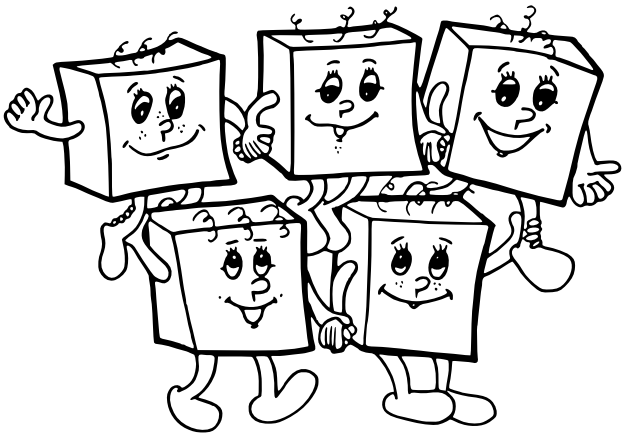
OLYMPIÁDA V INFORMATIKE

NA STREDNÝCH ŠKOLÁCH

http://oi.sk/



dvadsiaty piaty ročník
školský rok 2009/10

zadania krajského kola
kategória A

Priebeh krajského kola

Krajské kolo 25. ročníka Olympiády v informatike, kategória A, sa koná 12. januára 2010 v dopoludňajších hodinách. Na riešenie úloh majú súťažiaci 4 hodiny čistého času. Rôzne úlohy riešia súťažiaci na samostatné listy papiera. Akékoľvek pomôcky okrem písacích potrieb (napr. knihy, výpisy programov, kalkulačky) sú zakázané.

Čo má obsahovať riešenie úlohy?

- Slovne popíšte algoritmus.
  Slovný popis riešenia musí byť jasný a zrozumiteľný i bez nahliadnutia do samotného algoritmu/programu.
- Zdôvodnite správnosť vášho algoritmu.
- Uveďte a zdôvodnite jeho časovú a pamäťovú zložitosť.
- Podrobne uveďte dôležité časti algoritmu, ideálne vo forme programu v Pascale alebo C/C++.
- V prípade, že používate vo svojom programovacom jazyku knižnice, ktoré obsahujú implementované dátové štruktúry a algoritmy (napr. STL pre C++), v popise algoritmu stručne vysvetlite, ako by ste napísali program s rovnakou časovou zložitosťou bez použitia knižnice.

Hodnotenie riešení

Za každú úlohu môžete získať od 0 do 10 bodov.

Pokiaľ nie je v zadaní povedané ináč, základným kritériom hodnotenia riešenia je **správnosť** a **efektívnosť** navrhnutého algoritmu. Hodnotí sa aj kvalita popisu riešenia a zdôvodnenie tvrdení o jeho správnosti a efektívnosti.

Efektívnosť algoritmu posudzujeme vypočítaním jeho časovej zložitosti – funkcie, ktorá hovorí, ako dlho vykonanie algoritmu trvá v závislosti od veľkosti vstupných parametrov.

V zadaní úlohy môžu byť uvedené limity na veľkosť premenných. Tieto môžete použiť na odhad toho, ako dobré vaše riešenie je. Na počítači, ktorý vykoná miliardu inštrukcií za sekundu, zrieši vzorové riešenie ľubovoľný povolený vstup do niekoľko sekúnd.

A-II-1 Aquapark

Manažment aquaparku sa rozhodol zistiť, ako veľmi sú využívané tobogany. Konkrétne v tomto aquaparku sú 3 tobogany. Manažéri majú k dispozícii tieto informácie:

- Ako dlho trvá jazda na každom tobogane (T_1, T_2, T_3)
- Ako dlho trvá dostať sa od koncov toboganov k začiatkom (D).
  Tobogany sú umiestnené tak, že od hocijakého konca k hocijakému začiatku to trvá rovnako dlho.
- Fotobunkou zaznamenané časy, kedy niekto nasadol na daný tobogan (a_{ij}).

Z týchto informácií sa presný počet ľudí využívajúcich tobogany nemusí dať určiť. Vždy je ale jednoznačne určený *minimálny* počet ľudí, ktorí mohli tobogany využívať tak, aby to zodpovedalo zaznamenaným údajom.

Súťažná úloha

Napíšte program, ktorý z daných informácií o dĺžke jazdy na toboganoch, trvaní cesty nahor a časoch jednotlivých spustení určí minimálny možný počet ľudí, ktorí mohli využívať tobogany.

