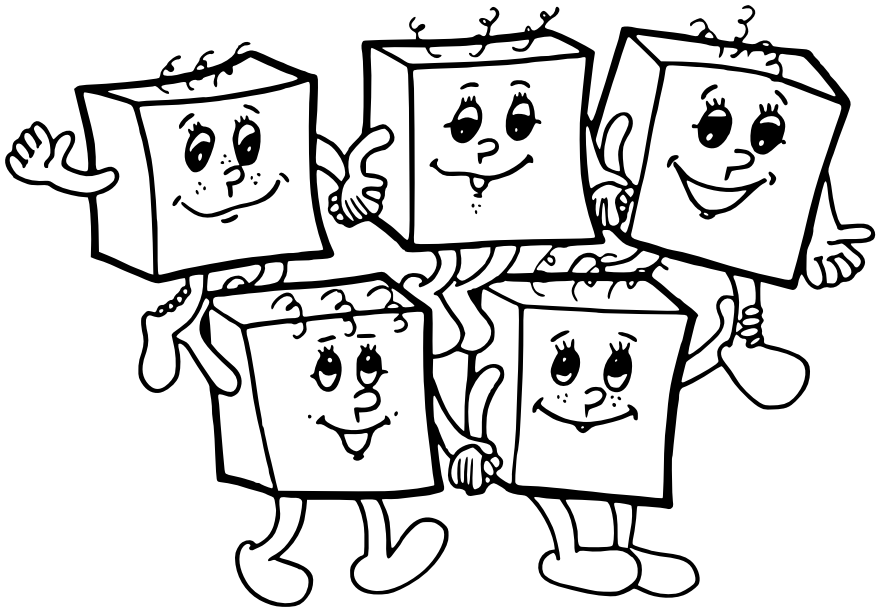
OLYMPIÁDA V INFORMATIKE
NA STREDNÝCH ŠKOLÁCH



dvadsiaty druhý (a zároveň prvý) **ročník**
školský rok 2006/07

zadania domáceho kola
kategória A

- **Olympiáda v informatike** je od tohto školského roku samostatnou súťažou. Predchádzajúcich 21 ročníkov tejto súťaže prebiehalo pod názvom **Matematická olympiáda, kategória P** (programovanie).
- Oficiálnu **webstránku** súťaže nájdete na `http://www.ksp.sk/oi/`.
- Novinkou je zavedenie **dvoch kategórií**. Nová kategória B je určená pre mladších riešiteľov. Podrobnosti nájdete v pravidlách.

Informácie a pravidlá

Pre koho je súťaž určená?

OI sa uskutočňuje v týchto kategóriách:

a) **kategória A** je určená pre žiakov tretieho a štvrtého ročníka stredných škôl a príslušných ročníkov viacročných gymnázií a má tri kolá: domáce, krajské a celoštátne
b) **kategória B** je určená pre žiakov prvého a druhého ročníka stredných škôl a príslušných ročníkov viacročných gymnázií a má dve kolá: domáce a krajské

Do každej kategórie sa môžu zapojiť aj žiaci nižších ročníkov ako tí, pre ktorých je určená.

Ako bude súťaž prebiehať?

V *domácom kole* každej kategórie účastníci riešia štyri úlohy. Riešenia odovzdajú svojmu učiteľovi informatiky do **15. novembra 2006**. Učitelia informatiky odošlú riešenia v tomto termíne zo školy na adresu príslušnej krajskej komisie OI. Adresy krajských komisií sú uvedené na poslednej strane tohto letáku.

Najúspešnejší riešitelia domáceho kola sú pozvaní do *krajského kola*, kde riešia štyri teoretické úlohy.

V kategórii A sú do *celoštátneho kola* pozývaní najúspešnejší riešitelia krajských kôl. Presnejšie, po vyhodnotení krajských kôl prebehne koordinácia bodovacích škál, spoja sa výsledkové listiny do jednej celoštátnej, a do celoštátneho kola sú pozvaní najlepší riešitelia podľa tejto výsledkovej listiny.

