59. ročník matematické olympiády na středních školách

Kategorie P

In: Zdeněk Dvořák (editor); Karel Horák (editor); Daniel Kráľ (editor); Peter Novotný (editor); Martin Panák (editor); Jaromír Šimša (editor); Jaroslav Švrček (editor); Pavel Töpfer (editor): 59. ročník matematické olympiády na středních školách. Zpráva o řešení úloh ze soutěže konané ve školním roce 2009/2010. 51. mezinárodní matematická olympiáda. 22. mezinárodní olympiáda v informatice. (Czech). Olomouc: Univerzita Palackého v Olomouci, 2012. pp. 101–120.

Persistent URL: http://dml.cz/dmlcz/405194

Texty úloh

P – I – 1

Malíř Bonifác

Radní v Kocourkově vypsali nedávno výběrové řízení na velmi odpovědnou a důležitou činnost: malování chodníku před radnicí. Ve výběrovém řízení zvítězil malíř Bonifác (jediný uchazeč a zcela náhodou také starostův bratr).

Jak už to bývá, sotva Bonifác podepsal smlouvu, hned začal dostávat z radnice jeden příkaz za druhým: Tento kus chodníku natřít zelenou barvou, tento růžovou, potom to skoro celé přetřít na bílo... Netrvalo dlouho a Bonifác si všiml, že se některé příkazy překrývají. A když si uvědomil, že ho smlouva zavazuje provést všechny příkazy v tom pořadí, v jakém je dostal, začaly ho obcházet mdloby.

Naštěstí však přišel na geniální nápad. Kdyby věděl, jak má chodník vypadat nakonec po provedení všech příkazů, mohl by ho tak namalovat rovnou a potom se tvářit, že on přece všechny příkazy dodržel. A hlavně potom radnici všechno vyúčtuje podle původních příkazů a ještě na tom pořádně vydělá.

Soutěžní úloha. Chodník před radnicí měří K kocourkovských kroků. Jeden jeho konec bude mít souřadnici 0, opačný konec má souřadnici K. V současnosti má celý chodník asfaltově černou barvu. Bonifác používá F jiných barev, očíslovaných od 1 do F. Postupně dostal N příkazů. Každý z nich zapíšeme ve tvaru ‚a_i b_i f_i', kde a_i a b_i jsou souřadnice začátku a konce úseku a f_i je barva, kterou se má tento úsek obarvit. Na obarvení jednoho kocourkovského kroku chodníku potřebuje Bonifác jeden litr barvy. Pro každou z barev spočítejte, kolik litrů bude Bonifác potřebovat.

Formát vstupu: Vstupní soubor se jmenuje `bonifac.in`. Na prvním řádku souboru jsou tři celá čísla N (počet příkazů), F (počet barev) a K (délka chodníku) oddělená mezerami ($1 \leqq N \leqq 100\,000$, $1 \leqq F, K \leqq$

$\leqq 1\,000\,000\,000$). Následuje N řádků, z nichž každý popisuje jeden příkaz, a to v pořadí, v jakém je Bonifác dostal. Přitom i-tý z těchto řádků obsahuje tři celá čísla a_i, b_i a f_i oddělená mezerami ($0 \leqq a_i < b_i \leqq K$, $1 \leqq f_i \leqq F$).

Pro 8 z 10 testovacích vstupů bude navíc platit $K \leqq 100\,000$. Pro 6 z těchto 8 testovacích vstupů bude navíc $N \leqq 1\,000$, a pro 3 z těchto 6 vstupů také $K \leqq 1\,000$.

Formát výstupu: Výstupní soubor se jmenuje `bonifac.out`. Pro každou z F barev (v pořadí jejich čísel) zapište do výstupního souboru jeden řádek obsahující jedno celé číslo — kolik litrů této barvy bude Bonifác potřebovat.

Příklad:

Vstupní soubor `bonifac.in`:	Výstupní soubor `bonifac.out`:
`4 5 7`	`1`
`1 5 1`	`3`
`2 4 3`	`1`
`4 6 4`	`0`
`3 6 2`	`0`

P – I – 2

Čokoláda

Mařenka bude mít brzy narozeniny. Její bratr Jeníček dlouho nemohl vymyslet, co by jí jenom mohl k narozeninám dát — až konečně ve své tajné skrýši na půdě objevil zbytek čokolády, kterou si tam kdysi ukryl. Pravda, myši už si vybraly svoji daň, ale i tak z čokolády zůstalo ještě docela dost. Děravé části oláme, aby mu vznikla pěkná čtvercová tabulka, a tu úhledně zabalí. A zbytek samozřejmě sní.

Soutěžní úloha. Je dán původní počet řádků R a sloupců S, které čokoláda kdysi měla. Dále máme matici $R \times S$ nul a jedniček určující, která políčka čokolády zůstala celá. Zjistěte, kolika různými způsoby může Jeníček uskutečnit svůj plán. Jinými slovy řečeno, spočítejte, kolika způsoby je možné ve zbytku čokolády vyznačit čtverec (libovolné velikosti) bez děr. Všechny hrany čtverce musí samozřejmě ležet na hranách políček. Stejně velké čtverce ležící na různých souřadnicích v tabulce čokolády považujeme za různá řešení.

Formát vstupu: Vstupní soubor se jmenuje `cokolada.in`. Na jeho prvním řádku jsou dvě celá celé čísla R a S oddělená mezerou ($1 \leqq R, S \leqq$

$\leqq 2\,500$). Následuje R řádků, v r-tém z nich je S mezerami oddělených celých čísel $a_{r,1}, \ldots, a_{r,S}$. Je-li políčko čokolády (r, s) celé, je $a_{r,s} = 1$, jinak $a_{r,s} = 0$.

