66. ročník Matematické olympiády – 2016/2017

Úlohy ústředního kola kategorie P – 1. soutěžní den

Na řešení úloh máte 4,5 hodiny čistého času. Řešení každé úlohy pište na samostatný list papíru. Při soutěži je zakázáno používat jakékoliv pomůcky kromě psacích potřeb (tzn. knihy, kalkulačky, mobily, apod.).

Řešení každého příkladu musí obsahovat:

- **Popis řešení,** to znamená slovní popis principu zvoleného algoritmu, argumenty zdůvodňující jeho správnost, 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 povoleno odkazovat se na Vaše řešení předchozích kol, opravovatelé je nemají k dispozici; na autorská řešení se odkazovat můžete.
- **Zápis algoritmu.** V úlohách **P-III-1** a **P-III-2** je třeba uvést zápis algoritmu, a to buď ve tvaru zdrojového textu nejdůležitějších částí programu v jazyce Pascal nebo C/C++, nebo v nějakém dostatečně srozumitelném pseudokódu. Nemusíte detailně popisovat jednoduché operace jako vstupy, výstupy, implementaci jednoduchých matematických vztahů, vyhledávání v poli, třídění apod. Řešení úlohy **P-III-3** bude vypadat podobně, ale místo pseudokódu musí obsahovat program pro stromochod.

Za každou úlohu můžete získat maximálně 10 bodů. Hodnotí se nejen správnost řešení, ale také kvalita jeho popisu (včetně zdůvodnění správnosti) a efektivita zvoleného algoritmu. Algoritmy posuzujeme zejména podle jejich časové složitosti, tzn. podle 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.

P-III-1 Bizoní rezervace

V Severní Americe stále přežívá několik stád bizonů. Jakkoliv nepravděpodobné se to může zdát, každé stádo se vždy pohybuje po předem známé přímce a nikdy ji neopustí. Každé setkání dvou stád na jednom místě je jedinečný zážitek, a proto ochránci přírody vyhlásili každé místo, kde by setkání mohlo nastat, chráněnou přírodní památkou. Nyní je potřeba všechny vytyčené přírodní památky ohradit a vytvořit tak unikátní bizoní rezervaci. Budování ohrady je nákladné, a proto bychom chtěli, aby celková délka ohrady byla co nejkratší.

Uvažujme, že Severní Amerika je nekonečná rovina, žádné dvě přímky, po kterých se stáda bizonů pohybují, nejsou rovnoběžné a žádné tři takové přímky se neprotínají v jediném bodě. Nalezněte ohradu s nejmenším obvodem, která obsahuje všechny průsečíky zadaných přímek. Ohradu popište souřadnicemi přírodních

památek na jejím obvodu ve směru chodu hodinových ručiček. Leží-li nějaká přírodní památka z obvodu na přímce mezi dvěma jinými přírodními památkami na obvodu, pak ji nevypisujte. Existuje-li více vyhovujících řešení, nalezněte libovolné z nich.

Formát vstupu

Na prvním řádku vstupu je jediné přirozené číslo N ($3 \leq N \leq 10^6$) – počet stád bizonů. Následuje N řádků a každý z nich popisuje jedno bizoní stádo. Konkrétně, i-té bizoní stádo je popsáno třemi celými čísly oddělenými mezerami A_i, B_i a C_i, žádné z nich v absolutní hodnotě nepřesáhne 10^6. Navíc můžete předpokládat, že pro žádné stádo nenastane $A_i = B_i = 0$. Potom se i-té stádo pohybuje po přímce určené takovými dvojicemi bodů (x, y), pro které platí

$$A_i \cdot x + B_i \cdot y = C_i.$$

Zadané přímky budou splňovat předpoklady ze zadání (žádné dvě nejsou rovnoběžné ani shodné a žádné tři se neprotínají v jednom bodě).