Inými slovami, nájdite najmenšie číslo K také, že existuje rozvrh pre K ľudí, pri ktorom by sa na každom z toboganov v každom zo zaznamenaných časov niekto spustil. Nezabudnite, že keď niekto nasadne na tobogan i v čase T, tak ďalšiu jazdu (na hociktorom tobogane) môže začať najskôr v čase $T + T_i + D$.
Nezabudnite uviesť dôkaz správnosti vášho algoritmu.

Formát vstupu

V prvom riadku sú 3 čísla T_1, T_2, T_3: dĺžky trvania jázd na jednotlivých toboganoch. V ďalšom riadku je jedno číslo D: čas, ktorý trvá výstup nahor. Potom nasledujú 3 riadky. Riadok $i + 2$ začína číslom N_i, udávajúcim počet jázd, ktoré sa na tobogane i uskutočnili. Potom nasleduje N_i čísel a_{ij} ($1 \leq j \leq N_i$), ktoré vyjadrujú časy nástupov na tento tobogan. Pre každý tobogan je táto postupnosť časov usporiadaná vzostupne.

Platí: $0 \leq N_1, N_2, N_3 \leq 1\,000\,000$, $1 \leq T_1, T_2, T_3, D, a_{ij} \leq 500\,000\,000$.

Formát výstupu

Vypíšte jedno číslo: minimálny počet ľudí, pre ktorý mohla zaznamenaná situácia nastať.

Príklady

vstup

```
1 2 3
1
2 1 7
3 2 5 11
1 3
```

výstup

```
2
```

Jeden možný spôsob, ako mohli dvaja ľudia spraviť uvedené jazdy: prvý sa spustil na prvom, treťom, a opäť prvom tobogane (v časoch 1, 3 a 7), zatiaľ čo druhý spravil všetky tri jazdy na druhom tobogane.

vstup

```
4 5 6
10
2 2 3
3 1 7 15
1 5
```

výstup

```
6
```

V tomto prípade žiaden návštevník nemohol stihnúť dve jazdy, preto určite bolo šesť návštevníkov.

A-II-2 Oplotenie farmy

Maroško sa dopočul, že v poľnohospodárstve sa točia veľké peniaze. Preto sa rozhodol, že v ňom začne podnikať. Netrvalo dlho a už vlastnil nádhernú veľkú farmu, na ktorej sa pestuje množstvo zaujímavých plodín. Maroškova poľnohospodárska pôda má obdĺžnikový tvar a je rozdelená na $R \times S$ rovnako veľkých štvorcových políčok. Na každom políčku je zasadená jedna poľnohospodárska plodina.

Nedávno sa farma ocitla v nebezpečí, pretože v okolí sa premnožili niektoré zvieratá, ktoré si radi pochutia na rastlinách na Maroškovej farme. Preto sa Maroško rozhodol, že postaví na farme plot, ktorý ochráni plody jeho práce.

Keďže v okolí nie je veľká konkurencia v oblasti stavebníctva, podarilo sa mu zohnať len jeden kontakt: firmu Štvorce s.r.o., ktorá sa špecializuje na stavebné práce štvorcového charakteru. Firma Štvorce s.r.o. sa ponúkla, že postaví na farme plot, ktorý ochráni štvorcový úsek pozemku.

Maroško teraz rozmýšľa, kde daný plot postaviť. Plot môže ochrániť štvorcové územie ľubovoľnej veľkosti. Navyše musí viesť cez hranice medzi políčkami, takže každé políčko ochráni alebo celé, alebo vôbec. Ďalšia Maroškova požiadavka je, aby plot ochránil políčka s aspoň dvoma rôznymi plodinami. Chce mať totiž istotu, že na trhu neostane len s jedným typom produktu. Pomôžte Maroškovi sa aspoň trochu zorientovať a zistite, koľko možností na postavenie plotu má.

Súťažná úloha

Daný je počet riadkov R, počet stĺpcov S a počet plodín K. Platí $1 \leq R, S \leq 2\,500$, $1 \leq K \leq 1\,000\,000\,000$. Na každom z ďalších R riadkov je S čísel – popis, ktoré plodiny sú zasadené na jednotlivých políčkach. Plodiny sú očíslované od 0 do $K - 1$. Vypíšte jedno číslo – koľkými spôsobmi môže Maroško postaviť na farme plot, ktorý ohradí štvorcovú oblasť, na ktorej sa pestujú aspoň dve rôzne plodiny.