V celoštátnom kole účastníci prvý deň riešia tri teoretické úlohy, druhý deň dve praktické úlohy (pri počítači). Z najlepších riešiteľov tohto kola SK OI vyberie družstvá pre Medzinárodnú informatickú olympiádu (IOI) a Stredoeurópsku informatickú olympiádu (CEOI).

Ako majú vyzerať riešenia domáceho kola?

Riešenia súťažných úloh domáceho kola pozostávajú z dvoch častí:

Popis riešenia. Riešenia musia obsahovať podrobný popis použitého algoritmu, **zdôvodnenie jeho správnosti** a diskusiu o efektivite zvoleného riešenia (t.j. posúdenie časových a pamäťových nárokov programu). Algoritmus by mal byť jasný už z popisu riešenia, teda bez toho, aby bolo potrebné nahliadnuť do programu.

Program. V úlohách 1, 2 a 3 je potrebné k riešeniu pripojiť odladený program napísaný v jazyku Pascal, C alebo C++. (Musí sa dať skompilovať kompilátorom *FreePascal*, resp. *gcc*.) Program sa odovzdáva v písomnej forme (vytlačený) aj v elektronickej forme (na CD, prípadne diskete).

Svoje programy pomenujte k`-i-`x`.pas/.c/.cpp`, kde k je kategória a x je číslo súťažnej úlohy. Z jednej školy možno poslať všetky riešenia na jednom CD. V takomto prípade pre každého riešiteľa vytvorte podadresár označený jeho priezviskom.

V úlohe 4 odovzdávate program len v písomnej podobe.

Popis riešenia vypracujte čitateľne na listy formátu A4. **Každú úlohu začnite na novom liste** a v záhlaví uveďte vaše meno, ročník, adresu školy a označenie príkladu podľa tohto letáku. Zadania úloh nemusíte opisovať. Ak sa vám riešenie úlohy nezmestí na jeden list, uveďte na ďalších listoch vľavo hore svoje meno a označenie úlohy, listy očíslujte a zopnite.

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 A

Do **15. novembra 2006** odovzdajte riešenia týchto úloh svojmu učiteľovi informatiky. Ten ich dňa 15. novembra 2006 odošle na adresu príslušnej krajskej komisie OI. Adresy krajských komisií sú uvedené na poslednej strane tohto letáku.

A-I-1 Rozvoz pizze

Marcove dve životné lásky boli kulinárstvo a cyklistika. A ako to tak chodí, jedného dňa prišiel na to, že môže obe uplatniť na svoju obživu, a založil si firmu na výrobu a rozvoz pizze. Plánuje ju prevádzkovať tak, že vždy najskôr nazbiera objednávky, a keď ich už bude mať dosť, napečie pizzu, sadne na bicykel a pôjde ju rozvážať.

Marco zatiaľ vie piecť len jediný druh pizze. Keďže si ale uvedomil, že tým konkurenciu nepredbehne, rozhodol sa nalákať zákazníkov na to, že bude prijímať objednávky aj na šestinové časti pizze. Bude sa teda dať objednať napríklad 1/6, 4/6 alebo 15/6 pizze.

Navyše Marco vyhlásil, že pizzu nebude pred doručením krájať viac ako je nutné (aby si ju zákazník mohol nakrájať podľa svojho gusta). Presnejšie, zákazník dostane čo najviac celých pízz a jeden kus tvoriaci necelú časť jeho objednávky. Teda napríklad zákazník, ktorý si objedná 15/6 pizze, dostane dve celé pizze a ešte jeden kus veľkosti pol pizze.

Presne v okamihu, kedy mu priniesli z tlačiarne rozmnožené reklamné letáky, si Marco uvedomil, že nebude vôbec ľahké skombinovať objednávky tak, aby mu nezostala kopa kúskov pizze, ktoré nik nechce. Obrátil sa preto na vás, aby ste mu pomohli.

Súťažná úloha

Napíšte program, ktorý na vstupe dostane údaje o jednotlivých objednávkach a vypočíta, koľko najmenej pízz stačí Marcovi napiecť a vhodne rozkrájať, ak chce uspokojiť všetky objednávky.

