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 sústredenie. Podľa jeho výsledkov SK OI vyberie družstvá pre Medzinárodnú olympiádu v informatike (IOI) a Stredoeurópsku olympiádu v informatike (CEOI).

Odovzdávanie 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 vyzerať 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 vyzerať 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 efektivite 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 **30. novembra 2012**.

B-I-1 Veže

Na veľkú šachovnicu ktosi rozostavil množstvo šachových veží. Niektoré boli biele, iné zas čierne. Barborka je vášnivá šachistka. Rada by si spočítala, koľko dvojíc veží sa navzájom ohrozuje. Keďže ich je ale tak veľa, bojí sa, že sa pomýli. Lepšie by bolo, keby ste jej napísali program, ktorý to vypočíta za ňu.

(Dve veže sa ohrozujú, ak sú opačných farieb, stoja v tom istom riadku alebo stĺpci a medzi nimi nestojí žiadna iná veža.)

Formát vstupu

V prvom riadku vstupu je jedno celé číslo n, udávajúce rozmer šachovnice. Šachovnica má tvar štvorca, jej riadky aj stĺpce sú očíslované od 1 po n. V druhom riadku vstupu je jedno celé číslo v, udávajúce počet veží.

Nasleduje v riadkov, z ktorých každý popisuje jednu vežu. Popis veže pozostáva z troch celých čísel: číslo riadku, číslo stĺpca a jej farba. (Farba 0 predstavuje čiernu vežu, farba 1 bielu.)

Váš program dostane body za správne vyriešenie piatich testovacích vstupov, ktoré sme si vopred pripravili. Rôzne vstupy popisujú rôzne veľké šachovnice s rôznym počtom veží:

	vstup 1	vstup 2	vstup 3	vstup 4	vstup 5
veľkosť šachovnice n	8	50	1000	1000	1 000 000
počet veží v	20	200	1500	100 000	100 000

Formát výstupu

Vypíšte jeden riadok a v ňom jedno celé číslo: počet dvojíc veží, ktoré sa ohrozujú.
(Môžete predpokladať, že toto číslo bude vždy menšie ako milión.)

Príklad

vstup

```
8
5
2 2 1
7 6 0
5 4 0
2 4 1
2 8 0
```

výstup

```
2
```

Tento vstup popisuje šachovnicu 8×8. Na nej je 5 veží. Ich rozmiestnenie je znázornené napravo.

```
........
.1.1...0
........
........
...0....
........
.....0..
........
```

Biela veža na (2,4) sa ohrozuje s čiernou na (2,8). Biela veža na (2,4) sa tiež ohrozuje s čiernou na (5,4).
Všimnite si, že biele veže na (2,2) a (2,4) sa neohrozujú, lebo sú obe tej istej farby.
Biela veža na (2,2) sa neohrozuje s čiernou vežou na (2,8), lebo medzi nimi stojí iná veža.

Odovzdávanie 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`.
Vyrobte 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 Dosah

Janko sa dostal do tímu, ktorý programuje umelú inteligenciu do novej tankovej strieľačky `Megatank 3000`. Táto strieľačka sa odohráva na mriežke rozdelenej na štvorcové políčka. Niektoré políčka sú voľné, na niektorých sú prekážky. Tank sa v jednom kroku môže pohnúť o jedno políčka vľavo, vpravo, hore alebo dole. Samozrejme, len vtedy, ak na cieľovom políčku nie je prekážka.

Jankovou úlohou je teraz napísať program, ktorý vyhodnocuje výhodnosť pozície tanku. Presnejšie, potrebuje zistiť, na koľko rôznych políčok sa z daného začiatočného políčka vie tank dostať použitím nanajvýš zadaného počtu krokov. Keďže Janko sa v predchádzajúcom zamestnaní živil ako programátor sústruhov a o tejto úlohe nemá ani šajnu, musíte mu pomôcť vy.

Formát vstupu

V prvom riadku vstupu je číslo t – počet úloh, ktoré musíte vyriešiť.

V prvom riadku úlohy sú vždy tri celé čísla r, s, q – rozmery mriežky a počet otázok. Mriežka má r riadkov očíslovaných od 1 do r a s stĺpcov očíslovaných od 1 do s.

Nasleduje r riadkov vstupu, pričom každý popisuje jeden riadok mriežky. V každom z týchto riadkov je s medzerou oddelených čísel 0 alebo 1. Číslo 0 predstavuje voľné políčko, číslo 1 predstavuje prekážku.

V ďalších q riadkoch vstupu sú jednotlivé otázky. Každá otázka pozostáva z troch celých čísel y, x, k. Čísla y, x označujú riadok a stĺpec začiatočného políčka. (Môžete predpokladať, že toto políčko neobsahuje prekážku.) Číslo k označuje maximálny počet krokov, ktoré môže tank urobiť.