Pro 7 z 10 testovacích vstupů bude navíc platit $R \leqq 500$. Pro 5 z těchto 7 testovacích vstupů bude $R, S \leqq 100$, a pro 3 z těchto 5 vstupů bude $R, S \leqq 20$.

Formát výstupu: Výstupní soubor `cokolada.out` obsahuje jediný řádek a na něm jedno celé číslo — hledaný počet čtverců.

Příklad

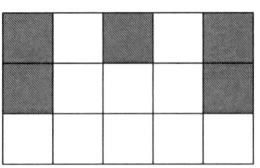
Vstupní soubor `cokolada.in`:

```
3 5
0 1 0 1 0
0 1 1 1 0
1 1 1 1 1
```

Výstupní soubor `cokolada.out`:

```
12
```

Na obrázku vpravo je nakreslena čokoláda popsaná ukázkovým vstupem. Šedou barvou jsou vyznačena políčka, která chybějí.



Čtverec 1×1 *na ní můžeme vyznačit deseti způsoby a čtverec* 2×2 *dvěma, což je celkem* $10 + 2 = 12$ *způsobů.*

P – I – 3

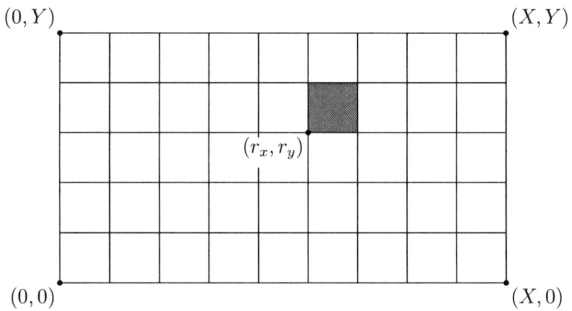
Koláč

Zlá ježibaba drží v kleci Jeníčka a Mařenku a snaží se je vykrmit. Právě pro ně upekla plech ježibabího koláče. Koláč má tvar obdélníka, celý je odpudivý a navíc je ozdoben ohavnou pečenou ropuchou.

Protože cokoliv je lepší než muset sníst tuto ropuchu, rozhodli se Jeníček s Mařenkou, že si z jedení koláče udělají hru. Mařenka na něm lžičkou nakreslila čáry, čímž ho rozdělila na $X \times Y$ stejných čtverců. Celá ropucha sedí na jednom z těchto čtverců.

Jeníček s Mařenkou se nyní budou pravidelně střídat na tahu. Ten z nich, kdo je na tahu, si vybere některou z vyznačených čar a podél ní koláč rozřízne na dvě obdélníkové části. Následně sní tu část koláče, ve které není ropucha. Kdo bude na tahu v okamžiku, kdy už z koláče zbude

pouze poslední čtverec s ropuchou, prohrál a musí ropuchu sníst. První tah provádí Mařenka.



Soutěžní úloha. Jsou dány rozměry koláče X, Y a souřadnice r_x, r_y levého dolního rohu čtverce, v němž je ropucha.

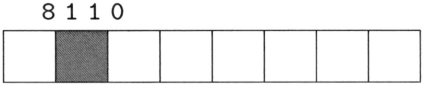
a) *(2 body)* Rozhodněte, kdo zvítězí v situaci znázorněné na obrázku — tedy pro $(X, Y) = (9, 5)$ a $(r_x, r_y) = (5, 3)$ — a popište jednu možnou strategii, která mu zabezpečí výhru.

b) *(8 bodů)* Popište co nejefektivnější algoritmus, který pro dané hodnoty X, Y, r_x a r_y zjistí, které z dětí hru vyhraje, jestliže budou obě hrát optimálně.

Příklady

Vstup:

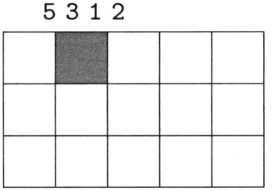
```
8 1 1 0
```



V prvním tahu Mařenka provede řez po přímce $x = 3$. Zbude ropucha a okolo ní z každé strany jeden čtverec. Jeníček sní jeden z nich, Mařenka druhý, a Jeníčkovi zůstane ropucha.

Vstup:

```
5 3 1 2
```



V druhém příkladu vyhraje Jeníček.

P – I – 4

Počítač Kvak

V letošním ročníku olympiády se budeme setkávat se speciálním počítačem nazvaným Kvak. Ve studijním textu uvedeném za zadáním této úlohy je popsáno, jak počítač Kvak funguje a jak se programuje.

Soutěžní úloha.

a) *(3 body)* V rouře počítače je jedno číslo. Napište program pro Kvak, který vypíše 1, jestliže je to prvočíslo, zatímco v opačném případě vypíše 0.

Plný počet bodů dostanete za libovolné řešení, které bude mít méně než 100 příkazů a pro libovolný vstup vykoná méně než 10 000 kroků.

b) *(4 body)* V rouře počítače je posloupnost *kladných* čísel. Délka této posloupnosti je menší než 65 000. Napište program pro Kvak, který tuto délku spočítá a vypíše. Plný počet bodů dostanete za řešení, které bude mít lineární časovou složitost.

c) *(3 body)* Počítač Kvak se nám poškodil, takže dokáže provést příkaz `put` pouze desetkrát a poté se definitivně zastaví. Všechny ostatní příkazy provádí počítač bez problémů.