Formát vstupu

Prvý riadok vstupu obsahuje jedno celé číslo N ($1 \leq N \leq 10\,000$) – počet objednávok.

Nasleduje N riadkov. Každý z nich popisuje jednu objednávku: obsahuje jedno celé číslo c_i ($1 \leq c_i \leq 100$) – počet objednaných šestín pizze.

Formát výstupu

Na výstup vypíšte jeden riadok a v ňom jedno celé číslo p – najmenší počet pízz, ktoré stačia na splnenie všetkých objednávok, keď ich Marco upečie a vhodne rozkrája. Nezabudnite na Marcov sľub zákazníkom, že doručené časti pizze nesmú byť rozkrojené.

Príklady

vstup	výstup
3 2 2 3	2

Je viac možností, ako uvedené kusy vyrobiť. Napríklad z jednej pizze vyrežeme dva kusy po 2/6 a druhú rozrežeme napoly a jednu z polovíc doručíme.

vstup	výstup
3 4 5 3	3

V tomto prípade treba kvôli každému z kusov upiecť celú pizzu. Dve pizze síce majú dostatočnú plochu, ale nevieme ich vhodne nakrájať.

A-I-2 Zasypané mesto

Archeológ Alfréd Hrozný skúma nedávno nájdené zasypané mesto v púšti. Ako prvý krok sa rozhodol, že pomocou sonaru určí, koľko miestností mali spolu všetky domy v meste.

Kus púšte, kde mesto ležalo, si Alfréd pokryl štvorcovou sieťou s rozmermi $M \times N$. V ruke so sonarom postupne prešiel všetky riadky takto vytvorenej štvorcovej siete a svoje merania si zaznamenal. Pre jednoduchosť predpokladal, že pod každým poľom tejto siete sa nachádza buď kameň, alebo piesok. Na základe získaných dát by rád určil, koľko miestností (súvislých oblastí piesku) v zasypanom meste bolo.

Súťažná úloha

Na vstupe je daný popis zasypaného mesta, ktoré si predstavujeme ako štvorcovú sieť s rozmermi $M \times N$. Políčka štvorcovej siete sú popísané po riadkoch od horného k spodnému a v jednotlivých riadkoch postupne zľava doprava.

Alfréd si svoje merania zapísal v skrátenej podobe. Ak bolo v riadku za sebou x políčok rovnakého typu, zapísal si jednoducho len číslo x. Zo zápisu „3 2 7“ by však ešte nebolo jasné, či je na prvých troch políčkach kamenie alebo piesok. Preto Alfréd zapísal svoje merania nasledovne:

Zápis každého riadku tvorí niekoľko dvojíc **nezáporných** celých čísel. Prvé číslo z každej dvojice udáva počet políčok s pieskom, druhé počet nasledujúcich políčok s kameňmi.

Vašou úlohou je spočítať počet miestností v zasypanom meste. Miestnosť je súvislá oblasť piesku, ktorú už v žiadnom smere nie je možné zväčšiť. Dve políčka štvorcovej siete považujeme za susedné, ak majú spoločnú hranu (spoločný vrchol nestačí).

Formát vstupu

V prvom riadku vstupu sa nachádzajú tri nezáporné čísla M, N a K – počet riadkov a stĺpcov štvorcovej siete mesta a celkový počet dvojíc, ktoré popisujú jej obsah. Je známe, že M a N sú menšie než 50 000 a že K je menšie ako 1 000 000 000.

Každý z ďalších K riadkov obsahuje dve nezáporné celé čísla p a q, kde p je počet políčok zasypaných pieskom a q je počet nasledujúcich políčok, pod ktorými sú kamene. Políčka sú popísané po riadkoch začínajúc od horného riadku siete, v rámci jedného riadku vždy zľava doprava. Každá dvojica čísel popisuje úsek, ktorý celý leží v jednom riadku.

Pokiaľ sa Vám nepodarí vyriešiť úlohu s vyššie popísanými obmedzeniami na M, N a K, predpokladajte, že M a N sú najviac 500 a K je najviac 100 000.