Váš program dostane body za správne vyriešenie piatich vstupných súborov, ktoré sme si vopred pripravili. (Každý súbor obsahuje viacero úloh. Body za vstup dostanete len vtedy, ak správne vyriešite úplne všetky úlohy, ktoré obsahuje.)

Rôzne vstupy popisujú rôzne veľké mriežky s rôznym maximálnym počtom krokov a rôznym počtom otázok:

	vstup 1	vstup 2	vstup 3	vstup 4	vstup 5
maximálny rozmer mriežky	7	500	100	500	2 000
maximálny počet krokov	3	3	5000	20000	1 000 000
celkový počet otázok	47	63	62	73	12

Formát výstupu

Pre každú otázku z každej úlohy vypíšte jeden riadok s jedným číslom – počtom políčok, na ktoré sa dá dostať zo začiatočného na maximálne k krokov.

Príklad

vstup

```
2
4 4 2
0 0 0 0
0 1 1 0
0 0 1 0
0 0 0 0
1 1 1
4 1 3
2 3 1
0 0 1
1 1 1
1 1 10
```

výstup

```
3
8
2
```

Tento súbor obsahuje dve úlohy. V prvej z nich má mriežka rozmery 4×4, v druhej 2×3.
V prvej úlohe treba zodpovedať dve otázky. V prvej tank začína na políčku $(1, 1)$ a smie spraviť 1 krok. V druhej otázke tank začína na políčku $(4, 1)$ a smie spraviť 3 kroky. Táto otázka je znázornená na nasledujúcom obrázku. Hviezdičkou sú označené políčka, na ktoré sa tank vie dostať.

```
* 0 0 0
* 1 1 0
* * 1 0
* * * *
```

B-I-3 Súčet súčtov

Deti v druhej bé zase na hodine matematiky nedávali pozor a ohadzovali sa kriedou. A tak im dal učiteľ za trest úlohu: Na tabuľu napísal postupnosť čísel. Každý žiak dostal pridelený jeden úsek postupnosti, ktorý mal sčítať. Zhodou okolností bolo všetkých možných úsekov presne toľko ako žiakov, takže každému sa ušiel iný úsek. Keď každý žiak sčítal svoj úsek, napísal svoj výsledok na list papiera. Nakoniec museli žiaci zozbierať všetky listy papiera a sčítať čísla na nich. A až keď učiteľovi oznámili celkový výsledok, pustil ich domov.

Ukážeme si to celé na malom príklade. Majme triedu s desiatimi žiakmi. Učiteľ na tabuľu napísal postupnosť $(10, 20, 10, 30)$. Následne rozdelil deťom prácu:

– štyria žiaci dostali sčítať úseky dĺžky 1: (10), (20), (10) a (30),
– traja žiaci dostali sčítať úseky dĺžky 2: $(10, 20)$, $(20, 10)$ a $(10, 30)$,
– dvaja žiaci dostali sčítať úseky dĺžky 3: $(10, 20, 10)$ a $(20, 10, 30)$,
– jeden žiak (ten, čo najviac vyrušoval) dostal sčítať celú postupnosť.

Žiaci takto dostali na svojich papieroch výsledky 10, 20, 10, 30, 30, 30, 40, 40, 60 a 70. Súčtom všetkých týchto výsledkov je číslo 340.

Súťažná úloha

Napíšte program, ktorý bude túto úlohu vedieť riešiť pre čo najdlhšie postupnosti.

Pri písaní programu môžete predpokladať, že sa vám výsledok zmestí do bežnej číselnej premennej. Netreba sa teda zaoberať pretečením premenných.

Ľubovoľné funkčné riešenie s dobrým popisom (bez ohľadu na časovú zložitosť) môže dostať aspoň štyri body. Riešenie, ktoré dostane plný počet bodov, si hravo poradí aj s postupnosťou tvorenou stotisíc číslami.

Formát vstupu a výstupu

V prvom riadku vstupu je jedno celé číslo n, udávajúce dĺžku postupnosti na tabulu. V druhom riadku je n malých kladných celých čísel, tvoriacich dotyčnú postupnosť.

Vypíšte jeden riadok a v ňom jedno celé číslo: súčet súčtov všetkých súvislých úsekov danej postupnosti.

Príklad

vstup

```
4
10 20 10 30
```

výstup

```
340
```

Toto je príklad rozpísaný v zadaní.

Odovzdávanie riešení

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

B-I-4 Paralelné trampoty

Podúloha A – 6 bodov

Bola raz jedna trieda, v tej triede bola krieda. Tá krieda bola biela a bolo jej dosť veľa. Okrem zásob kriedy na desať rokov bola v triede ešte tabuľa a na tabuli krásna veľká **nula**. Pred triedou stálo 30 detí. Každé z nich postupne 30-krát zopakovalo nasledovný postup:

– Nejaký čas (aký dlhý sa mu chcelo) počkalo.
– Vošlo do triedy, prečítalo si číslo z tabule, zapamätalo si ho a vyšlo zase von.
– S vypätím všetkých síl k zapamätanému číslu pripočítalo 1. (Aj toto mohlo trvať ľubovoľne dlho.)
– Vošlo do triedy, zmazalo tabuľu a napísalo na ňu výsledok svojho výpočtu.

