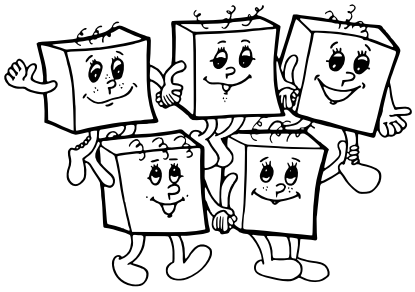
OLYMPIÁDA V INFORMATIKE

NA STREDNÝCH ŠKOLÁCH

http://oi.sk/



tridsiaty druhý ročník

školský rok 2016/2017

zadania celoštátneho kola, deň 1
kategória A

Priebeh celoštátneho kola

Celoštátne kolo 32. ročníka Olympiády v informatike, kategórie A, sa koná v dňoch 29. marca – 1. apríla 2017. Na riešenie úloh prvého, teoretického dňa majú súťažiaci 4,5 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 prvého (teoretického) dňa

Za každú úlohu môžete získať od 0 do 10 bodov.
Pokiaľ nie je v zadaní povedané ináč, najdôležitejšie dve kritériá hodnotenia sú v prvom rade **správnosť** a v druhom rade **efektívnosť** navrhnutého algoritmu. Na výslednom počte bodov sa môže prejaviť 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. Nezávisí pri tom na konštantných faktoroch, len na rádovej rýchlosti rastu tejto funkcie.
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, vyrieši vzorové riešenie ľubovoľný povolený vstup nanajvýš za niekoľko sekúnd.

A-III-1 Bizónia rezervácia

Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo.
(Bizóny z Buffala ktoré sú zastrašované inými bizónmi z Buffala zastrašujú ďalšie bizóny z Buffala.)

V severnej Amerike ešte stále prežívajú stáda bizónov. Voľakedy tieto stáda behali voľne po celej prérii. Klimatické zmeny však ovplyvnili ich život natoľko, že sa voľného pohybu boja. Každé stádo má svoju jednu „rokmi overenú“ priamku po ktorej behá ako sa mu zachce, avšak nikdy sa z nej neodkloní.
Ochrancovia prírody by chceli všetky bizónie stáda oplotiť. To však nie je len tak. Občas sa stane, že sa dve bizónie stáda stretnú. Toto je veľmi dôležitý jav, lebo má kladný vplyv na diverzitu populácií stád. Preto by bolo dobré, aby ohrada nebránila žiadnym potenciálnym stretnutiam.

Súťažná úloha

V rovine máme daných n priamok, z ktorých žiadne dve nie sú rovnobežné a žiadne tri neprechádzajú jedným bodom. Nájdite ohradu **s najkratším možným obvodom** v ktorej alebo na ktorej obvode ležia všetky priesečníky zadaných priamok.

Formát vstupu a výstupu

V prvom riadku vstupu je číslo n udávajúce počet stád bizónov. V každom z nasledujúcich n riadkov je popis jedného stáda: čísla a_i, b_i a c_i udávajúce, že stádo behá po priamke danej rovnicou $a_i \cdot x + b_i \cdot y + c_i = 0$. (Čísla a_i a b_i nikdy nebudú obe nulové.)

Je zjavné, že hľadaná ohrada bude mať tvar mnohouholníka. Na výstup vypíšte súradnice jeho vrcholov v poradí, v akom ležia na jeho obvode.

Obmedzenia a hodnotenie

Riešenie, ktoré efektívne vyrieši vstupy s $3 \leq n \leq 1\,000\,000$, môže získať 10 bodov.
Riešenie, ktoré efektívne vyrieši vstupy s $3 \leq n \leq 1\,000$, môže získať 5 bodov.

Príklad

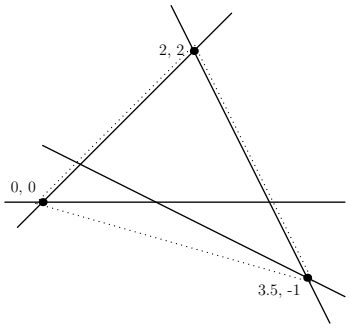
vstup

```
4
0 1 0
1 -1 0
2 1 -6
2 4 -3
```

výstup

```
0 0
2 2
3.5 -1
```

Vstup a výstup zodpovedajú obrázku. Bodkovaná čiara predstavuje ohradu.