Formát výstupu

Na výstup vypíšte jediný riadok a v ňom jediné nezáporné číslo – počet miestností v zasypanom meste. Všimnite si, že ak sú úplne všade kamene, tak je toto číslo rovné nule.

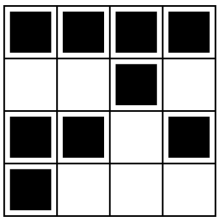
Príklad

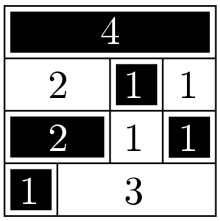
vstup

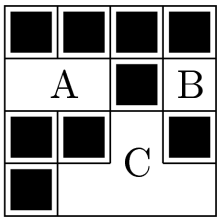
```
4 4 7
0 4
2 1
1 0
0 2
1 1
0 1
3 0
```

výstup

```
3
```







Na prvom obrázku je znázornené mesto z príkladu. Na druhom sú uvedené čísla, ktoré si preň zapísal Alfréd (okrem núl). Na treťom obrázku sú písmenami označené miestnosti nachádzajúce sa v meste.

A-I-3 Okružná jazda

V mestečku Blatislava je neuveriteľne komplikovaný dopravný systém. Tvoria ho navzájom poprepájané križovatky. V každej križovatke sa môže stretávať ľubovoľne veľa ulíc. Nedávno sa miestny starosta rozhodol, že je čas na reformu dopravnej situácie.

Najskôr z každej ulice urobil jednosmerku, pričom si dal pozor na to, aby do každej križovatky aspoň jedna cesta vchádzala a aby z každej križovatky aspoň jedna cesta vychádzala. Následne v každej križovatke zakázal jednu možnosť odbočenia. (Teda ak do križovatky viedlo k ulíc a z nej ℓ ulíc, po zavedení zákazu sa dala prejsť $k\ell - 1$ spôsobmi.)

Reforma mala úspech, Blatislavčanom neuveriteľne sťažila pohyb po uliciach. Mnohí sa nevedeli dostať do práce, prípadne (tí menej šťastní) z práce domov. Aby starosta podobné tvrdenia vyvrátil, rozhodol sa nájsť *okružnú jazdu* – cyklickú postupnosť ulíc, ktoré nasledujú po sebe, všetky odbočenia v nej sú povolené, nikdy nejdeme v protismere a každá ulica v meste sa v tejto postupnosti vyskytuje práve raz.

Ak by starosta okružnú jazdu našiel, bolo by každému jasné, že z každej ulice mesta sa dá dostať na každú inú. Všimnite si ale, že takáto postupnosť nemusí existovať. Na to napríklad stačí, aby v meste bola križovatka, z ktorej vychádza menej ulíc ako do nej vchádza.

Súťažná úloha

Napíšte program, ktorý zistí, či v meste existuje okružná jazda, a ak áno, tak jednu ľubovoľnú nájde.

Formát vstupu

V prvom riadku vstupu sú dve prirodzené čísla N a M – počet križovatiek a počet jednosmeriek. Križovatky sú očíslované od 1 do N.

Nasleduje M riadkov, každý popisuje jednu ulicu: obsahuje dve čísla u a v ($1 \leq u, v \leq N$), ktoré hovoria, že z križovatky u vedie jednosmerka do križovatky v. Medzi každou dvojicou križovatiek vedie v každom smere najviac jedna ulica.

Nasleduje N riadkov, i-ty z nich popisuje zakázané odbočenie na križovatke s číslom i: obsahuje dve čísla u a v ($1 \leq u, v \leq N$), ktoré hovoria, že ak do križovatky i prichádzame z križovatky u, nesmieme odísť ulicou vedúcou ku v.