V rouře počítače je neprázdná posloupnost čísel. Je možné napsat program pro takto poškozený Kvak, který bez ohledu na délku vstupní posloupnosti spočítá a vypíše její maximum? Jestliže ano, napište takový program. V opačném případě dokažte, že to není možné.

V letošním ročníku olympiády se budeme setkávat se speciálním počítačem zvaným Kvak.

Jediný datový typ, se kterým Kvak pracuje, se nazývá `number`, což je celé číslo z rozsahu od 0 do 65 535 včetně.[1] Všechny matematické výpočty provádí Kvak modulo 65 536, takže například hodnotou výrazu 65 530+10 je 4.

Kvak používá 26 proměnných, které nazýváme *registry*. Registry jsou označeny písmeny `a` až `z` a v každém z nich může být uložena jedna hodnota typu `number`. Na začátku výpočtu jsou ve všech registrech nuly.

Kromě registrů má Kvak ještě jednu *jednosměrnou rouru* neomezené délky, do které se mohu ukládat hodnoty typu `number`. Je to jediná datová struktura, kterou Kvak používá. S rourou lze provádět dvě operace:

▷ vložit do ní číslo z registru X příkazem `put` X,

[1] $65\,535 = 2^{16} - 1$, typ `number` je tedy přesně to, co znáte jako 16bitové celé číslo bez znaménka.

▷ z opačného konce roury odebrat číslo a uložit ho do registru X příkazem `get` X.

Čísla se v rouře počítače nemohou předbíhat, Kvak je tedy bude odebírat ve stejném pořadí, v jakém je do roury vložil.[2] Roura má neomezenou kapacitu, lze do ní vložit libovolné množství čísel. Není-li řečeno jinak, roura je na začátku výpočtu prázdná.

Počítač Kvak má také možnost vypisovat čísla (výsledky výpočtu) na výstup.

Příkazy

V následující tabulce jsou shrnuty všechny příkazy, které Kvak umí provádět a které tedy můžete používat v programech.

příkaz	*význam příkazu*
`get` X	Kvak odebere jedno číslo z roury a uloží ho do registru X.
`put` X	Kvak vloží do roury číslo z registru X.
`put` *číslo*	Kvak vloží dané číslo do roury.
`print`	Kvak odebere jedno číslo z roury a vypíše ho na výstup.
`add`	sčítání: Kvak odebere dvě čísla z roury a vloží do roury jejich součet.
`sub`	odčítání: Kvak odebere dvě čísla z roury a vloží do roury jejich rozdíl (první minus druhé).
`mul`	násobení: Kvak odebere dvě čísla z roury a vloží do roury jejich součin.
`div`	dělení: Kvak odebere dvě čísla z roury a vloží do roury celou část jejich podílu (první lomeno druhé).
`mod`	zbytek: Kvak odebere dvě čísla z roury a vloží do roury zbytek, který dá první z nich po celočíselném dělení druhým.
`label` L	návěstí: Toto místo v programu dostane označení L (kde L může být libovolný řetězec). Stejné návěstí nesmí být v programu vícekrát.
`jump` L	skok: Kvak bude pokračovat v provádění programu od místa, které má označení L.
`jz` X L	skok jestliže nula: Je-li v registru X nula, Kvak provede příkaz `jump` L.
`jeq` X Y L	skok jestliže se rovnají: Je-li v registrech X a Y stejná hodnota, Kvak provede příkaz `jump` L.

[2] Takovou datovou strukturu obvykle nazýváme *fronta*.

`jgt` X Y L skok jestliže je větší: Je-li v registru X větší hodnota než v registru Y, Kvak provede příkaz `jump` L.

`jempty` L skok jestliže je prázdná: Není-li v rouře žádné číslo, Kvak provede příkaz `jump` L.

`stop` konec: Kvak ukončí svůj výpočet.

Pokud se během výpočtu stane, že se pokusíme odebrat číslo z roury počítače a roura přitom bude prázdná, nastane chyba. Chyba nastane také tehdy, když se pokusíme dělit nulou, počítat zbytek po dělení nulou, nebo skočit na neexistující místo v programu. Dojde-li výpočet programu na konec, Kvak po provedení posledního příkazu korektně skončí (jako kdyby na konci programu byl ještě příkaz `stop`.)

V zápisu programu můžeme psát více příkazů na jeden řádek, v takovém případě je od sebe oddělujeme středníkem.

Příklad 1

Následující program spočítá a vypíše součet všech čísel od 1 do 20.

```
put 20
put 0
label start
get a
jz a
end
put a ; put a ; put 1
add
sub
get b ; put b
jump start
label end
print
```

Pokaždé, když se Kvak při provádění programu dostane ke třetímu řádku (`label start`), budou v rouře právě dvě čísla. Jestliže první z nich označíme N, hodnota druhého bude rovna součtu $S = (N+1)+\ldots+20$. Poté načteme N do registru `a`. Je-li $N = 0$, máme v rouře hledaný součet, můžeme ho vypsat na výstup a skončit. V opačném případě chceme provést dvě věci: Přičíst N k dosud získanému součtu, a následně N zmenšit o 1. Po provedení řádku šest (tři příkazy `put`) máme v rouře postupně čísla: S, N, N, 1. Příkaz `add` sečte první dvě, po jeho provedení bude v rouře trojice čísel N, 1, $N+S$. Po vykonání dalšího příkazu `sub` budou v rouře hodnoty $N+S$ a $N-1$. To už je téměř to, co potřebujeme, jenom

v opačném pořadí. Proto první z nich načteme do registru `b` a znovu vložíme do roury.