A-III-2 Kvietky

Šandyna pestuje rôzne rastliny. Každá rastlina má svoj stabilný 24-hodinový cyklus, počas ktorého práve raz otvorí kvety a následne ich práve raz zavrie. Rôzne rastliny otvárajú svoje kvety v rôznych časoch. Niektoré orchidey sú dokonca také zvláštne, že kvety otvárajú v noci a kvitnú aj cez polnoc.

Súťažná úloha

Šandyna si chce nafotiť svoju zbierku. Koľko najmenej snímok musí spraviť, ak chce, aby každá rastlina mala aspoň na jednej fotke otvorené kvety?

Formát vstupu a výstupu

Deň si predstavíme ako zľava uzavretý interval $\langle 0, 10^{18})$. Čas 0 je polnoc ktorou deň začína, čas 10^{18} je polnoc ktoru začína nasledujúci deň (čiže zároveň ide o čas 0 nasledujúceho dňa).

V prvom riadku vstupu je číslo n udávajúce počet Šandyniných rastliniek.
V každom z nasledujúcich n riadkov je popis jednej rastlinky: čas a_i kedy otvorí kvety a iný čas b_i kedy ich zavrie. Ak $a_i > b_i$, znamená to teda, že táto rastlinka má kvety otvorené cez polnoc.
(Rastlina má kvety otvorené aj v okamihoch a_i a b_i, ide teda o uzavretý interval časov.)

Vypíšte jedno číslo f: najmenší počet fotografií ktoré stačí spraviť.

Obmedzenia a hodnotenie

Na plný počet bodov riešte úlohu pre $n \leq 1\,000\,000$.
Riešenia dostatočne rýchle pre $n \leq 5\,000$ môžu získať nanajvýš 7 bodov.
Nanajvýš 5 bodov môžete dostať za riešenie, ktoré je efektívne pre $n \leq 1\,000\,000$ ale funguje len za dodatočného predpokladu že žiaden kvet nekvitne cez polnoc.
Každé korektné riešenie, nech je ľubovoľne pomalé, môže získať 3 body.

V tejto úlohe budeme pri hodnotení klásť dôraz na *dôkaz správnosti* – ak nie je očividné, že vaše riešenie funguje, a neuvediete dostatočný dôkaz jeho správnosti, môžete stratiť veľa bodov.

Príklad

vstup	výstup
3 1 4 15 18 1000 3	2

Tretí kvietok kvitne cez polnoc: od času 1000 až po 10^{18} a následne ďalej od času 0 po čas 3 v nasledujúci deň. Všetky tri kvietky nikdy nebudú kvitnúť naraz, preto Šandyna potrebuje spraviť aspoň dve snímky. A to skutočne stačí. Môže napríklad odfotiť kvety v čase 3 (kvitne prvý a tretí) a v čase 16 (kvitne druhý).

A-III-3 Stromochod

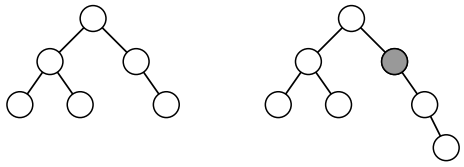
Súťažná úloha, podúloha A (1 bod)

Strom voláme *cesta doľava* ak žiaden vrchol nemá pravého syna. Naprogramujte stromochod tak, aby nastavil značku `ok` v koreni na `true` alebo `false` podľa toho, či je dotyčný strom cestou doľava. Bod za túto podúlohu dostane len riešenie obsahujúce korektný program.

Súťažná úloha, podúloha B (2 bod)

Na vstupe je strom ktorý je cestou doľava. Navyše platí, že jeho počet vrcholov je nepárny. Na začiatku má každý vrchol značku `stred` nastavenú na `false`. Naprogramujte stromochod tak, aby nastavil značku `stred` na `true` len vo vrchole, ktorý leží v strede cesty. Ľubovoľne pomalé korektné riešenie môže dostať 2 body.