Formát výstupu

Ak existuje riešenie, vypíšte jeden riadok a v ňom postupnosť medzerami oddelených čísel v_1, v_2, ..., v_m ($1 \leq v_i \leq N$ pre každé i) takú, že:

- z v_i do v_{i+1} (pre $1 \leq i \leq M$) vedie ulica,
- ak ideme po ulici z v_i do v_{i+1}, je povolené odbočiť do ulice z v_{i+1} do v_{i+2} (pre $1 \leq i \leq M$),
- každá ulica sa v postupnosti nachádza, t. j. ak existuje ulica z u do v, tak existuje i také, že $v_i = u$ a $v_{i+1} = v$.

Indexy počítáme cyklicky, t. j. $v_{m+1} = v_1$ a $v_{m+2} = v_2$. Ak existuje viac riešení, vypíšte jedno ľubovoľné. Ak neexistuje žiadne riešenie, namiesto postupnosti vypíšte reťazec „`Okruzna jazda neexistuje.`“.

Príklady

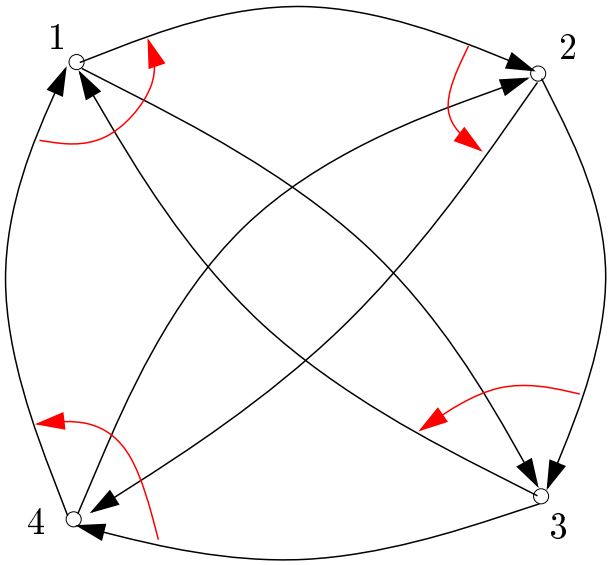
vstup

```
4 8
1 2
2 3
3 1
3 4
1 3
4 2
2 4
4 1
4 2
1 4
2 1
3 1
```

výstup

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

Na obrázku je cestná sieť z prvého príkladu. Šípky okolo križovatiek ukazujú zakázaný smer odbočenia. Vypísaný výstup je jeden z viacerých možných.



vstup

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

výstup

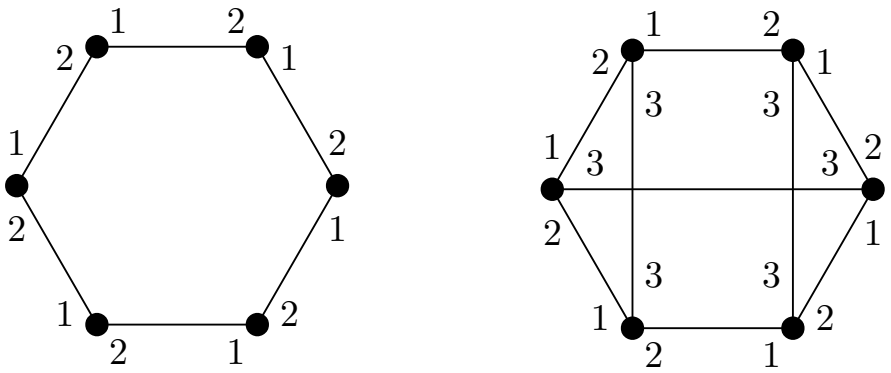
```
Okruzna jazda neexistuje.
```

A-I-4 Grafomat

Študijný text

Grafom nazývame ľubovoľnú konečnú množinu V *vrcholov* grafu spolu s množinou E *hrán*, čo sú neusporiadané dvojice vrcholov. Každá hrana teda predstavuje spojenie medzi dvoma vrcholmi. Žiadne dva vrcholy nie sú spojené viac ako jednou hranou, žiadna hrana nespája vrchol sám so sebou. Počet hrán grafu budeme označovať N a počet jeho hrán M.