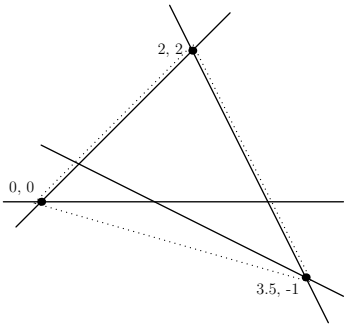
Formát výstupu

Vypište souřadnice přírodních památek na obvodu hledané ohrady, v pořadí po směru hodinových ručiček. Souřadnice každé památky vypisujte na zvláštní řádek a oddělte je mezerou.

Příklad

Vstup:	*Výstup:*
`4`	`0 0`
`0 1 0`	`2 2`
`1 -1 0`	`3.5 -1`
`2 1 6`	
`2 4 3`	

Vstup odpovídá následujícímu obrázku, tečkovaná čára znázorňuje hledanou ohradu.



Bodování

Plných 10 bodů získáte za správné řešení, které úlohu efektivně vyřeší pro $N \leq 10^6$. Až 5 bodů můžete získat za správné řešení, které úlohu efektivně vyřeší pro $N \leq 1\,000$.

P-III-2 Hostina

Hostimír si našel nové zaměstnání. Každý zaměstnanec jeho nové firmy má pevně stanovenou pracovní dobu. Každému je jeho pracovní doba stanovena s ohledem na jeho potřeby – někteří mohou pracovat jen ráno, někteří pracují i přes půlnoc, a podobně; nikdo ale nepracuje celý den. Stejný rozvrh se opakuje pravidelně každý den.

Hostimír by rád upekl koláčky a pohostil ostatní zaměstnance. Nicméně ví, že jakmile koláčky vybalí, všichni se na ně vrhnou a okamžitě je snědí. Vzhledem k různým rozvrhům zaměstnanců ale nemusí nikdy být ve firmě všichni přítomni zároveň. Aby tedy pohostil všechny zaměstnance, bude Hostimír muset upéct více dávek; pomozte mu určit kolik.

Soutěžní úloha

Na vstupu dostanete celé číslo m udávající délku dne (v milisekundách) na Hostimírově planetě a n intervalů $\langle a_i, b_i \rangle$ udávajících začátky a konce pracovních dob zaměstnanců, opět v milisekundách. Může být i $b_i < a_i$, v tom případě daný zaměstnanec pracuje přes půlnoc. Vybalí-li Hostimír koláčky v čase t, pohostí všechny zaměstnance splňující $a_i \leq t \leq b_i$ nebo $b_i < a_i \leq t$ nebo $t \leq b_i < a_i$.

Nalezněte co nejméně časů $t_1, \ldots, t_k$ takových, že vybalí-li Hostimír koláčky v těchto časech, pak pohostí všechny zaměstnance firmy.

Formát vstupu

Na prvním řádku dostanete dvě přirozená čísla, n a m ($1 \leq n \leq 10^6$, $4 \leq m \leq 10^9$), kde číslo m udává počet milisekund dne na Hostimírově planetě a n je počet zaměstnanců firmy. Na i-tém z dalších n řádků jsou dvě různá celá čísla a_i a b_i, kde $0 \leq a_i, b_i < m$.

Formát výstupu

Vypište jedno celé číslo k udávající nejmenší počet různých časů takových, že každý zaměstnanec je ve firmě v alespoň jeden z nich (vybalí-li tedy Hostimír koláčky v těchto k časech, pohostí všechny zaměstnance).

Příklad

Vstup:	*Výstup:*
3 20 1 4 15 18 19 3	2

Hostimír může vybalit koláčky v časech 3 *a* 16.