Příklad 2

V rouře je neprázdná posloupnost čísel. Napíšeme program, který spočítá a vypíše na výstup jejich součet. (Přesněji, jeho zbytek po dělení 65 536.)

Budeme stále opakovat následující postup: Zjistíme, zda jsou v rouře aspoň dvě čísla. Jestliže ano, některá dvě z nich sečteme a nahradíme je jejich součtem. Pokud tam už dvě čísla nejsou, zůstalo tam tady už jenom jediné a to zjevně součtem všech původních čísel. V programu pro počítač Kvak můžeme tuto myšlenku implementovat například následovně:

```
label cyklus
get a
jempty konec
put a
add
jump cyklus

label konec
put a
print
```

Na začátku každé iterace odebereme z roury jedno číslo a vložíme ho do registru `a`. Pokud se tím roura vyprázdnila, máme v registru `a` hledaný součet, stačí ho už jenom vypsat. Pokud ne, číslo z registru `a` vrátíme zpět do roury. V tom okamžiku jsou v rouře alespoň dvě čísla a můžeme tedy bez obav provést příkaz `add`.

Časová složitost tohoto řešení je lineární vzhledem k počtu čísel, která byla na začátku výpočtu v rouře. Každá iterace cyklu totiž provádí jen konstantní počet příkazů a zmenší nám o jedno počet čísel v rouře.

P – II – 1

Aquapark

V aquaparku mají tři tobogany. Správce aquaparku se rozhodl zjistit, nakolik je návštěvníci využívají. Má k dispozici následující informace:

▷ Jak dlouho trvá jedna jízda na každém toboganu (časy T_1, T_2, T_3).

- ▷ Jak dlouho trvá, než návštěvník dojde od konců toboganů k jejich začátkům (čas D). Tobogany jsou umístěny tak, že cesta od konce libovolného toboganu k začátku libovolného toboganu trvá stejně dlouho.
- ▷ Pro každý tobogan fotobuňkou zaznamenané časy, kdy některý z návštěvníků aquaparku nasedl na tento tobogan.

Z těchto informací se přesný počet návštěvníků využívajících tobogany většinou nedá určit. Můžeme ale určit minimální počet lidí, kteří mohli tobogany využívat tak, aby to odpovídalo zaznamenaným údajům.

Soutěžní úloha. Na základě informací o délce jízdy na toboganech, trvání cesty zpět nahoru a časech jednotlivých nasednutí na tobogany určete minimální možný počet lidí, kteří mohli využívat tobogany.

Jinými slovy, najděte nejmenší číslo K takové, že existuje rozvrh pohybu pro K lidí, podle kterého někdo nasedne v každém ze zaznamenaných časů na příslušný tobogan. Nezapomeňte, že když někdo nasedne na tobogan i v čase T, tak další jízdu (na kterémkoliv z toboganů) může tento člověk začít nejdříve v čase $T + T_i + D$.

Nezapomeňte také zdůvodnit správnost svého algoritmu.

Formát vstupu: Na prvním řádku vstupu jsou tři čísla T_1, T_2, T_3 — délky jízd na jednotlivých toboganech. Na druhém řádku je jedno číslo D — čas potřebný na výstup od konců toboganů nahoru k jejich začátkům. Následují tři řádky s informacemi o uskutečněných jízdách na jednotlivých toboganech. Každý z nich začíná číslem N_i, udávajícím počet jízd, které se na toboganu i uskutečnily. Na řádku pak následuje N_i čísel a_{ij} ($1 \leqq j \leqq N_i$), která představují časy nasednutí na tento tobogan. Pro každý tobogan je tato posloupnost časů uspořádána vzestupně.

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

Formát výstupu: Vypište jediné číslo — minimální počet návštěvníků aquaparku, pro něž mohla zaznamenaná situace nastat.

Příklady

Vstup:	*Výstup*:
1 2 3 1 2 1 7 3 2 5 11 1 3	2

Jeden možný způsob, jak mohli dva lidé uskutečnit všechny zaznamenané jízdy: první z nich jel na prvním, pak na třetím, a opět na prvním

toboganu (v časech 1, 3 a 7), zatímco druhý člověk absolvoval všechny tři jízdy na druhém toboganu.

Vstup:	*Výstup*:
4 5 6	6
10	
2 2 3	
3 1 7 15	
1 5	

V tomto případě žádný návštěvník nemohl stihnout dvě jízdy, proto jich určitě muselo být šest.

P – II – 2

Oplocení farmy

Přemysl se doslechl, že se v zemědělství točí velké peníze, a rozhodl se, že v něm také začne podnikat. Netrvalo dlouho a už vlastnil rozlehlou farmu, na níž pěstoval množství zajímavých plodin. Přemyslova zemědělská půda má obdélníkový tvar a je rozdělena na $R \times S$ stejně velkých čtvercových políček. Na každém políčku se pěstuje jedna plodina.

Nedávno se farma ocitla v nebezpečí, neboť v jejím okolí se přemnožili zajíci, kteří si rádi pochutnávají na pěstovaných rostlinách. Proto se Přemysl rozhodl, že na farmě nechá postavit plot, který ochrání plody jeho práce.

Jelikož v okolí není velká konkurence v oblasti stavebnictví, Přemyslovi se podařilo sehnat pouze jeden kontakt — firmu Čtverce s.r.o., která se specializuje na stavební práce čtvercového charakteru. Firma Čtverce s.r.o. nabídla, že postaví na farmě plot, který ochrání zvolenou čtvercovou část pozemku.