Konkrétne dieťa vždy najskôr dokončí celý postup, až potom začne odznova. Teda sa napríklad nemôže stať, že to isté dieťa dvakrát po sebe príde prečítať tabuľu, musí medzi tým prísť zapísať výsledok.

Na prvý pohľad je očividné, ako to celé dopadne, nie? Na konci bude na tabuli číslo $30 \times 30 = 900$.

Na druhý pohľad to už až také očividné nie je. Samozrejme, 900 je *najväčší* možný výsledok. Mohlo sa však ľahko stať, že skutočný výsledok bude menší. Nás zaujíma opačný extrém. Vašou úlohou je nájsť *čo najmenšiu* hodnotu, ktorá mohla byť na tabuli po tom, ako každé z 30 detí 30-krát zopakovalo popísaný postup.

Počet bodov, ktoré dostanete, bude závisieť od toho, ako malú hodnotu sa vám podarí zostrojiť. Na plný počet bodov je potrebné nájsť *najmenšiu možnú* hodnotu a stručne zdôvodniť, prečo menšiu už nevieme dosiahnuť.

Podúloha B – 4 body

Na podobnom princípe je založený jeden typ zraniteľnosti počítačových systémov, tzv. *race condition*. Ide o situácie, kedy si programátor systému neuvedomil, že výsledok súbežného vykonávania viacerých procesov môže závisieť od toho, v akom poradí sa jednotlivé kroky vykonajú. Ukážeme si to na jednoduchom príklade. (Na pár miestach používame Linuxovú terminológiu. Neboj sa jej, na riešenie súťažnej úlohy netreba poznať Linux.)

Samko je správcom počítača, na ktorom má účty viacero ďalších užívateľov. Nás budú zaujímať dvaja z nich: maliarka Marienka a hacker Janko. Marienka nakreslila krásneho poníka a uložila ho do súboru `ponik1.png` vo svojom domovskom adresári. Janko by si ho chcel pozrieť, Marienka však nastavila prístupové práva tak, že sa k nemu vie dostať len ona (a samozrejme správca). A tak mal Janko smolu. Až jedného dňa prišiel Samko za Jankom a hovorí: „Pozri, akú kúl vec som spravil. Program `posli`, ktorému napíšeš názov súboru a e-mailovú adresu, a on ti na ňu ten súbor pošle.“ Janko si všimol, že Samkov program beží s právami správcu systému. A viac mu už nebolo treba. Spustil „`posli /home/marienka/ponik1.png jankov@mail`“ a mal poníka.

Samko chybu odhalil a opravil: prirobil do programu kontrolu, že posielaný súbor musí byť v domovskom adresári užívateľa, ktorý ho práve spustil. Ani to však Janka nezastavilo, mal v zásobe ďalší trik: *symbolickú linku*. (Vo Windows je podobným objektom *zástupca*, po anglicky shortcut.) To je súbor, ktorý nezaberá skoro žiadne miesto, len hovorí: „môj obsah je rovnaký ako obsah tohto iného súboru“. A tak si Janko vo svojom adresári vyrobil symbolickú linku `2.png`, ukazujúcu na súbor `/home/marienka/ponik2.png`, ktorý Marienka medzičasom nakreslila. A už to išlo: Janko spustil „`posli /home/janko/2.png jankov@mail`“. Samkov program skontroloval, že ide o súbor v adresári `/home/janko`, prečítal ho, poslal mailom a Janko mal už druhého poníka.

A tu sa už dostávame k našej súťažnej úlohe. Medzi časom Marienka nakreslila tretieho poníka a Samko vylepšil bezpečnosť svojho programu. Samkov program teraz funguje nasledovne: keď ho užívateľ X spustí ako „`posli /nejaka/cesta/subor email`“, program postupne:

1. otvorí adresár `/nejaka/cesta` a skontroluje, či X má právo čítať `subor`.
2. ak áno, otvorí `subor`, načíta ho a prepošle ho na adresu `email`.

Toto vylepšenie by hravo zmarilo Jankov predchádzajúci pokus – totiž Janko síce vedel vyrobiť v svojom adresári symbolickú linku ukazujúcu na obrázok poníka, ale obsah takto vytvoreného súboru nemal právo čítať.

Navrhnite, ako má Janko postupovať, aby sa dostal k súboru `ponik3.png`.

Odovzdávanie 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. novembra 2012**.

A-I-1 Špióni

Kontrarozviedke v Absurdistane sa podaril husársky kúsok: odhalili celú sieť nepriateľských špiónov v krajine.