K-graf budeme hovoriť takému grafu, v ktorom z každého vrcholu vedie práve K hrán a konce týchto hrán sú očíslované prirodzenými číslami od 1 po K. Konce jednej hrany môžu byť očíslované rôzne. Pokiaľ budeme hovoriť o hranách vychádzajúcich z nejakého vrcholu v, budeme spomínať *miestne* čísla hrán (čísla toho konca, ktorým je v) a *vzdialené* čísla (to sú tie na opačných koncoch hrán). Nasledujúci obrázok ukazuje príklad 2-grafu a 3-grafu.



Ohodnotením grafu nazveme priradenie prvkov nejakej konečnej množiny vrcholom grafu - teda napríklad rozdelenie vrcholov na čierne a biele, alebo označenie vrcholov číslami od 1 po 5.

Grafomat je zariadenie na automatické riešenie grafových úloh. Jeho vstupom je ľubovoľný K-graf G spolu s jeho ohodnotením. Výstupom je nejaké ďalšie ohodnotenie toho istého grafu. Samotný výpočet je vykonávaný *automatmi* umiestnenými v jednotlivých vrcholoch grafu. Každý automat má svoju pamäť a riadi sa programom. Programy všetkých automatov sú identické. Okrem toho,

že automat môže ľubovoľne narábať so svojou pamäťou, môže aj nahliadať do pamäte svojich susedov.

Pamäť automatu si môžeme predstaviť ako pascalovské premenné typu interval. Teda každá premenná obsahuje jedno prirodzené číslo, ktoré je z nejakého pevne zvoleného rozsahu, ktorý nezávisí na veľkosti vstupu. Okrem toho je tiež možné používať pole takýchto premenných. Opäť, rozmery poľa musia byť dopredu známe a nesmú závisieť od veľkosti vstupu. Žiadne iné typy premenných (neobmedzene veľké čísla, smerníky, ...) sa nedajú používať.

Zvláštnu rolu hrajú premenné x a y. Premenná x na začiatku výpočtu obsahuje vstupné ohodnotenie toho vrcholu grafu, v ktorom program beží, hodnota premennej y na konci výpočtu určí výstupné ohodnotenie tohto vrcholu. Všetky premenné s výnimkou premennej x majú svoju počiatočnú hodnotu pevne určenú. Deklarácia premenných vyzerá napríklad takto:

```
var x: 1..5;          { číslo od 1 do 5, na začiatku je to vstup }
    y: 1..5 = 3;           { y je na začiatku 3, na konci výstup }
    z: array [1..2] of 3..4 = (3, 4);         { dvojprvkové pole }
```

Riadiaci program automatu si môžeme predstavit ako pascalovský program, v ktorom zakážeme používanie rekurzie a dovolíme manipulovať len s premennými v pamäti automatu a s premennými v pamäti susedných automatov. Na svoje vlastné premenné sa automat odkazuje ich menami, ako by to boli obyčajné pascalovské globálne premenné. Na to, aby sme vedeli odkazovať na premenné susedov, využijeme miestne čísla hrán. Presnejšie, nech `i` je celočíselný výraz s hodnotou z rozsahu $1 \ldots K$. Potom `S[i].p` je výraz predstavujúci premennú `p` u suseda, do ktorého od nás vedie hrana s miestnym číslom `i`. (Samozrejme, `i` môže byť ľubovoľný výraz a `p` ľubovoľné meno premennej.) Premenné susedov sa dajú len čítať.

Aby program mohol dávať do súvislosti svoje hrany s hranami svojich susedov, má k dispozící ešte premenne `P[1]` až `P[K]`, ktoré sú pevne nastavené tak, že `P[i]` obsahuje vzdialené číslo tej hrany, ktorá má miestne číslo `i`.

Použitie si ukážeme na výraze `S[i].S[P[i]].x`. Označme vrchol, v ktorom práve sme, písmenom v. Výrazy `S[i].volaco` sú premenné suseda, do ktorého sa dostaneme hranou s číslom `i`. Označme tohto suseda písmenom u. Premenná,

ktorú chceme, je `S[nieco].x`. Takže chceme premennú `x` od niektorého suseda vrcholu u. Ale ktorého? Toho, do ktorého sa dostaneme hranou s číslom `P[i]`. To je ale práve vzdialené číslo hrany, ktorou sme do u prišli. Inými slovami, v u je to jeho miestne číslo hrany, ktorá vedie späť do v. Takže výraz `S[i].S[P[i]].x` predstavuje to isté ako výraz `x`.