Príklad

vstup

```
3 6 10
1 0 0 0 1 7
2 0 0 0 7 7
3 0 0 0 7 7
```

výstup

```
8
```

Pre daný popis farmy máme 5 možností, ako postaviť plot okolo oblasti 2×2 a 3 možnosti, ako postaviť plot okolo oblasti 3×3.

A-II-3 Obmedzovač rýchlosti

Spoločnosť Expresná Pošta doručuje zásielky po celej Európe. V poslednom čase jej vodiči príliš často prehliadali dopravné značky, kvôli čomu niekoľkokrát dostali pokutu za vysokú rýchlosť. Riaditeľ spoločnosti preto rozhodol do každého auta zakúpiť obmedzovač rýchlosti. Ten funguje nasledovne: vodič si pred jazdou nastaví rýchlosť v a prístroj sa automaticky postará o to, že auto počas celej jazdy neprekročí rýchlosť v. Riaditeľ spoločnosti naviac vydal predpis, podľa ktorého si vodič musí nastaviť také obmedzenie rýchlosti, aby na trase, ktorou pôjde, neprekročil žiadnu maximálnu povolenú rýchlosť.

Súťažná úloha

Daná je cestná sieť, po ktorej jazdia vodiči spoločnosti Expresná Pošta. Táto cestná sieť obsahuje N miest, medzi ktorými vedie dokopy M rôznych ciest. Každá cesta spája práve dve mestá, pričom cesty sa mimo miest navzájom križujú len mimoúrovňovo (teda mimo mesta nie je možné odbočiť na inú cestu). Pre každú cestu poznáme jej dĺžku (v kilometroch) a maximálnu povolenú rýchlosť (v kilometroch za hodinu).

Pre danú dvojicu miest x a y môže existovať viacero spôsobov, ako sa po cestách dostať z x do y. Vašou úlohou je napísať program, ktorý pre **všetky možné** dvojice miest x a y určí minimálny čas potrebný na cestu z mesta x do mesta y pri použití obmedzovača rýchlosti a dodržaní riaditeľovho predpisu.

Formát vstupu

Prvý riadok obsahuje dve kladné celé čísla N, M ($1 \leq N \leq 100$, $1 \leq M \leq 5\,000$). Číslo N určuje počet miest. Mestá na vstupe sú očíslované číslami 1 až N. Číslo M určuje počet ciest medzi nimi.

Každý z nasledujúcich M riadkov popisuje jednu cestu a obsahuje štyri čísla i j d m, udávajúce, že cesta spájajúca mestá i a j má dĺžku d kilometrov a maximálnu povolenú rýchlosť m kilometrov za hodinu. Všetky cesty na vstupe sú obojsmerné. Medzi dvoma mestami môže byť postavených viacero ciest s rôznou dĺžkou, resp. maximálnou rýchlosťou.

Môžete predpokladať, že medzi každou dvojicou miest existuje aspoň jedna trasa (možno tvorená viacerými nadväzujúcimi cestami).

Všetky vzdialenosti a rýchlosti na vstupe sú uvedené s presnosťou na najviac 3 desatinné miesta. Pre každú vzdialenosť d na vstupe platí $1 \leq d \leq 10^6$. Pre každú rýchlosť m na vstupe platí $5 \leq m \leq 10^5$.

Formát výstupu

Výstup má obsahovať N riadkov, každý riadok obsahujúci N čísel. Číslo v i-tom riadku a j-tom stĺpci určuje minimálny čas (v hodinách) potrebný na jazdu medzi mestom i a j. Výsledok stačí uviesť s presnosťou na 3 desatinné miesta.

Pri práci s reálnymi číslami v počítači môžu vznikať zaokrúhľovacie chyby. Túto skutočnosť môžete vo svojom riešení ignorovať – ako keby všetky výpočty, ktoré potrebujete, boli presné.

Príklad

vstup

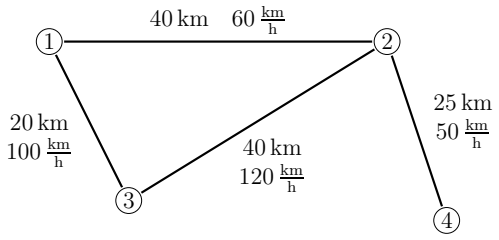
```
4 4
1 2 40.0 60.0
1 3 20.0 100.0
2 3 40.0 120.0
2 4 25.0 50.0
```

output

```
0.000 0.600 0.200 1.300
0.600 0.000 0.333 0.500
0.200 0.333 0.000 1.300
1.300 0.500 1.300 0.000
```

Najrýchlejší spôsob, ako sa dostať z mesta 1 do mesta 2 je cez mesto 3: pôjdeme 60 km rýchlosťou 100 km/h.