Niektoré dvojice špiónov spolu komunikujú. Z bezpečnostných dôvodov je táto sieť pomerne riedka – komunikujúcich dvojíc je málo. Presnejšie, pre ľubovoľné prirodzené číslo k platí: ak ľubovoľným spôsobom vyberieme k špiónov, bude medzi nimi najviac $3k$ dvojíc špiónov, ktorí spolu komunikujú.

Súťažná úloha

Nájdite v danej sieti špiónov najväčšiu *kliku*: skupinu špiónov, v ktorej spolu komunikuje každá dvojica.

(Pomôcka: Z riedkosti komunikačnej siete vyplýva, že určite neexistuje klika tvorená viac ako 7 špiónmi.)

Formát vstupu

V prvom riadku vstupu sú dve celé čísla n a m. Číslo n udáva počet špiónov, číslo m je počet dvojíc, ktoré spolu komunikujú. Špióni majú pridelené čísla od 1 po n.

Každý z nasledujúcich m riadkov obsahuje dve rôzne čísla od 1 po n, popisujúce jednu dvojicu špiónov, ktorá spolu komunikuje. (Žiadne dva riadky nepopisujú tú istú dvojicu špiónov.)

Môžete predpokladať, že vstup popisuje riedku komunikačnú sieť spĺňajúcu podmienku zo zadania. Špeciálne teda platí $0 \leq m \leq 3n$.

Formát výstupu

Na výstup vypíšte jediný riadok a v ňom niekoľko navzájom rôznych čísel z rozsahu od 1 po n: čísla špiónov tvoriacich najväčšiu kliku.

Čísla môžete vypísať v ľubovoľnom poradí. Ak existuje viacero najväčších klík, stačí nájsť a vypísať ľubovoľnú jednu z nich.

Hodnotenie

Vo vstupoch, za ktoré môžete získať 2 body, bude platiť $1 \leq n \leq 40$.

Ďalších 5 bodov môžete získať za vstupy, v ktorých platí: $1 \leq n \leq 1000$ a zároveň celkový počet klík špiónov (všetkých, nie len maximálnych) neprevýši 64 000.

Vo všetkých vstupoch platí $1 \leq n \leq 100\,000$.

Príklad

vstup

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

výstup

```
1 2 3
```

Iným správnym výstupom by bolo „4 1 2“.
Výstup „1 2 3 4“ nie je správny, lebo špióni 3 a 4 spolu nekomunikujú, táto skupina teda nie je klikou.
Výstup „1 5“ nie je správny, lebo to síce je klika, ale nie je najväčšia možná.

Odovzdávanie riešení

Toto je praktická úloha. Pomocou webového rozhrania odovzdajte **funkčný, odladený program**.
Časový limit pre túto úlohu sú 2 sekundy, pamäťový limit je 256 MB.

A-I-2 Rozvod elektriny

Za siedmimi horami a za siedmimi dolinami vyšla z domčeka ježibaba a povedala: „Prečo práve ja musím bývať tak ďaleko? Ani elektrinu sem ešte nedotiahli!“

Po tom, ako sa ježibaba sťažovala na krajskom úrade, rozhodli sa radní, že je načase zaviesť do celého kraja elektrinu (skôr, ako sa niekto z nich čisto náhodou premení na ropuchu). V prvej fáze výstavby rýchlo postavili v kraji niekoľko elektrární. Potom prišla druhá fáza: V kraji bolo n lokalít (miest a elektrární), ktoré potrebovali pospájať elektrickým vedením do rozvodnej siete.

Toto spájanie spravili systematicky, aby sa na nič nezabudlo. Začali tým, že ku krajskému mestu priamym vedením pripojili jednu elektráreň. Takto dostali základ rozvodnej siete, ktorú následne rozširovali. Vždy si vybrali jednu lokalitu, ktorá ešte nebola pripojená, a postavili vedenie medzi ňou a jednou z lokalít, ktoré už pripojené boli. Takto vznikla rozvodná sieť so stromovou topológiou. V niektorých uzloch tejto siete boli mestá, v ostatných zas elektrárne.

Súťažná úloha

Ukázalo sa, že sieť treba zase rozpojiť na niekoľko častí, pričom v každej bude niekoľko miest a práve jedna elektráreň. Totiž ako tak narýchlo stavali elektrárne, stalo sa, že každá z nich produkuje striedavý prúd s inou frekvenciou. Preto si každé mesto musí vybrať práve jednu elektráreň, z ktorej bude brať elektrinu. Vašou úlohou bude nájsť toto priradenie miest k elektrárňam. Presnejšie, treba dodržať nasledovné podmienky:

- Jedným vedením sa nedá prenášať elektrickú energiu z dvoch rôznych elektrární.
- Nevieme cez mesto pripojené k jednej elektrárni preniesť elektrinu z inej elektrárne.
  Taktiež nevieme cez jednu elektráreň preniesť elektrinu z inej elektrárne.
- Každá elektráreň má svoj výkon – množstvo energie, ktoré vie za deň vyrobiť.
  A naopak, každé mesto má svoju spotrebu – množstvo energie, ktorú za deň spotrebuje.
- Každé vedenie má svoju kapacitu – maximálne množstvo energie, ktorú ním za deň vieme preniesť.

