64. ročník Matematické olympiády – 2014/2015

Úlohy krajského kola kategorie P

Krajské kolo 64. ročníku MO kategorie P se koná v úterý 20. 1. 2015 v dopoledních hodinách. Na řešení úloh máte 4 hodiny čistého času. V krajském kole MO-P se neřeší žádná praktická úloha, pro zajištění rovných podmínek řešitelů ve všech krajích je použití počítačů při soutěži zakázáno. Zakázány jsou rovněž jakékoliv další pomůcky kromě psacích potřeb (např. knihy, výpisy programů, kalkulačky, mobilní telefony). Řešení každé úlohy vypracujte na samostatný list papíru.

Řešení každé úlohy musí obsahovat:

- **Popis řešení,** to znamená slovní popis principu zvoleného algoritmu, *argumenty zdůvodňující jeho správnost* (případně důkaz správnosti algoritmu), diskusi o efektivitě vašeho řešení (časová a paměťová složitost). Slovní popis řešení musí být jasný a srozumitelný i bez nahlédnutí do samotného zápisu algoritmu (do programu). Není vhodné odkazovat se na vaše řešení úloh domácího kola, opravovatelé je nemusí mít k dispozici; na autorská řešení domácího kola se odkazovat můžete.
- **Zápis algoritmu.** V úlohách **P-II-1**, **P-II-2** a **P-II-3** je třeba uvést zápis algoritmu v nějakém dostatečně srozumitelném pseudokódu (případně v programovacím jazyce Pascal nebo C/C++). Nemusíte detailně popisovat jednoduché operace jako vstupy, výstupy, implementaci jednoduchých matematických vztahů, vyhledávání v poli, třídění apod. V úloze **P-II-4** je potřeba popsat požadovanou magickou síť a dokázat, že splňuje zadané vlastnosti; pokud požadovaná síť neexistuje, je nutné její neexistenci zdůvodnit.

Za každou úlohu můžete získat maximálně 10 bodů. Hodnotí se nejen správnost řešení, ale také kvalita jeho popisu a efektivita zvoleného algoritmu. Algoritmy posuzujeme podle jejich časové složitosti, tzn. závislosti doby výpočtu na velikosti vstupních dat. Záleží přitom pouze na řádové rychlosti růstu této funkce. V zadání každé úlohy najdete přibližné limity na velikost vstupních dat. Efektivním vyřešením úlohy rozumíme to, že váš program spuštěný s takovými daty na současném běžném počítači dokončí výpočet během několika sekund.

Vzorová řešení úloh naleznete krátce po soutěži na webových stránkách olympiády *http://mo.mff.cuni.cz/*. Na stejném místě bude zveřejněn i seznam úspěšných řešitelů postupujících do ústředního kola a také popis prostředí, v němž se budou v ústředním kole řešit praktické úlohy.

P-II-1 Rušení stanic podruhé

Zatímco se v Kocourkově čile bourá metro, kocourkovští matematici nemají co na práci. Rozhodli se proto spočítat, kolika způsoby mohlo zbourání sítě metra proběhnout. Připomeňme, že kocourkovské metro má strukturu souvislého neorientovaného grafu, kde vrcholy jsou stanice a hrany spojují dvojice stanic, mezi kterými se lze obousměrně přepravit. Matematici si navíc všimli, že síť metra tvoří strom, tzn. v síti není žádný cyklus. Kocourkovští vědci by chtěli znát počet různých pořadí stanic, ve kterých je lze bourat, aniž by někdy byla narušena souvislost zbývající sítě. Tento problém je pro ně však příliš obtížný, a proto se obrátili na vás jako na počítačové experty, abyste jim hledanou hodnotu vyčíslili.

Soutěžní úloha

Na vstupu je zadán neorientovaný strom. Vypište počet různých permutací jeho vrcholů, které určují pořadí vrcholů, v němž lze vrcholy postupně odebírat ze stromu, aniž by zbytek stromu někdy přestal být souvislý. Výsledné číslo může být obrovské, pro účely této úlohy ale můžete předpokládat, že celočíselné typy mají neomezený rozsah a běžné aritmetické operace (sčítání, odčítání, násobení, dělení) s nimi lze provádět v konstantním čase.

Formát vstupu

Program čte vstupní data ze standardního vstupu. Na prvním řádku vstupu je počet vrcholů N. Vrcholy jsou očíslovány od 1 do N. Na každém z dalších $N - 1$ řádků jsou dvě celá čísla x a y ($1 \leq x, y \leq N$), udávající že vrcholy x a y jsou spojeny hranou.

Formát výstupu

Program vypíše na standardní výstup jediné celé číslo udávající počet pořadí vrcholů, v nichž je lze odebírat ze zadaného stromu bez narušení souvislosti.