Ak sa vám to zdá zložité, povieme si to ešte raz v dvoch krokoch. Najskôr sa vyhodnotí `nieco=P[i]`. (`P[i]` je naša lokálna premenná.) Teraz sa pozrieme na premennú `S[i].S[nieco].x`. (To prvé `S` je naše lokálne pole s premennými susedov, to druhé je podobné pole u suseda.) No a tento sused v premennej `S[nieco].x` vidí to, čo máme v našej premennej `x`.

Výpočet automatu prebieha v taktoch, a to nasledovne: V nultom takte sa premenné všetkých automatov nastavia na počiatočnú hodnotu a premenné x na vstupné ohodnotenie jednotlivých vrcholov. V každom ďalšom takte sa vždy znova spustí program každého automatu, pričom premenné svojich susedov vidí program v stave, v akom boli na začiatku taktu. Aj keď jednotlivé automaty bežia súčasne, nemôže sa teda stať, že by jeden čítal z premennej, do ktorej práve druhý zapisuje.

Výpočet pokračuje tak dlho, až kým v nejakom takte všetky automaty nevykonajú špeciálny príkaz `stop`. Potom sa výpočet zastaví a z premenných y grafomat prečíta výstupné ohodnotenie grafu. Pokiaľ príkaz `stop` urobia len niektoré automaty, výpočet pokračuje a to aj na tých automatoch, ktoré príkaz `stop` spravili. Štruktúra grafu a obsahy premenných `P` zostávajú po celú dobu výpočtu nezmenené.

Za *časovú zložitosť* výpočtu budeme považovať počet taktov, ktoré ubehnú po zastavenie programu. Nijako teda nezávisí na rýchlosti jednotlivých automatov. Podobne ako je to u časovej zložitosti klasických algoritmov, nebudeme hľadieť na multiplikatívne konštanty a bude nás zaujímať len asymptotické správanie zložitosti – teda či je lineárna, kvadratická, atď. Prípady, kedy výpočet neskončí, nebudeme pripúšťať, pre úplnosť ale dodajme, že vtedy sa hodnoty premenných musia nutne opakovať.

Príklad 1

Je zadaný 3-graf a v ňom vyznačený jeden vrchol v, a to tak, že jeho pre-

menná x bude inicializovaná jednotkou a ostatné vrcholy budú mať nulu. Napíšte program pre grafomat, ktorý označí všetky vrcholy, do ktorých sa dá dostať z vrcholu v po hranách, a to tak, že ich premenná y bude mať na konci výpočtu hodnotu jedna, pre ostatné vrcholy bude mať hodnotu nula.

Riešenie: Inšpirujeme sa prehľadávaním do šírky. V každom takte sa každý vrchol pozrie, či je niektorý z jeho susedov už označený. Ak je, tak sa sám označí. Pokiaľ sa označenie nezmení, vrchol volaním `stop` súhlasí so zastavením. Priebeh výpočtu bude teda vyzerať tak, že po i-tom takte budú označené tie vrcholy, ktorých vzdialenosť od v je menšia alebo rovná i. Výpočet sa zastaví vtedy, keď sa hodnoty premenných prestanú meniť, čo znamená, že po najviac N taktoch. Preto je časová zložitosť nášho programu lineárna od počtu vrcholov (na rozdiel od klasického prehľadávania do šírky, ktoré závisí od počtu hrán).