Formát vstupu

V prvom riadku sú celé čísla n a d_1. Číslo n je počet lokalít (miest a elektrární dokopy). Lokality sú očíslované od 1 po n v poradí, v akom boli pripojené k rozvodnej sieti. Číslo 1 má teda krajské mesto a číslo 2 nejaká elektráreň. Číslo d_1 je spotreba krajského mesta.

Nasledujúce riadky popisujú ostatné lokality. Presnejšie, i-ty riadok vstupu (pre $2 \leq i \leq n$) popisuje lokalitu i. Popis má tvar „z_i m_i c_i d_i“. Znak z_i je buď „`M`“ alebo „`E`“, podľa toho, či ide o mesto alebo o elektráreň. Nasledujú tri celé čísla: m_i ($1 \leq m_i < i$) je číslo lokality, ku ktorej sme lokalitu i pripojili, keď sme budovali sieť. Kapacita vedenia medzi lokalitami i a m_i je c_i. No a hodnota d_i má dva možné významy: ak ide o mesto, d_i je jeho spotreba, ak ide o elektráreň, d_i je jej výkon.

Môžete predpokladať, že platí $2 \leq n$ a tiež, že pre všetky i platí $0 \leq c_i, d_i \leq 10^9$.

Hodnotenie

Vo všetkých testovacích vstupoch bude platiť $n \leq 1\,000\,000$.
V testovacích vstupoch za 5 bodov bude platiť $n \leq 1\,000$.

Formát výstupu

Ak nie je možné priradiť mestám elektrárne tak, aby boli splnené všetky požiadavky, vypíšte jeden riadok a v ňom text „`nemozne`“. V opačnom prípade vypíšte jeden riadok a v ňom n čísel: $a_1, \ldots, a_n$. Ak je i-ta lokalita elektráreň, má byť $a_i = i$. Ak je to mesto, a_i má byť číslo elektrárne, z ktorej bude mesto i brať elektrinu.

Ak existuje viacero možných riešení, vypíšte ľubovoľné z nich.

(Nezabudnite: Pre každú elektráreň musí jej výkon byť väčší alebo rovný ako súčet spotreby miest, ktoré sú ku nej pripojené. Pre každé mesto musí platiť, že všetky mestá, ktoré ležia na ceste medzi ním a elektrárňou, ku

ktorej je pripojené, sú pripojené k tej istej elektrárni. Pre každé vedenie, ktoré spája dve lokality pripojené k tej istej elektrárni, musí platiť: súčet spotrieb miest, ktoré by boli prerušením tohto vedenia oddelené od elektrárne, nesmie presiahnuť kapacitu dotyčného vedenia.)

Príklad

vstup

```
5 5
E 1 5 10
E 1 5 20
M 3 15 10
M 4 5 5
```

výstup

```
2 2 3 3 3
```

vstup

```
5 5
E 1 5 10
E 1 5 20
M 3 12 10
M 4 5 5
```

výstup

```
nemozne
```

Odovzdávanie riešení

Toto je praktická úloha. Pomocou webového rozhrania odovzdajte **funkčný, odladený program**.
Časový limit pre túto úlohu sú 2 sekundy, pamäťový limit je 256 MB.

A-I-3 Húsenice

Vo Vihorlatskom pralese sa nedávno rozmnožili nebezpečné húsenice. Ohryzávajú stromy a hrozí, že onedlho bude z pralesa už len obyčajný les. Ako to však často býva, príroda si poradí aj sama. V pralese totiž žije škorec Ignác, ktorý práve vyráža na lov. Keď sa vydá na lov, letí po polpriamke zo svojho hniezda až na kraj pralesa a zožerie všetky húsenice, ponad ktoré preletí.

Súťažná úloha

Pre jednoduchosť si prales predstavíme ako vodorovnú rovinu a jednotlivé húsenice ako úsečky v tejto rovine. Ignác má hniezdo v bode $(0, 0)$. Nájdite polpriamku, po ktorej má Ignác letieť, ak chce húseníc uloviť najviac.

Formát vstupu

V prvom riadku je počet úsečiek n. Nasleduje n riadkov, každý z nich popisuje jednu úsečku. V i-tom z týchto riadkov sú štyri celé čísla $x_{i,1}$, $y_{i,1}$, $x_{i,2}$ a $y_{i,2}$. Tieto určujú úsečku s koncovými bodmi $(x_{i,1}, y_{i,1})$ a $(x_{i,2}, y_{i,2})$.

Úsečky sa môžu navzájom pretínať aj prekrývať. Taktiež je možné, že niektorými bodmi roviny prechádzajú viac ako dve úsečky. Niektoré súradnice koncových bodov úsečiek môžu byť aj záporné.