Příklad

Vstup:	*Výstup:*

`4`
`1 2`
`1 3`
`1 4`
	`12`

Možná pořadí odebírání jsou 2341, 2431, 3241, 3421, 4231, 4321, 2314, 3214, 2413, 4213, 3412, 4312.

Bodování

Plných 10 bodů získáte za správné řešení, které efektivně vyřeší libovolný vstup s $N \leq 1\,000\,000$. Až 7 bodů získáte za správné řešení, které efektivně vyřeší libovolný vstup s $N \leq 1\,000$. Až 4 body obdržíte za správné řešení, které efektivně vyřeší libovolný vstup s $N \leq 10$.

P-II-2 Ďábel

Starý prašivý ďábel již dávno nikomu hrůzu nenahání. Je potřeba vybrat nového vůdce pekla, nejlépe takového čerta, který bude mít následující vlastnosti:

- všichni ostatní čerti v pekle se ho bojí,
- on se nebojí žádného jiného čerta v pekle.

Pomozte pekelné komisi zjistit, zda je v pekle vhodný kandidát na ďábla, a pokud ano, jednoho takového nalezněte.

Soutěžní úloha

V pekle žije N čertů očíslovaných od 1 do N. Není předem známo, který čert se kterého bojí. Abyste to zjistili, můžete se pro libovolné dva čerty s čísly a, b zeptat, zda se čert a bojí čerta b. Je ovšem možné, že dva různí čerti se ani jeden druhého nebojí, nebo naopak se bojí jeden druhého navzájem.

Navrhněte postup, jak pokládat jednotlivé dotazy, aby bylo možné co nejdříve určit, zda se v pekle nachází vhodný kandidát na ďábla. Jako kvalitu vašeho řešení budeme hodnotit počet dotazů v nejhorším případě v závislosti na hodnotě N. Oproti běžným úlohám nás tentokrát bude zajímat přesný počet dotazů, nikoliv jen asymptotický odhad. Přestože časová složitost algoritmu není hlavním předmětem hodnocení, dbejte na to, abyste nevykonali řádově větší počet operací, než bude počet dotazů.

Implementační detaily

Implementuje funkci *ďábel*(N), která má jeden argument N – počet čertů v pekle. Funkce vrátí hodnotu 0, pokud vhodný kandidát na ďábla neexistuje, jinak vrátí číslo nějakého vhodného kandidáta. Vaše funkce může během svého výpočtu volat funkci *bojí_se*(a, b), která vrátí 1, pokud se čert a bojí čerta b, v opačném případě vrátí 0.

V programovacím jazyce by naše funkce byly deklarovány takto:

```
// C nebo C++
int dabel(int n);
int boji_se(int a, int b);

{ Pascal }
function dabel(n: integer): integer;
function boji_se(a, b: integer): integer;
```

Příklad

Pokud vaše funkce bude zavolána jako *ďábel*(3) a postupně položí dotazy

$$\begin{aligned} &\textit{bojí\_se}\,(1,2) \to 1\\ &\textit{bojí\_se}\,(2,3) \to 0\\ &\textit{bojí\_se}\,(3,2) \to 1\\ &\textit{bojí\_se}\,(2,1) \to 0, \end{aligned}$$

je již jasné, že jediným vhodným kandidátem na ďábla je čert s číslem 2.

P-II-3 Okružní jízda

Po zrušení metra se v Kocourkově zhoršila dopravní situace a zvýšil se počet dopravních nehod. Radní se proto rozhodli zavést ve městě jednosměrky. Pustili se do toho s takovým nadšením, že nyní zůstaly na každé křižovatce nanejvýš dvě ulice, které jsou obousměrné. To trochu komplikuje život popelářům, kteří by rádi projeli každou ulicí právě jednou.

Soutěžní úloha

Je zadána síť ulic s N křižovatkami očíslovanými od 1 do N a s M ulicemi spojujícími dvojice křižovatek (ulice se potkávají pouze v popsaných křižovatkách, všechna další křížení jsou řešena nadjezdy). U každé ulice je specifikováno, zda a kterým směrem je jednosměrná. Do každé křižovatky vedou nejvýše dvě ulice, které nejsou jednosměrné. Nalezněte způsob, jak vyjet z křižovatky číslo 1, projet každou ulicí právě jednou a skončit opět na křižovatce číslo 1, nebo rozhodněte, že žádný takový způsob neexistuje. Jednosměrnými ulicemi smíte projíždět pouze v povoleném směru.

Formát vstupu

Program čte vstupní data ze standardního vstupu. První řádek obsahuje dvě celá čísla N, M oddělená mezerou ($N \geq 1$, $M \geq 0$) udávající po řadě počet křižovatek a počet ulic. Každý z následujících M řádků popisuje jednu ulici. Obsahuje dvě různá celá čísla k_1 a k_2 z rozmezí od 1 do N (čísla křižovatek) a písmeno s (povolený směr jízdy). Je-li $s =$ `J`, pak je ulice jednosměrná a je povoleno projet ji pouze ve směru z k_1 do k_2. Je-li $s =$ `O`, pak je ulice obousměrná. Mezi každými dvěma křižovatkami vede nejvýše jedna ulice.