Soutěžní úloha. Přemysl přemýšlí, kde má nechat plot postavit. Plot může chránit čtvercové území libovolné velikosti. Musí ale vést po hranicích mezi políčky, takže každé políčko ochrání buď celé, nebo vůbec. Přemysl navíc požaduje, aby plot chránil políčka s alespoň dvěma různými plodinami. Chce mít totiž jistotu, že nezůstane na trhu jenom s jedním produktem. Pomozte Přemyslovi zjistit, kolik má možností na postavení plotu.

Formát vstupu: Na prvním řádku vstupu jsou zadány rozměry pozemku — počet řádků R a počet sloupců S, a dále počet plodin K, které je možné na farmě pěstovat. Platí $1 \leqq R, S \leqq 2\,500$, $1 \leqq K \leqq 1\,000\,000\,000$.

Na každém z dalších R řádků vstupu je vždy uvedeno S čísel — popis, které plodiny se pěstují na jednotlivých políčkách. Plodiny jsou označeny čísly od 0 do $K - 1$.

Formát výstupu: Vypište jediné číslo — kolika způsoby může Přemysl postavit na farmě plot, který ohradí čtvercovou oblast, na níž se pěstují aspoň dvě různé plodiny.

Příklad

Vstup:	*Výstup*:
3 6 10 1 0 0 0 1 7 2 0 0 0 7 7 3 0 0 0 7 7	8

Máme pět možností, jak lze postavit plot kolem oblasti velikosti 2×2, *a tři možnosti, jak lze postavit plot kolem oblasti velikosti* 3×3.

P – II – 3

Omezovač rychlosti

Společnost Expresní pošta doručuje zásilky po celé Evropě. V poslední době ale její řidiči často přehlíželi dopravní značky a dostávali pokuty za překročení maximální povolené rychlosti. Ředitel společnosti proto rozhodl, že nechá do každého auta namontovat omezovač rychlosti. Ten funguje následovně: řidič si na něm před jízdou nastaví maximální přípustnou rychlost v a přístroj se pak automaticky postará o to, aby auto během celé jízdy nikdy nepřekročilo tuto rychlost. Ředitel společnosti navíc vydal nařízení, že si řidič musí před každou jízdou nastavit takové omezení rychlosti, aby na trase, kterou pojede, nepřekročil žádnou maximální povolenou rychlost.

Soutěžní úloha. Je dána silniční síť, po níž jezdí řidiči společnosti Expresní pošta. Tato silniční síť obsahuje N měst, mezi nimiž vede celkem M různých cest. Každá cesta spojuje dvě města, přičemž cesty se mimo města kříží pouze mimoúrovňově (tzn. mimo města nelze odbočit z jedné cesty na jinou). Pro každou cestu známe její délku (v kilometrech) a maximální povolenou rychlost (v kilometrech za hodinu).

Pro danou dvojici měst x a y může existovat více způsobů, jak lze po cestách dojet z města x do města y. Napište program, který pro všechny dvojice měst x a y určí minimální čas potřebný na cestu z města x do

města y při použití omezovače rychlosti a dodržení nařízení ředitele společnosti.

Formát vstupu: První řádek obsahuje dvě kladná celá čísla N, M ($1 \leqq N \leqq 50$, $1 \leqq M \leqq 1\,000$) — počet měst a počet cest mezi nimi. Jednotlivá města jsou na vstupu označena čísly 1 až N.

Každý z následujících M řádků popisuje jednu cestu a obsahuje čtyři čísla i, j, d, m. Ta udávají, že cesta spojující města i a j má délku d kilometrů a maximální povolenou rychlost m kilometrů za hodinu. Všechny cesty jsou obousměrné. Mezi dvěma městy může být postaveno více cest různé délky a s různou maximální povolenou rychlostí.

Můžete předpokládat, že mezi každou dvojicí měst existuje aspoň jedna trasa (která může být tvořena více navazujícími cestami).

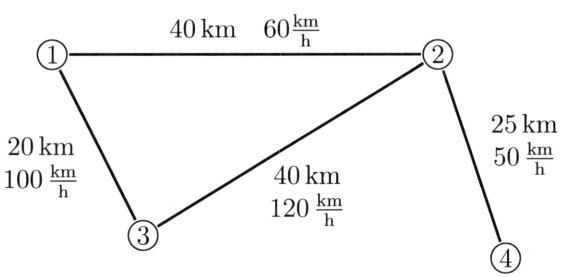
Všechny vzdálenosti a rychlosti jsou na vstupu uvedeny s přesností nejvýše na 3 desetinná místa. Pro každou vzdálenost d platí $1 \leqq d \leqq$ $\leqq 1\,000\,000$, pro každou maximální povolenou rychlost m platí $5 \leqq m \leqq$ $\leqq 100\,000$.

Formát výstupu: Výstup bude tvořen N řádky, z nichž každý obsahuje N čísel. Číslo v i-tém řádku a j-tém sloupci určuje minimální čas (v hodinách) potřebný na jízdu mezi městy i a j. Výsledek uveďte s přesností na tři desetinná místa.

Při práci s reálnými čísly v počítači mohou vznikat zaokrouhlovací chyby. Tuto skutečnost můžete ve svém řešení ignorovat — postupujte, jako kdyby všechny výpočty, které provádíte, byly přesné.

Příklad

Vstup:	*Výstup*:
`4 4`	`0.000 0.600 0.200 1.300`
`1 2 40.0 60.0`	`0.600 0.000 0.333 0.500`
`1 3 20.0 100.0`	`0.200 0.333 0.000 1.300`
`2 3 40.0 120.0`	`1.300 0.500 1.300 0.000`
`2 4 25.0 50.0`	



Z města 1 *do města* 2 *se nejrychleji dostaneme přes město* 3*: pojedeme celkem 60 kilometrů rychlostí 100 km/h. Nejrychlejší cesta z města* 1 *do města* 4 *ovšem vede po trase* 1–2–4*, nikoliv* 1–3–2–4.