Môžete ale predpokladať, že pre každú úsečku platí podmienka $x_{i,1}y_{i,2} \neq x_{i,2}y_{i,1}$. (Z tejto podmienky vyplýva, že každá úsečka má kladnú dĺžku a tiež že žiadna úsečka neprechádza hniezdom škorca Ignáca).

Formát výstupu

Vypíšte jeden riadok a v ňom dve celé čísla x a y. Tieto čísla nesmú byť obe nulové. Tieto čísla sú súradnice bodu, ktorým prechádza polpriamka, po ktorej má škorec Ignác letieť. Počet húseníc, ktoré pretne táto polpriamka, musí byť maximálny možný. Ak je viacero optimálnych riešení, vypíšte ľubovoľné z nich.

Hodnotenie

Plných 10 bodov dostanete za riešenie, ktoré dokáže efektívne vyriešiť vstup obsahujúci stotisíce húseníc. Až 7 bodov môžete získať za riešenie, ktoré dokáže efektívne vyriešiť vstup s tisícmi húseníc.

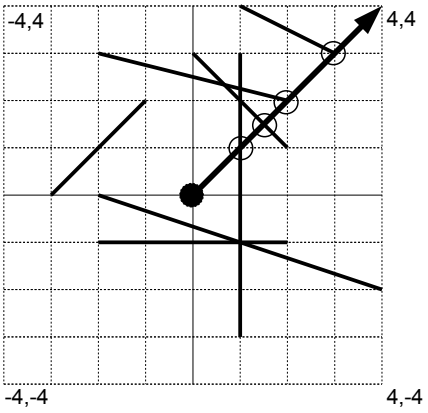
Príklad

vstup

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

output

```
4 4
```



Existuje viacero správnych riešení, okrem bodu $(4, 4)$ *vyhovujú napr. aj body* $(10, 10)$ *a* $(3, 4)$. *Vo všetkých týchto prípadoch Ignác uloví 4 húsenice.*

Odovzdávanie riešení

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

A-I-4 Log-space výpočty

V tomto ročníku olympiády sa budeme v každom súťažnom kole stretávať s programami, ktoré môžu používať len veľmi malé množstvo pamäte. V študijnom texte uvedenom za zadaním tejto úlohy sú popísané obmedzenia, ktoré musíte pri riešení tejto úlohy dodržať.

Súťažná úloha

Pri nasledujúcich úlohách nám vôbec nezáleží na časovej zložitosti vašich programov. Nemusíte ju ani odhadovať. Každý korektný program, spĺňajúci požiadavky uvedené v študijnom texte, dostane plný počet bodov.

a) (4 body) Na vstupe je číslo n a pole $A[1..n]$ obsahujúce postupnosť čísel. Napíšte log-space program, ktorý do výstupného poľa $B[1..n]$ vyplní túto postupnosť usporiadanú od najmenšieho čísla po najväčšie. (Pozor, niektoré čísla sa v poli A môžu opakovať!)

   Teda napr. pre vstupné pole $A = (3, 1, 4, 1, 5)$ máte vyrobiť výstupné pole $B = (1, 1, 3, 4, 5)$.

b) (4 body) Na vstupe je číslo n a dve polia $A[1..n]$ a $B[1..n]$ obsahujúce postupnosti čísel. Napíšte log-space program, ktorý zistí, či sa tieto postupnosti líšia len poradím prvkov.

   Teda napr. pre polia $A = (3, 1, 4, 1, 5)$ a $B = (1, 4, 5, 1, 3)$ je odpoveď „áno“, pre $A = (1, 2, 2)$ a $B = (2, 1, 1)$ je odpoveď „nie“.

c) (2 body) Majme dva log-space programy $\mathcal{F}$ a $\mathcal{G}$. Program $\mathcal{F}$ dostane vstup v poli $A[1..n]$ a vyrobí z neho výstupné pole $B[1..n]$. Program $\mathcal{G}$ dostane na vstupe pole B, ktoré vyrobil program $\mathcal{F}$, a svoj výstup zapíše do výstupného poľa $C[1..n]$.

   Dokážte, že existuje log-space program $\mathcal{H}$, ktorý na vstupe dostane pole A a na výstupe vyrobí zodpovedajúce pole C.

   (V čom je problém? Keby nám nezáležalo na pamäťovej zložitosti, mohli by sme najskôr ako podprogram spustiť $\mathcal{F}$, vypočítať si B, a potom spustiť $\mathcal{G}$ a vypočítať tak C. Problém je v tom, že na takéto riešenie nemáme dosť pamäte – nezmestí sa nám do nej pole B.)

Odovzdávanie riešení

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

Študijný text

Ak poznáme viac algoritmov, ktoré riešia tú istú úlohu, väčšinou za lepší považujeme ten, ktorý má menšiu časovú zložitosť. Pamäťovú zložitosť zväčša používame len ako dodatočné kritérium (s výnimkou situácií, kedy sú pamäťové nároky algoritmu absurdne vysoké). V úlohách, ku ktorým patrí tento študijný text, bude situácia presne opačná: zaujímať nás bude takmer výlučne pamäťová zložitosť programu. Tá bude musieť byť veľmi malá.