Formát výstupu

Program vypíše na standardní výstup jediný řádek. Na něm bude $M + 1$ mezerami oddělených čísel k_0, k_1, …, k_M udávajících pořadí, v jakém mají být projety křižovatky tak, aby $k_0 = k_M = 1$ a každá ulice byla projeta právě jednou. Pokud existuje více možných řešení, vypište jedno libovolné z nich. Neexistuje-li řešení, vypište slovo `nelze`.

Příklady

Vstup:	*Výstup:*

`5 6`
`1 3 J`
`3 2 J`
`2 1 J`
`2 4 O`
`5 4 O`
`2 5 O`
	`1 3 2 5 4 2 1`

Jiné možná řešení je `1 3 2 4 5 2 1`.

Vstup:

```
3 3
2 1 J
3 1 J
2 3 O
```

Výstup:

```
nelze
```

Bodování

Plných 10 bodů dostanete za správné řešení, které efektivně vyřeší libovolný vstup s $N \leq 100\,000$ a $M \leq 1\,000\,000$. Až 8 bodů získáte za správné řešení, které efektivně vyřeší libovolný vstup s $N \leq 1\,000$. Částečné řešení předpokládající, že s každou křižovatkou sousedí nejvýše jedna obousměrná ulice, získá 5–7 bodů v závislosti na jeho efektivitě. Částečné řešení předpokládající, že všechny ulice jsou jednosměrné, může získat až 5 bodů.

P-II-4 Magická síť

K této úloze se vztahuje studijní text uvedený na následujících stranách. Studijní text je identický se studijním textem z domácího kola.

V zadání úloh se vyskytují následující omezení:

- ZERO(x) předepisuje, že proměnná x má hodnotu 0.
- NAE(x, y, z) předepisuje, že proměnné x, y a z nemají všechny stejnou hodnotu.
- XOR(x, y, z) předepisuje, že z proměnných x, y a z jich lichý počet musí mít hodnotu 1.
- XOR$_4(w, x, y, z)$ předepisuje, že z proměnných w, x, y a z jich lichý počet musí mít hodnotu 1.

Úkol 1: *(3 body)* Ukažte, že pomocí XOR nelze simulovat XOR$_4$.

Úkol 2: *(3 body)* Nalezněte síť používající pouze omezení typu XOR a ZERO, která simuluje XOR$_4$.

Úkol 3: *(4 body)* Nalezněte síť používající pouze omezení typu NAE, která simuluje XOR$_4$.

Studijní text

Magická síť se skládá z *omezení* a *proměnných*. Každá proměnná může nabývat hodnot 0 nebo 1. Omezení pak předepisují podmínky, které ohodnocení proměnných musí splňovat. Například omezení XOR(x, y, z) předepisuje, že z proměnných x, y a z jich lichý počet musí mít hodnotu 1, omezení OR(x, y, z) předepisuje, že alespoň jedna z x, y a z musí mít hodnotu 1, omezení EQ(x, y) předepisuje, že x a y musí mít stejnou hodnotu, a podobně. Proměnné se v jednom omezení mohou opakovat.

Některé z proměnných jsou *vstupní* a můžeme jim nastavit konkrétní hodnotu. Po seslání příslušného zaklínadla se pak ostatním proměnným nastaví takové hodnoty, aby všechna omezení byla splněna. Pokud žádná taková volba hodnot neexistuje, zaklínadlo nás na to upozorní. V prvním případě říkáme, že magická síť zadaný vstup *přijímá*, ve druhém ho *odmítá*. Magická síť *simuluje omezení* O, jestliže přijímá právě stejné hodnoty proměnných jako omezení O.

Příklad 1: Magickou síť zapisujme jako seznam typů omezení, k nimž do závorek budeme připisovat proměnné, na které jsou aplikovány. Síť XOR(a, b, c), XOR(b, c, d) tedy vynucuje, že lichý počet z proměnných a, b a c má hodnotu 1 a že lichý počet z proměnných b, c a d má hodnotu 1.

Nechť a a d jsou vstupní proměnné této sítě. Jestliže a má hodnotu 0, pak právě jedna z proměnných b a c musí mít hodnotu 1, a proto d musí mít hodnotu 0. Naopak, má-li a hodnotu 1, pak hodnota proměnné b musí být stejná jako hodnota proměnné c, a proto d musí mít hodnotu 1.

Tato magická síť tedy přijímá právě ty vstupy, kde a a d mají stejnou hodnotu, a simuluje tedy omezení EQ(a, d).