Všimnite si ale, že najrýchlejší spôsob, ako sa dostať z mesta 1 do mesta 4 je po trase 1-2-4, nie 1-3-2-4.



A-II-4 Počítač s gumenou rúrou

V tomto ročníku olympiády sa v teoretickej úlohe stretávame so špeciálnym počítačom zvaným Kvak. V študijnom texte uvedenom za zadaním tejto úlohy je popísané, ako tento počítač funguje. Študijný text je až na štylistické zmeny identický s tým z domáceho kola.

Súťažná úloha

a) (3 body)
Lucasove čísla sú definované nasledovne: $L_0 = 2$, $L_1 = 1$ a pre ľubovoľné $n \geq 2$ platí $L_n = L_{n-1} + L_{n-2}$. V rúre je jedno číslo n. Napíšte program pre Kvak, ktorý vypočíta a vypíše hodnotu (L_n mod 65 536).

b) (3 body)
V rúre je neprázdna postupnosť *kladných* čísel. Napíšte program pre Kvak, ktorý zistí, či sa v tejto postupnosti vyskytuje číslo 47 a podľa toho vypíše buď číslo 1 (ak áno) alebo číslo 0 (ak nie).

c) (4 body)
V rúre je neprázdna postupnosť *kladných* čísel. Napíšte program pre Kvak, ktorý na výstup vypíše všetky párne čísla v nej. (Na poradí, v akom ich vypíše, nezáleží, ale každé párne číslo musí vypísať práve toľkokrát, koľkokrát sa vyskytlo na vstupe.)

Tabuľka inštrukcii

príkaz	účinok príkazu
`get X`	Kvak vyberie číslo z rúry a uloží ho do registra `X`.
`put X`	Kvak vloží do rúry číslo z registra `X`.
`put` číslo	Kvak vloží dané číslo do rúry.
`print`	Kvak vyberie číslo z rúry a vypíše ho na výstup.
`add`	sčítanie: Kvak vyberie dve čísla z rúry a vloží tam ich súčet.
`sub`	odčítanie: Kvak vyberie dve čísla z rúry a vloží tam ich rozdiel (prvé mínus druhé).
`mul`	násobenie: Kvak vyberie dve čísla z rúry a vloží tam ich súčin.
`div`	delenie: Kvak vyberie dve čísla z rúry a vloží tam celú časť ich podielu (prvé lomeno druhé).
`mod`	zvyšok: Kvak vyberie dve čísla z rúry a vloží tam zvyšok, ktorý dá prvé z nich po delení druhým.
`label L`	návestie: Toto miesto v programe dostane meno `L` (kde `L` môže byť ľubovoľný reťazec).
`jump L`	skok: Kvak bude pokračovať vo vykonávaní programu od miesta, ktoré sa volá `L`.
`jz X L`	skok ak nula: Ak je v registri `X` nula, Kvak vykoná príkaz `jump L`.
`jeq X Y L`	skok ak sa rovnajú: Ak je v registroch `X` a `Y` to isté, Kvak vykoná príkaz `jump L`.
`jgt X Y L`	skok ak je väčšie: Ak je v registri `X` väčšia hodnota ako v `Y`, Kvak vykoná príkaz `jump L`.
`jempty L`	skok ak je rúra prázdna: Ak v rúre nie sú žiadne čísla, Kvak vykoná príkaz `jump L`.
`stop`	koniec: Kvak prestane vykonávať program.

Študijný text

Vedci z Kolégia Skúmania Potrubí (KSP) nedávno vyvinuli nový počítač, zvaný Kvak. Kvak má jedinú dátovú štruktúru: jednosmernú gumenú rúru.

Jediný dátový typ, ktorý Kvak pozná, sa volá `number`, a je to celé číslo z rozsahu od 0 po 65 535, vrátane.[1] Všetky matematické operácie počíta Kvak modulo 65 536. Teda napríklad hodnota výrazu 65530 + 10 je 4.