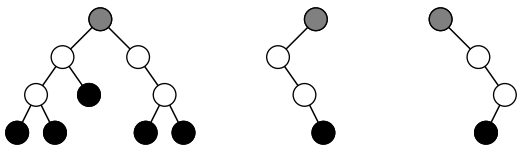
Súťažná úloha, podúloha C (7 bodov)



Ľavý podstrom vrcholu v tvorí jeho ľavý syn a všetky vrcholy, do ktorých sa ide cez dotyčného ľavého syna. (Ak v ľavého syna nemá, jeho ľavý podstrom je prázdny.) Analogicky definujeme aj *pravý podstrom*. *Veľkosťou* podstromu je počet vrcholov, ktoré obsahuje.
Strom voláme *dokonale vyvážený*, ak v každom vrchole platí, že sa veľkosti jeho ľavého a pravého podstromu líšia nanajvýš o 1. Na obrázku sú dva stromy. Ľavý je dokonale vyvážený, pravý nie je: pre sivý vrchol platí, že jeho ľavý podstrom je prázdny, zatiaľ čo pravý podstrom obsahuje dva vrcholy.
Naprogramujte stromochod tak, aby nastavil značku `ok` v koreni na `true` alebo `false` podľa toho, či je dotyčný strom dokonale vyvážený.
Počet bodov závisí od časovej zložitosti vášho riešenia. Program nemusí obsahovať implementáciu technických detailov, v takom prípade by ale slovný popis riešenia mal byť dostatočne detailný na to, aby bolo jasné, ako by sa tieto detaily programovali a akú by mali časovú zložitosť.

Študijný text: stromochod

V tomto ročníku OI budeme programovať **stromochod** – zariadenie určené na výpočty na binárnych stromoch. Študijný text začneme definíciou toho, čo presne sú binárne stromy.
Graf sa skladá z konečnej množiny **vrcholov** a konečnej množiny **hrán**. Každá hrana spája dva vrcholy. Každé dva vrcholy sú spojené nanajvýš jednou hranou. **Cesta** je postupnosť *navzájom rôznych* vrcholov, v ktorej sú každé dva po sebe idúce vrcholy spojené hranou.
Strom je graf, v ktorom medzi každými dvoma vrcholmi vedie práve jedna cesta. Strom môžeme **zakoreniť** – jeden z jeho vrcholov prehlásiť za **koreň.** Pre každé dva vrcholy spojené hranou následne hovoríme, že ten z nich, čo je bližšie ku koreňu, je **otec** a ten druhý je jeho **syn**. Koreň je jediný vrchol, ktorý nemá otca. Vrcholy, ktoré nemajú žiadnych synov, nazývame **listy** stromu. **Podstrom** je časť stromu, ktorá je vytvorená daným vrcholom, jeho synmi, synmi jeho synov, a tak ďalej až k listom.



Binárny strom je špeciálny zakorenený strom. Každý vrchol v binárnom strome má najviac dvoch synov. Navyše (oproti všeobecnému binárnemu stromu) rozlišujeme **ľavých** a **pravých** synov – každý vrchol má teda nanajvýš jedného ľavého a nanajvýš jedného pravého syna. (Aj ak má vrchol len jedného syna, rozlišujeme, či je ľavý alebo pravý.)
Na obrázku vpravo vidíte niekoľko rôznych binárnych stromov. Stromy kreslíme tak, aby koreň bol úplne na vrchu obrázku. Na každom obrázku je koreň vyplnený sivou a všetky listy sú vyplnené čiernou farbou.