Obecněji: Typicky nás bude zajímat, která omezení jdou vyjádřit pomocí jiných. Říkáme, že množina typů omezení $\{O_1, O_2, \ldots, O_n\}$ *simuluje* omezení O, jestliže existuje magická síť používající pouze omezení typu O_1, O_2, …, O_n, která simuluje O. Příklad 1 tedy ukazuje, že XOR simuluje EQ.

Omezení $O(x_1, \ldots, x_n)$ nazveme *slabé*, jestliže zakazuje právě jednu kombinaci hodnot proměnných $x_1, \ldots, x_n$. Slabé omezení budeme zapisovat jako $S_{h_1 h_2 \ldots h_n}$, kde h_1, …, h_n jsou hodnoty proměnných, které zakazuje. Třeba omezení $S_{001}(x, y, z)$ je splněno, jestliže $x = 1$ nebo $y = 1$ nebo $z = 0$, a omezení S_{000} je stejné jako OR. Nechť SAT_n označuje množinu všech slabých omezení s právě n proměnnými.

Příklad 2: SAT_3 simuluje XOR, jelikož síť

$$S_{000}(x, y, z), S_{011}(x, y, z), S_{101}(x, y, z), S_{110}(x, y, z)$$

zakazuje všechny kombinace hodnot proměnných x, y a z, v nichž se hodnota 1 vyskytuje suděkrát.

Příklad 3: Množina omezení SAT_2 nesimuluje OR. Abychom si to dokázali, zaveďme si nejprve funkci maj se třemi vstupy. Ta bude vracet ten vstup, který se vyskytuje nejčastěji (proto se jí také někdy říká *majorita*). Tedy třeba $maj(1, 1, 1) = maj(1, 0, 1) = 1$ a $maj(0, 0, 1) = 0$.

Uvažujme nyní libovolnou síť s omezeními z množiny SAT_2. Nechť x_1, …, x_n jsou proměnné této sítě. Řekněme, že by tato síť simulovala $\mathrm{OR}(x_1, x_2, x_3)$. Jelikož omezení OR je splněno pro hodnoty 1, 0 a 0, existuje nějaké přiřazení hodnot proměnným, které splňuje všechna omezení, x_1 má hodnotu 1 a x_2 a x_3 mají hodnoty 0. Nechť a_i označuje hodnotu proměnné x_i v tomto přiřazení, pro $i = 1, \ldots, n$. Obdobně existuje přiřazení s hodnotami b_i splňující všechna omezení takové, že $b_2 = 1$ a $b_1 = b_3 = 0$, a přiřazení s hodnotami c_i splňující všechna omezení takové, že $c_3 = 1$ a $c_1 = c_2 = 0$.

Uvažme ohodnocení s hodnotami $d_i = maj(a_i, b_i, c_i)$. Tvrdíme, že d_i také splňuje všechna omezení: Mějme nějaké omezení $S_{h_1 h_2}(x_i, x_j)$ ze sítě. Pokud $a_i = b_i$ a $a_j = b_j$, pak $d_i = maj(a_i, a_i, c_i) = a_i$ a $d_j = maj(a_j, a_j, c_j) = a_j$ splňuje podmínku $S_{h_1 h_2}$, protože ji splňuje a_i a a_j. Proto předpokládejme, že $a_i \neq b_i$ nebo $a_j \neq b_j$, a obdobně $a_i \neq c_i$ nebo $a_j \neq c_j$ a stejně tak $b_i \neq c_i$ nebo $b_j \neq c_j$. Ze symetrie mezi i a j a mezi ohodnoceními a, b a c stačí uvažovat případ, že $a_i \neq b_i$ a $a_i \neq c_i$. Z toho odvodíme, že $b_i = c_i$, a proto $b_j \neq c_j$. Díky symetrii mezi b a c pak stačí uvažovat případ, že $a_j = b_j$ a $a_j \neq c_j$. Pak ale $\mathrm{maj}(a_i, b_i, c_i) = maj(a_i, b_i, b_i) = b_i$ a $\mathrm{maj}(a_j, b_j, c_j) = maj(b_j, b_j, c_j) = b_j$, a proto $d_i = b_i$ a $d_j = b_j$ splňuje podmínku $S_{h_1 h_2}$.

Povšimněme si, že $d_1 = maj(1, 0, 0) = 0$ a obdobně $d_2 = d_3 = 0$. Ale omezení $\mathrm{OR}(x_1, x_2, x_3)$ předepisuje, že alespoň jedna z proměnných x_1, x_2 a x_3 má hodnotu 1, a proto v každé magické síti simulující $\mathrm{OR}(x_1, x_2, x_3)$ musí přiřazení hodnot d_i proměnným porušovat nějaké omezení. Uvažovaná síť tedy nesimuluje $\mathrm{OR}(x_1, x_2, x_3)$.