Bodování

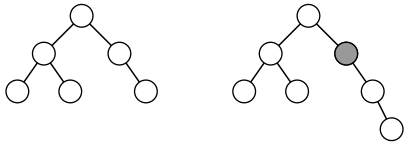
Plných 10 bodů získáte za správné řešení, které úlohu efektivně vyřeší pro $n \leq 10^6$. Až 7 bodů můžete získat za řešení, které úlohu efektivně vyřeší pro $n \leq 1\,000$. Až 5 bodů můžete získat za řešení, které předpokládá, že nikdo nepracuje přes půlnoc (tj. všechny zadané intervaly $\langle a_i, b_i \rangle$ splňují $a_i < b_i$).

P-III-3 Stromochod

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

Pro každý vrchol stromu definujeme jeho *levý podstrom,* který obsahuje levého syna a všechny jeho (i nepřímé) potomky. Pokud levý syn neexistuje, levý podstrom je prázdný. Analogicky *pravý podstrom.*

O stromu pak řekneme, že je *dokonale vyvážený,* pokud pro každý jeho vrchol platí, že počet vrcholů v levém a pravém podstromu se liší nejvýše o 1. Například první strom na následujícím obrázku je dokonale vyvážený, zatímco druhý ne: levý podstrom šedivého vrcholu je prázdný, ale pravý dvouvrcholový.



Naprogramujte stromochod, aby ověřil, zda je zadaný strom dokonale vyvážený. Odpovězte značkou `ok` v kořeni.

Ve svém řešení nemusíte programovat technické detaily, ale popište je dostatečně podrobně na to, aby bylo jasné, jak se na stromochodu naprogramují a s jakou složitostí.

Studijní text

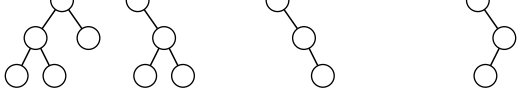
V tomto ročníku olympiády budeme programovat *stromochod* – speciální zařízení určené k procházení binárních stromů.

Graf se skládá z konečné množiny *vrcholů* a konečné množiny *hran.* Každá hrana spojuje dva vrcholy, každé dva vrcholy jsou spojeny nejvýše jednou hranou. Hrany nemají směr – nerozlišujeme, který vrchol je počáteční a který koncový. *Cesta* říkáme posloupnosti vrcholů, které na sebe navazují (i-tý je spojený hranou s $(i+1)$-ním).

Strom je graf, v němž mezi každými dvěma vrcholy vede právě jedna cesta. Strom můžeme *zakořenit* – jeden jeho vrchol prohlásit za *kořen.* Pro každé dva vrcholy spojené hranou pak umíme říci, který z nich je *otec* (to je ten bližší kořeni) a který *syn.* Kořen je jediný vrchol, který nemá otce. Vrcholům, které nemají žádné syny, říkáme *listy* stromu. *Podstrom* budeme říkat části stromu tvořené daným vrcholem, jeho syny, syny jeho synů, a tak dále až k listům. Podstrom je tedy také strom a daný vrchol je jeho kořenem.

Binární je takový zakořeněný strom, jehož každý vrchol má nejvýše dva syny. U synů rozlišujeme, který z nich je *levý* a který *pravý.* Dokonce i v případech, kdy existuje jen jeden syn, zajímá nás, zda je levý, nebo pravý.

Na následujícím obrázku vidíte několik různých binárních stromů:



Stromochod slouží k řešení různých problému týkajících se binárních stromů. Můžeme si ho představit jako počítač, který se pohybuje po stromu a v každém okamžiku výpočtu se nachází v právě jednom z vrcholů stromu.