Presnejšie, budeme písať *log-space programy*. Pôjde o obyčajné programy vo vašom obľúbenom bežnom programovacom jazyku, len budú navyše musieť spĺňať nasledujúce obmedzenia:

- Každá použitá premenná musí byť jednoduchého *celočíselného* typu (napr. `int` v C++ či `longint` v Pascale).

- U celočíselných premenných nám nebude záležať na presnom rozsahu, nemusí vás teda napr. trápiť, či sú 32-bitové alebo 64-bitové. Namiesto toho si povolený rozsah hodnôt, ktoré sa do premenných zmestia, definujeme nasledovne: Nech n je veľkosť vstupu (teda napr. počet čísel na vstupe). Potom do premenných môžeme ukladať len hodnoty ktoré sú *polynomiálne veľké* v závislosti od n.

  Môžeme mať teda napr. premennú, ktorá bude nadobúdať hodnoty $-n \ldots n$, hodnoty $-3n^5 \ldots 3n^5$, čí len hodnoty $-4 \ldots 7$. Nesmieme však už mať premennú nadobúdajúcu napr. hodnoty $0 \ldots 2^n$.
- Pre pohodlie budeme aj typy `char` a `boolean` (resp. `bool` v C++) považovať za celočíselné, a teda povolené.
- Žiadne iné typy premenných (teda napr. ani polia alebo ukazovatele) nie sú povolené.
- Výnimku z predchádzajúcich pravidiel tvoria vstup a výstup. Vstup programu bude dostupný v nejakých špeciálnych premenných (zväčša to budú polia), ktoré môže váš program *len čítať*. A podobne výstup bude váš program ukladať do iných špeciálnych premenných, do ktorých smie *len zapisovať*.

  (Špeciálne upozorňujeme, že nesmiete výstupnú premennú zväčšiť o danú hodnotu. Teda na ňu nesmiete napr. v Pascale použiť príkaz `inc`, v C++ operátor `++` či `+=`. Všetky tieto zmeny totiž vyžadujú nie len zápis novej hodnoty, ale najskôr aj prečítanie starej – lenže výstupnú premennú čítať nesmiete.)

  Čísla na vstupe vždy budú mať veľkosť polynomiálnu od veľkosti vstupu, a teda každé z nich sa zmestí do pracovnej premennej.
- Vaše programy nesmú používať rekurziu.

Príklad 1. Ukážeme si log-space program, ktorý nájde maximum v poli čísel.

Listing programu (Pascal)

```
var n: integer;                  { vstupná premenná: počet čísel }
    A: array [1..n] of integer;  { vstupná premenná: pole čísel }
    m: integer;                  { výstupná premenná: pozícia maxima }
    i, j: integer;               { pracovné premenné }
begin
    j := 1;
    for i := 2 to n do
        if A[i] > A[j] then j := i;
    m := j;
end;
```

Jediné dve premenné sú `i` a `j` a zjavne nadobúdajú len hodnoty z rozsahu $1 \ldots n$. Tým sú teda splnené všetky potrebné podmienky a teda skutočne ide o log-space program.

Príklad 2. Nasledujúci log-space program nájde v poli čísel to, ktoré sa tam vyskytuje najčastejšie.

Listing programu (Pascal)

```
var n: integer;                  { vstupná premenná: počet čísel }
    A: array [1..n] of integer;  { vstupná premenná: pole čísel }
    m: integer;                  { výstupná premenná: jedna pozícia najčastejšieho }
    i, j, c, cmax: integer;      { pracovné premenné }
begin
    cmax := 0;
    for i := 1 to n do begin
        { spočítame, koľkokrát sa vyskytuje A[i] }
        c := 0;
        for j := 1 to n do
            if A[j] = A[i] then c := c+1;
        { je to viac ako doterajšie maximum? }
        if c > cmax then begin
            cmax := c;
            m := c;
        end;
    end;
end;
```

Opäť si ľahko rozmyslíme, že hodnoty pracovných premenných nikdy neprekročia n. Taktiež všetky ostatné požiadavky na log-space program sú dodržané.

Prečo práve takéto programy?

Isto ste si už položili otázku, prečo takéto programy nazývame log-space a načo je vlastne dobré sa nimi zaoberať.

Najpresnejší spôsob merania pamäťovej zložitosti daného programu dostaneme vtedy, keď zistíme počet *bitov pamäte*, ktorú potrebuje v závislosti od veľkosti vstupu.