P – II – 4

Počítač Kvak

K úloze se vztahuje studijní text z úlohy P–I–4.

Soutěžní úloha.

a) *(3 body)* Lucasova čísla jsou definována následovně: $L_0 = 2$, $L_1 = 1$ a pro každé $n \geqq 2$ platí $L_n = L_{n-1} + L_{n-2}$.

V rouře počítače je jedno číslo n. Napište program pro počítač Kvak, který spočítá a vypíše hodnotu (L_n mod 65 536).

b) *(3 body)* V rouře počítače je neprázdná posloupnost *kladných* čísel. Napište program pro počítač Kvak, který zjistí, zda se v této posloupnosti vyskytuje číslo 47, a podle toho vypíše buď číslo 1 (pokud ano), nebo číslo 0 (pokud tam není).

c) *(4 body)* V rouře počítače je neprázdná posloupnost *kladných* čísel. Napište program pro počítač Kvak, který na výstup vypíše všechna sudá čísla obsažená ve vstupní posloupnosti. Nezáleží přitom na pořadí, v jakém je vypíše, ale každé sudé číslo musí vypsat přesně tolikrát, kolikrát se vyskytlo na vstupu.

P – III – 1

Znovu čokoláda

Jeníček bude mít zanedlouho narozeniny. Jeho sestra Mařenka si ještě dobře pamatuje na olámanou čokoládu, kterou od Jeníčka dostala před několika měsíci. Rozhodla se proto, že mu dá k narozeninám podobný dárek. Zašla do sklepa a ze své tajné skrýše vzala čokoládu, kterou si tam kdysi ukryla. Myši již stihly ohryzat i tuto čokoládu, ale to Mařence nevadilo — stačí přece ohryzané části olámat.

Mařenka je ovšem šikovnější než Jeníček a uvědomila si, že nemusí lámáním vytvořit čtverec. Čokolády mají přece často obdélníkový tvar. Tím získá mnoho nových možností, jak lze vyrobit dárek pro Jeníčka.

Soutěžní úloha. Je dán původní počet řádků R a počet sloupců S čokolády a matice $R \times S$ nul a jedniček udávající, která políčka čokolády zůstala zachována celá. Určete, kolika způsoby může Mařenka uskutečnit svůj plán. Jinými slovy, spočítejte, kolika způsoby lze ve zbytku čokolády

vyznačit obdélník bez děr. Všechny hrany obdélníka musí samozřejmě ležet na hranách políček. Stejně velké obdélníky umístěné na různých místech původní čokolády považujeme za různá řešení.

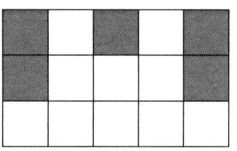
Formát vstupu: Na prvním řádku vstupu jsou dvě celá čísla R a S oddělená mezerou. Následuje R řádků, přičemž na r-tém z nich je S mezerami oddělených celých čísel $a_{r,1}, \ldots, a_{r,S}$. Je-li políčko čokolády na souřadnicích (r, s) celé, bude $a_{r,s} = 1$, v opačném případě $a_{r,s} = 0$.

Formát výstupu: Program vypíše na výstup jediné číslo — hledaný počet obdélníků.

Příklad:

Vstup:
```
3 5
0 1 0 1 0
0 1 1 1 0
1 1 1 1 1
```

Výstup:
```
33
```



Na obrázku vpravo je zobrazena čokoláda popsaná na vstupu. Šedou barvou jsou vyznačena políčka, která už myši stihly poškodit.

Obdélník 1×1 lze na této čokoládě vyznačit deseti způsoby, 2×1 pěti, 1×2 šesti, 3×1 dvěma, 1×3 čtyřmi, 1×4 dvěma, 2×2 dvěma, 1×5 jedním, a 2×3 také jedním způsobem.

P – III – 2

Šachovnice

„Šach-mat," oznámil s úšklebkem Jeníček. Mařenka měla sice dosud šachy velmi ráda, ale už ji to přestává bavit: právě s Jeníčkem prohrála sedmnáctou partii po sobě. Vymyslela si proto novou, vlastní hru, v níž Jeníčka určitě porazí.

Hracím plánem je šachovnice, která je směrem doprava a nahoru nekonečná. Každé políčko této šachovnice můžeme označit dvojicí nezáporných celých čísel (x, y). Políčko v levém dolním rohu šachovnice má označení $(0, 0)$, směrem doprava roste souřadnice x, směrem nahoru vzrůstá souřadnice y.

Na šachovnici je rozmístěno N šachových koní. Na začátku i kdykoliv během hry může stát více koní na témže políčku. Koně se pohybují podle šachových pravidel. Jsou povoleny pouze takové tahy, při nichž kůň neopustí šachovnici a navíc klesne součet obou souřadnic (viz obrázek na následující straně).

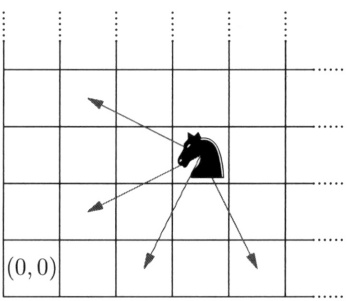
Hráč, který je na tahu, si vybere několik koní (může si jich vybrat, kolik chce, ale musí vždy aspoň jednoho) a každým z nich provede jeden tah. Hráči se ve hře pravidelně střídají. Prohrává ten, kdo nemůže provést další tah (tzn. nemůže pohnout podle pravidel žádným koněm).