Program vyzerá nasledovne:

```
var x: 0..1;                    { bol vrchol označený vo vstupe? }
    y: 0..1 = 0;                              { je označený teraz? }
    prev: 0..1 = 0;                          { predchádzajúci stav }
    i: 1..3;

begin
   prev := y;                  { zapamätáme si, či už bol označený }
   if x=1 then y := 1;            { prenesieme označenie zo vstupu }
   for i := 1 to 3 do           { pozrieme sa na všetkých susedov }
      if S[i].y <> 0 then             { ak je i-ty sused označený }
         y := 1;                                 { označ aj sám seba }
   if y = prev then stop;      { ak sa nič nemení, môžeme skončiť }
end.
```

Príklad 2

Majme 2-graf zložený z jedného cyklu párnej dĺžky, teda z vrcholov očíslovaných $0 \ldots N-1$, pričom vrchol i je spojený hranou označenou 1 s vrcholom $(i+1) \bmod N$ a hranou označenou 2 s vrcholom $(i-1) \bmod N$. (Príklad takého grafu pre $N = 6$ nájdete na obrázku na začiatku tohto textu.) V tomto grafe

je vyznačený jeden vrchol v, rovnako ako v predchádzajúcom príklade. Napíšte program pre grafomat, ktorý označí vrchol protiľahlý k v, teda vrchol s číslom $(v + N/2) \bmod N$.

Riešenie: Vyšleme signál putujúci z vrcholu v v smere jednotkových hrán rýchlosťou 1 vrchol za takt. Zároveň vyšleme druhý signál putujúci rovnakou rýchlosťou opačným smerom. Akonáhle nejaký vrchol zistí, že do neho prišli oba signály, označí sa a signály už ďalej neposiela.

```
var x: 0..1;                                  { vstupná značka vrcholu }
    y: 0..1 = 0;                                      { výstupná značka }
    l, p: 0..1 = 0;           { už týmto vrcholom prešiel signál
                                                 doľava a doprava? }
begin
   if x=1 then                                     { začíname posielať }
      begin l := 1; p := 1; end;
   if (S[2].l=1) and (S[1].p=1) then        { signály sa stretli }
      begin y := 1; stop; end
   else if (S[2].l=1) and (l=0) then l := 1    { pošleme doľava }
   else if (S[1].p=1) and (p=0) then p := 1   { pošleme doprava }
   else stop;                        { nič sa nedeje, môzeme končiť }
end.
```

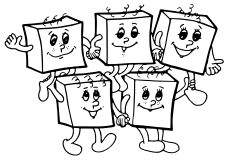
Súťažná úloha

Napíšte program pre grafomat, ktorý v zadanom 3-grafe s vyznačenými dvoma vrcholmi nájde najkratšiu cestu medzi nimi a vyznačí vrcholy ležiace na tejto ceste. Môžete predpokladať, že cesta vždy existuje. Pokiaľ je najkratších ciest viac, vyberte si ľubovoľnú z nich.

V jednom zo zadaných vrcholov bude mať na začiatku premenná x hodnotu 1, v druhom 2 a vo všetkých ostatných vrcholoch bude mať hodnotu 0.

Na konci výpočtu by mala mať premenná y hodnotu 1 vo vrcholoch tvoriacich jednu najkratšiu cestu a hodnotu 0 v ostatných vrcholoch.

Pokúste sa napísať taký program, ktorého časová zložitosť bude závisieť len **na dĺžke zostrojenej cesty** a nie na veľkosti celého grafu.



Adresy krajských komisií OI

Bratislavský kraj
KSP, KZVI FMFI UK, Mlynská dolina, 842 48 Bratislava

Košický a Prešovský kraj
RNDr. Rastislav Krivoš-Belluš, Ústav informatiky PF UPJŠ,
Park Angelinum 9, 040 01 Košice

Žilinský kraj
RNDr. Peter Varša, Ph.D., KI FRI ŽU, Moyzesova 20, 010 26 Žilina

Banskobystrický kraj
PaedDr. L. Huraj, KI FPV UMB, Tajovského 40, 974 01 Banská Bystrica

Trnavský, Trenčiansky a Nitriansky kraj
prof. Ing. V. Stoffová, CSc., KI PF UJS, Roľníckej školy 1519, 945 01 Komárno

SLOVENSKÁ KOMISIA OLYMPIÁDY V INFORMATIKE

DVADSIATY DRUHÝ ROČNÍK OLYMPIÁDY V INFORMATIKE

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