Existuje síce pár veľmi jednoduchých problémov, ktoré vieme riešiť napríklad len s troma bitmi pamäte, veľa ich ale nie je a nebudú nás príliš zaujímať. My sa sústredíme na programy, ktoré vstupné dáta dostanú v nejakom n-prvkovom poli. Na to, aby sme vôbec mohli k prvkom takéhoto poľa rozumne pristupovať, potrebujeme do neho vedieť *indexovať*. A na to treba mať premennú, ktorá môže nadobudnúť n rôznych hodnôt.

Jeden bit pamäte má dva rôzne stavy: 0 alebo 1. Ak máme k-bitovú premennú, tá môže nadobúdať 2^k rôznych stavov – nezávisle pre každý z k bitov máme dve možnosti. Tieto stavy si my potom rôzne interpretujeme: raz ako znaky, inokedy ako čísla, a podobne. Napríklad taká 8-bitová premenná môže mať 256 rôznych stavov. Niekedy tieto stavy môžeme považovať za čísla od 0 do 255, inokedy za čísla od -100 do 155, a ešte inokedy za znaky v 8-bitovom ASCII kódovaní.

A tú istú úvahu môžeme spraviť aj opačne. Ak potrebujeme premennú, ktorá vie mať n rôznych hodnôt, potrebujeme, aby pre jej počet bitov k platila nerovnosť $2^k \geq n$. Inými slovami, najmenšie vyhovujúce k je $\lceil \log_2 n \rceil$ (t.j. horná celá časť dvojkového logaritmu čísla n).

Dostávame teda nasledovný záver: na prácu s n-prvkovým poľom určite treba aspoň rádovo $\log_2 n$ bitov pamäte. Toto je teda v istom zmysle najmenšia prakticky zaujímavá pamäťová zložitosť programov.

No a práve takúto (asymptotickú) pamäťovú zložitosť budú mať aj všetky programy, ktoré budete písať vo svojich riešeniach. Totiž naše obmedzenie na počet a veľkosť premenných, ktoré smiete používať, zaručuje, že celkový počet bitov pamäte, ktoré použijete, bude nanajvýš rádovo logaritmický od n. Odtiaľ teda pochádza názov „log-space“ – sú to programy, ktoré používajú $O(\log n)$ bitov pamäte.

(Príklad: Ak použijete 3 premenné, z ktorých každá môže nadobúdať hodnoty od 1 po n^5, bude váš program používať $3\lceil \log_2 n^5 \rceil \approx 15 \log_2 n$ bitov pamäte.)

O zákaze používania polí a rekurzie

Teoreticky by sa do logaritmického počtu bitov pamäte nejaké maličké polia zmestili – ale chcelo by to, aby súčet veľkostí všetkých prvkov bol nanajvýš logaritmický. Teda napríklad by sme mohli mať pole obsahujúce $2 \log n$ čísel z rozsahu 0 až 7. Alebo pole obsahujúce 3 čísla z rozsahu 1 až n^2. Nič z toho vám zrejme pri súťažných úlohách nepomôže, preto sme pre jednoduchosť polia zakázali úplne. (Namiesto poľa obsahujúceho 3 čísla predsa vždy môžete použiť 3 vhodne pomenované premenné.)

Druhou konštrukciou, ktorú sme museli zakázať, je rekurzia. Treba si totiž uvedomiť, že pamäť počas behu programu nepoužívame len na premenné. Vždy, keď v programe zavoláme nejakú funkciu, musí sa vyrobiť (na tzv. zásobníku) záznam, kde si okrem iného program zapamätá správnu návratovú adresu a tiež hodnoty parametrov, s ktorými sme danú funkciu zavolali. Taktiež lokálne premenné pre danú funkciu niekam treba uložiť. A tu práve rekurzia začína robiť problémy.

Totiž ak máme program pozostávajúci z viacerých funkcií, ale bez použitia rekurzie (t.j. každá funkcia volá len tie, ktoré boli v programe uvedené pred ňou), vždy ho vieme prerobiť („dosadením“ celého tela funkcie namiesto každého jej volania) na program, ktorý počíta to isté, ale už nepoužíva funkčné volania. Môžete si rozmyslieť, že sa pri tejto zmene nijak výrazne nezmenia pamäťové nároky programu.

Akonáhle však povolíme rekurziu (funkciu, ktorá volá sama seba, prípadne viacero funkcií, ktoré sa vedia volať navzájom), mohlo by sa stať, že pri behu programu nám bude postupne vznikať viac a viac lokálnych premenných a celková pamäť ľahko narastie nad logaritmickú. Z tohto dôvodu radšej rekurziu úplne zakazujeme. A rovnako ako pri poliach, aj tu by malo platiť, že vám toto obmedzenie nijak zásadne nesťaží riešenie súťažných úloh.

DVADSIATY ÔSMY ROČNÍK OLYMPIÁDY V INFORMATIKE

Príprava úloh: Michal Anderle, Vladimír Boža, Peter Fulla, Ján Hozza, Marek Špano
Recenzia: Michal Forišek
Slovenská komisia Olympiády v informatike
Vydal: IUVENTA – Slovenský inštitút mládeže, Bratislava 2012