Mařenka si už napsala program, který za ni bude hrát tuto hru optimálním způsobem. Jeníček ale programovat neumí, a proto by přivítal vaši pomoc.

Soutěžní úloha. Je dán počet koní N a jejich počáteční rozmístění na šachovnici. První tah provádí Jeníček.

Napište program, který zjistí, kdo vyhraje, když budou oba hráči hrát optimálně. Pokud zvítězí Jeníček, váš program by mu měl také poradit první tah libovolné vyhrávající strategie.

Jestliže úlohu nedokážete vyřešit pro obecné N, část bodů dostanete i v případě, že ji vyřešíte pro jednoho koně, případně pro dva koně začínající na souřadnicích mezi $(0, 0)$ a $(100, 100)$.



Všechny povolené tahy koněm na souřadnicích $(3, 2)$

Formát vstupu: První řádek vstupu obsahuje počet koní N. Na každém z následujících N řádků jsou dvě čísla r_i a s_i, která určují řádek a sloupec, kde se nachází i-tý kůň na začátku hry.

Formát výstupu: První řádek výstupu bude obsahovat jméno hráče, který vyhraje (`Jenicek` nebo `Marenka`). Jestliže vyhraje Jeníček, vypište pro každého koně, kterým má Jeníček v úvodním tahu táhnout, řádek s čísly r_a, s_a, r_b, s_b — původní a nová poloha koně. Na pořadí těchto řádků nezáleží.

Příklad 1:

Vstup:	*Výstup*:
`1`	`Marenka`
`0 4`	

Jeníček musí táhnout na $(1, 2)$, *odtud Mařenka táhne koněm na* $(0, 0)$ *a vyhraje.*

Příklad 2:

Vstup:	*Výstup*:
`2`	`Jenicek`
`3 1`	`3 1 1 0`
`2 1`	`2 1 0 0`

Po uvedeném tahu Jeníčka Mařenka ihned prohraje, neboť ani jedním koněm už nemůže pohnout.

Všimněte si, že kdyby Jeníček nechal koně z políčka $(3, 1)$ *na původním místě, prohraje. Prohrál by i v případě, že by tohoto koně přesunul na políčko* $(1, 2)$, *bez ohledu na to, zda by druhým koněm pohnul, nebo ne.*

P – III – 3

Počítač Kvak

K úloze se vztahuje studijní text z úlohy P–I–4.

Soutěžní úloha.

a) *(3 body)* V rouře počítače je posloupnost kladných celých čísel. Označme si je $a_1, a_2, \ldots, a_N$ v pořadí, v němž se v rouře nacházejí. Napište program, který zkontroluje, zda je tato posloupnost rostoucí. Pokud ano, program ukončí výpočet, aniž by cokoliv vypsal. Jestliže posloupnost není rostoucí, program zjistí a vypíše nejmenší i takové, že $a_i \geqq a_{i+1}$.

b) *(7 bodů)* V rouře počítače je posloupnost kladných celých čísel. Víte, že jedno z těchto čísel má v rouře nadpoloviční většinu — toto číslo se tedy v rouře vyskytuje vícekrát, než všechna ostatní čísla dohromady. Napište program, který toto číslo najde a vypíše.

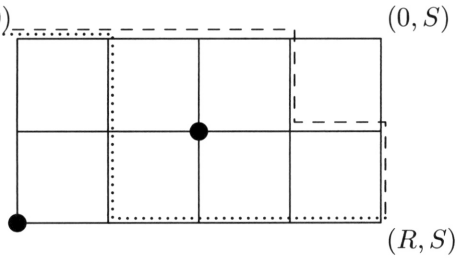
P – III – 4

Mravenci

Program: `mravenci.pas` / `mravenci.c` / `mravenci.cpp`
Vstup: `mravenci.in`
Výstup: `mravenci.out`

Právě začíná jedna velmi podivná soutěž. Porota si připravila N stejných mřížek obdélníkového tvaru. Mřížka je tvořena $R + 1$ vodorovnými a $S + 1$ svislými čarami. Každý soutěžící dostane jednu mřížku a několik

překážek. Jeho úkolem je rozmístit všechny překážky, které dostal, na různé mřížové body. Takto upravenou mřížku pak odevzdá porotě.



Příklad mřížky pro $R = 2$ a $S = 4$
Jsou vyznačeny 2 z 5 cest mravenců.

Porota do levého horního rohu mřížky vypustí speciální druh mravenců. Tito mravenci se pohybují pouze směrem dolů a doprava, a to jen po čarách tvořících mřížku. Mravenci se navíc navzájem nemají rádi, a proto žádní dva nepůjdu úplně stejnou cestou. Do pravého dolního rohu dorazí tudíž přesně tolik mravenců, kolik existuje různých cest mezi levým horním a pravým dolním rohem mřížky. Jelikož tento počet může být velký, při vyhodnocování soutěže se uvažuje jenom zbytek, který dostaneme po dělení tohoto počtu číslem $10^9 + 9$.

Soutěžní úloha. Je dáno N mřížek, všechny mají $R+1$ vodorovných a $S+1$ svislých čar. Pro každou mřížku je dán počet rozmístěných překážek a jejich souřadnice. Vaším úkolem je určit pro každou mřížku hodnotu $X \bmod 1\,000\,000\,009$, kde X je počet různých způsobů, jimiž se lze dostat z levého horního do pravého dolního rohu příslušné mřížky.