Kvak má 26 premenných, ktoré voláme registre. Tie sú označené písmenami `a` až `z` a v každom z nich môže byť uložená jedna hodnota typu `number`. Na začiatku výpočtu sú vo všetkých registroch nuly.

Okrem registrov má Kvak, ako sme už spomínali, jednu jednosmernú gumenú rúru. S tou vie robiť dve operácie: vložiť do nej číslo z registra `X` príkazom `put X` a z opačného konca rúry vybrať číslo a uložiť ho do registra `X` príkazom `get X`.

[1] $65\,535 = 2^{16} - 1$, typ `number` je teda 16-bitové celé číslo bez znamienka (to isté ako `word` v Pascale, `unsigned short` v C/C++).

Čísla sa v rúre samozrejme nemôžu predbiehať, Kvak ich teda bude vyberať v tom istom poradí, v akom ich tam vložil. Rúra je gumená, takže sa do nej čísel zmestí ľubovoľne veľa. Ak nie je povedané ináč, je na začiatku výpočtu rúra prázdna. Okrem rúry má Kvak ešte jeden kotúč žltej pásky, na ktorú môže písať svoj výstup.

Ak sa stane, že pri pokuse vybrať z rúry je rúra prázdna, nastane chyba. Rovnako nastane chyba, ak sa pokúsime deliť nulou, počítať zvyšok po delení nulou, alebo skočiť na neexistujúce miesto. Ak sa program dostane na koniec, Kvak korektne skončí (ako keby na konci programu bola ešte inštrukcia `stop`.)

Pre stručnosť môžeme písať viac príkazov do jedného riadku, v takomto prípade ich od seba treba oddeliť bodkočiarkou.

Príklad 1

Nasledujúci program spočíta a vypíše súčet čísel od 1 do 20.

```
put 20 ; put 0
label start
  get n ; jz n end
  put n ; put n ; put 1 ; add ; sub
  get x ; put x
jump start
label end
print
```

Vždy, keď sa Kvak pri vykonávaní programu dostane k riadku `label start`, budú v rúre práve dve čísla. Keď prvé z nich označíme n, hodnota druhého bude súčet $s = (n + 1) + \cdots + 20$. Následne načítame n do registra `n`. Ak je $n = 0$, máme v rúre hľadaný súčet, môžeme ho vypísať a skončiť. V opačnom prípade chceme spraviť dve veci: Prirátať toto n k doteraz získanému súčtu, a následne zmenšiť n o jedna. Po vykonaní príkazov `put n ; put n ; put 1` máme v rúre postupne čísla: s, n, n, 1. Príkaz `add` sčíta prvé dve, po jeho vykonaní je v rúre: n, 1, $n + s$. Po vykonaní ďalšieho príkazu `sub` máme v rúre hodnoty $n + s$ a $n - 1$. To už je skoro to, čo treba, len sú v opačnom poradí. Preto jednu z nich načítame do `x` a znovu vložíme.

Príklad 2

V rúre je neprázdna postupnosť čísel. Napíšeme program, ktorý spočíta a vypíše jej súčet. (Presnejšie, jeho zvyšok po delení 65 536.)

Dokola budeme opakovať nasledujúci postup: Zistíme, či sú v rúre aspoň dve čísla. Ak áno, tie dve, ktoré sú práve na začiatku, sčítame. Ak už zostalo len jedno, je zjavne súčtom všetkých pôvodných čísel. V programe pre Kvak môžeme túto myšlienku implementovať napríklad nasledovne:

```
label cyklus ; get a ; jempty koniec ; put a ; add ; jump cyklus
label koniec ; put a ; print
```

Na začiatku každej iterácie načítame do registra `a` číslo z rúry. Ak je tá v tomto okamihu prázdna, máme v registri `a` hľadaný súčet, stačí ho už len vypísať. Ak nie, číslo z registra `a` vrátime do rúry. V tomto okamihu sú v rúre aspoň dve čísla, a teda môžeme bez obáv zavolať inštrukciu `add`.

Časová zložitosť tohto riešenia je lineárna od počtu čísel, ktoré boli na začiatku v rúre. Totiž každá iterácia cyklu trvá len konštantne veľa krokov a zmenší nám o jedno počet čísel v rúre.

SLOVENSKÁ KOMISIA OLYMPIÁDY V INFORMATIKE

DVADSIATY PIATY ROČNÍK OLYMPIÁDY V INFORMATIKE

Zodpovedný redaktor: Michal Forišek
Sadzba programom LaTeX