Paměť stromochodu je omezená: může si pamatovat jen konečné množství bitů informace, tedy v každém okamžiku se může nacházet jen v jednom z konečně mnoha stavů. Mimo to stromochod smí poznamenávat informace do navštívených vrcholů: do každého vrcholu se vejde opět jen konečné množství informace a tyto informace může stromochod číst a zapisovat pouze v tom vrcholu, v němž se zrovna nachází. Těmto údajům uloženým ve vrcholu budeme říkat *značky.*

Stromochod budeme programovat v jazyce podobném Pascalu nebo C. Kvůli omezení na konečný počet stavů můžeme používat pouze celočíselné proměnné s předem daným rozsahem (třeba `0..9`) a booleovské proměnné pro pravdivostní hodnoty. To například znamená, že není možné používat pole ani ukazatele. Z řídicích konstrukcí jsou k dispozici podmínky, cykly a `goto`. Procedury a funkce je možné používat, ale na jejich parametry se vztahují stejná omezení na typy a není povolena rekurze.

Na aktuální vrchol se můžeme odkazovat prostřednictvím proměnné `V`. Ta se chová jako záznam (pascalský `record`, Céčková `struct`) a obsahuje značky uložené

ve vrcholu. Podobu značek si můžete předepsat, ale musí jich být konstantní počet a opět platí omezení na povolené typy.

Pokud chceme stromochod přesunout do levého syna, zavoláme funkci `jdi_l`. Podobně `jdi_p` pro pravého syna a `jdi_o` pro otce. Tato funkce nemá žádné parametry ani výsledek. Pokud se pokusíte přesunout do neexistujícího vrcholu, program se zastaví s chybou. Proto se hodí předem otestovat, zda syn či otec existuje: k tomu slouží funkce `ex_l`, `ex_p` a `ex_o`. Ty nemají žádné parametry a vracejí booleovskou hodnotu.

Výpočet stromochodu probíhá následovně. Na vstupu dostaneme nějaký strom, stromochod umístíme do jeho kořene a všechny značky vynulujeme. Výjimku mohou tvořit značky, o kterých je výslovně řečeno, že jsou součástí vstupu. Poté se rozeběhne program, stromochod se prochází po stromu a nakonec se zastaví. Výstup potom přečteme z určených značek.

Pozor, jeden program pro stromochod musí danou úlohu řešit pro všechny stromy. Rozsah proměnných tedy nesmí záviset na velikosti stromu.

Efektivitu programů můžeme posuzovat obvyklým způsobem. Doba běhu programu bude odpovídat počtu přesunů po stromu, časová složitost je maximum z dob běhu přes všechny stromy s daným počtem vrcholů. Paměťovou složitost neposuzujeme, protože všechny programy ukládají lineární množství informace.

Příklad

Rozmysleme si, jak stromochodem navštívit všechny vrcholy stromu a do každého umístit značku. Strom budeme procházet do hloubky: začneme v kořeni, pak se zanoříme do levého podstromu, až se z něj vrátíme, přesuneme se do pravého podstromu, a nakonec se vrátíme z celého stromu. (Pokud si strom nakreslíme na papír, tento průchod odpovídá obcházení z kořene proti směru hodinových ručiček.)

Jelikož nemáme k dispozici rekurzi, budeme si v každém vrcholu pamatovat značku jménem `stav`, která bude říkat, kolikrát jsme už ve vrcholu byli. Navštívíme-li vrchol poprvé, zamíříme do levého syna; podruhé zamíříme do pravého a potřetí se už vrátíme do otce. Pokud levý nebo pravý syn neexistují, budeme se chovat, jako bychom se z nich okamžitě vrátili.

Program bude vypadat následovně.

```
type vrchol = record
   { Tento typ popisuje, jaké značky ukládáme do vrcholů }
   byl_jsem_tu: boolean;      { Chceme nastavit ve všech vrcholech }
   stav: 0..3;                { Kolikrát jsme ve vrcholu byli }
end;

begin
   repeat
      V.byl_jsem_tu := true;
      V.stav := V.stav + 1;
      case V.stav of
         1: if ex_l then jdi_l;
         2: if ex_p then jdi_p;
         3: if not ex_o then halt else jdi_o;
```

```
      end;
   until false;
end.
```

Časová složitost tohoto programu je lineární s počtem vrcholů, protože každý vrchol navštívíme nejvýše třikrát.