Poznámka. Je možné, že v některých výpočtech bude potřeba používat 64bitová celá čísla (typ `long long` v C/C++, `int64` v Pascalu).

Formát vstupu: Na prvním řádku vstupního souboru `mravenci.in` jsou tři kladná celá čísla R, S a N oddělená mezerami. Následuje N popisů jednotlivých mřížek.

Každý popis mřížky začíná řádkem obsahujícím jedno nezáporné celé číslo K, které představuje počet překážek rozmístěných na této mřížce. Dalších K řádků popisuje polohy překážek. Každý z nich obsahuje dvě mezerou oddělená čísla r_i, s_i ($0 \leqq r_i \leqq R$, $0 \leqq s_i \leqq S$), přičemž r_i je souřadnice řádku a s_i souřadnice sloupce, kde leží i-tá překážka. Žádné dvě překážky neleží na stejných souřadnicích a žádná překážka neleží na souřadnicích $(0, 0)$, kde mravenci začínají.

Omezení velikosti proměnných:

V testovacích datech budou proměnné R, S, N a K rovny nejvýše hodnotám uvedeným v následující tabulce:

č. testu	R	S	N	K
1	5	5	5	5
2	7	7	7	7
3	10	10	10	10
4	100	100	50	15
5	1 000	1 000	50	15
6	1200	100 000	1	15
7	1 000	1 000	1 000	2
8	1 000	1 000	1 000	10
9	2 000	2 000	200	15
10	15 000	15 000	50	10
11	1 000	1 000	1 000	50
12	2500	2500	200	100
13	20 000	10 000	200	50
14	100 000	100 000	2	30
15	100 000	100 000	100	50

Formát výstupu: Pro každou mřížku vypište jeden řádek s jedním celým číslem do souboru `mravenci.out`: hodnotu X mod 1 000 000 009, kde X je počet různých způsobů, jimiž lze dojít z levého horního do pravého dolního rohu této mřížky.

Příklad:

Vstup:

```
2 4 2
2
1 2
2 0
2
0 1
1 1
```

Výstup:

```
5
1
```

První mřížka je zobrazena na obrázku. Ve druhé mřížce vede jediná možná cesta bodem $(2, 0)$.

P – III – 5

Hurikán

Program: `hurikan.pas` / `hurikan.c` / `hurikan.cpp`
Vstup: `hurikan.in`
Výstup: `hurikan.out`

Souostroví Kiribati je tvořeno N ostrovy. Ještě předevčírem byla mezi těmito ostrovy postavena celá síť mostů, po nichž se dalo pohodlně přejet z každého ostrova na libovolný jiný. Včera se ale přehnal přes Kiribati hurikán Hermano a mnohé mosty zničil, takže jich zbylo pouze M. A co je ještě horší, K z těchto M mostů má narušenou statiku. Místní statik zjistil, že v každém z následujících K dní spadne jeden z těchto K poškozených mostů.

Místní vláda potřebuje zajistit, aby se obyvatelé i nadále mohli dostat každý den z libovolného ostrova na libovolný jiný. Rozhodla se proto najmout několik převozníků. Každý převozník zabezpečí dopravu mezi jednou konkrétní přidělenou dvojicí ostrovů.

Soutěžní úloha. Program dostane na vstupu popis mostů, které zůstaly zachovány po řádění hurikánu, a pořadí, v němž spadnou ty z nich, které jsou poškozeny. Program určí pro dnešek i pro každý z následujících K dní, jaký nejmenší počet převozníků stačí na příslušný den najmout.

Formát vstupu: První řádek vstupního souboru `hurikan.in` obsahuje dvě celá čísla N a M ($2 \leqq N$, $1 \leqq M$), která udávají počet ostrovů a počet mostů. Ostrovy jsou očíslovány od 1 do N.

Každý z následujících M řádků popisuje jeden most. Most je určen dvojicí celých čísel a_i b_i ($1 \leqq a_i, b_i \leqq N$, $a_i \neq b_i$) — čísla ostrovů, které daný most spojuje. Mosty si očíslujeme od 1 po M v pořadí, v němž jsou zadány na vstupu. Jednu dvojici ostrovů může spojovat i více mostů.

Následuje řádek obsahující celé číslo K ($1 \leqq K \leqq M$), které představuje počet poškozených mostů. Na posledním řádku vstupu je K mezerami oddělených čísel — čísla poškozených mostů v pořadí, v jakém spadnou.

Omezení velikosti proměnných:

Ve všech testovacích vstupech bude platit $M, N \leqq 200\,000$. V sadách testovacích dat 1 až 4 bude platit $N \leqq 1\,000$. V sadách 1 až 7 bude platit $K \leqq 100$.

Formát výstupu: Program vypíše do souboru `hurikan.out` $K + 1$ řádků obsahujících vždy jedno celé číslo. Číslo na i-tém řádku výstupu

udává minimální počet převozníků, kteří jsou zapotřebí, když spadne prvních $i - 1$ poškozených mostů.

Příklad:

Vstup:	*Výstup*:
4 4	0
1 2	0
3 2	1
1 3	2
3 4	
3	
2 4 3	

Dnes je ještě možné přejít po mostech mezi každými dvěma ostrovy. Bude to možné i zítra, až spadne most 3–2. *Až pozítří spadne most* 3–4, *bude již třeba najmout jednoho převozníka, aby nebyl ostrov* 4 *izolovaný od ostatních. Po pádu mostu* 1–3 *zůstane stát jediný most* 1–2. *Tehdy už bude zapotřebí zaměstnávat dva převozníky.*