Stromochod slúži na riešenie problémov týkajúcich sa binárnych stromov. Môžeme si ho predstaviť ako počítač, ktorý sa pohybuje po strome. V každom okamihu výpočtu se nachádza v práve jednom z vrcholov stromu.
Pamäť stromochodu je ohraničená: môže si pamätať len konečné množstvo bitov informácie, teda v každom okamihu sa môže nachádzať len v jednom z konečného počtu stavov. Okrem toho stromochod smie zaznamenávať informácie len do navštívených vrcholov. Do každého vrchola sa taktiež zmestí len konečné množstvo informácie a tieto informácie môže stromochod čítať a zapisovať vždy len v tom vrchole, v ktorom sa práve nachádza. Údaje uložené vo vrchole budeme nazývať **značky.**
Stromochod môžeme programovať v ľubovoľnom bežnom programovacom jazyku. Kvôli ohraničeniu na konečný počet stavov môžeme používať len celočíselné premenné s vopred daným rozsahom (napríklad 0..9) a booleovské premenné pre pravdivostné hodnoty. To napríklad znamená, že nie je možné používat polia ani smerníky. Z príkazov sú k dispozícii podmienky, cykly a goto. Procedúry a funkcie je možné používať, ale na ich parametre sa vzťahujú rovnaké ohraničenia na typy a navyše je zakázaná rekurzia. Pozor, použité rozsahy premenných **nesmú závisieť** od veľkosti stromu.
K aktuálnemu vrcholu (teda k tomu, v ktorom sa práve nachádza) vie stromochod pristupovať pomocou špeciálnej premennej „`V`“. Tá sa správa ako záznam (napr. `record` v Pascale, `struct` v C) ktorý obsahuje značky uložené v aktuálnom vrchole. Množinu značiek si môžete určiť vy pri písaní programu, ale musí ich byť konštantný počet a opäť platí ohraničenie na povolené typy.
Ak chceme stromochod presunúť na ľavého syna, pravého syna, resp. otca aktuálneho vrcholu, zavoláme funkciu `krok_l`, `krok_p`, resp. `krok_o`. Táto funkcia nemá žiadne parametre ani návratovú hodnotu. Ak sa pokúsite presunúť do neexistujúceho vrcholu, program sa zastaví s chybou. Preto sa hodí vopred otestovať, či syn alebo otec aktuálneho vrcholu existuje. Na to slúžia funkcie `ex_l`, `ex_p` a `ex_o`. Tieto pomocné funkcie nemajú žiadne parametre a vracajú booleovskú hodnotu.

Výpočet stromochodu prebieha následovne. Na vstupe dostaneme nejaký strom. Stromochod umiestnime **do jeho koreňa** a všetky značky v celom strome **vynulujeme** (teda číselné premenné nastavíme na nulu a boolovské na false). Výnimkou môžu byť značky, o ktorých je výslovne povedané, že sú súčasťou vstupu. Potom sa rozbehne program. Počas behu programu sa stromochod prechádza po strome, mení svoj stav (svoje lokálne premenné) a údaje zapísané vo vrcholoch stromu (značky). Keď beh programu korektne skončí, jeho výstup si prečítame z určených značiek.

Za čas behu programu na konkrétnom strome budeme považovať počet presunov po ňom. Časová zložitosť programu je funkcia, ktorá pre n vráti maximálny čas behu dotyčného programu na n-vrcholovom strome. Pamäťovú zložitosť neposudzujeme, pretože všetky programy si pamätajú lineárne množstvo informácie.

Na záver si ukážeme, ako stromochod naprogramovať tak, aby pomocou prehľadávania do hĺbky navštívil všetky vrcholy stromu a do každého umiestnil značku. Časová zložitosť tohoto programu je lineárna od počtu vrcholov, pretože každý vrchol navštívime najviac trikrát.

```
var zbytocna: 0..47;      { ukazka lokalnej premennej stromochodu }
type vrchol = record      { typ "vrchol" popisuje, ake znacky ukladame do vrcholov }
   bol_som_tu: boolean;   { tuto znacku chceme nastavit na true vo vsetkych vrcholoch }
   stav: 0..3;            { pocitadlo kolkokrat sme uz tento vrchol navstivili }
end;                      { na zaciatku je v kazdom vrchole bol_som_tu = false a stav = 0 }
begin
   zbytocna := 0;
   while zbytocna < 7 do begin { nekonecny cyklus }
      V.bol_som_tu := true;
      V.stav := V.stav + 1;
      case V.stav of
         1: if ex_l then krok_l;
         2: if ex_p then krok_p;
         3: if not ex_o then halt else krok_o;
      end;
   end;
end.
```

TRIDSIATY DRUHÝ ROČNÍK OLYMPIÁDY V INFORMATIKE

Príprava úloh: Michal Anderle, Eduard Batmendijn, Michal Forišek, Jaroslav Petrucha
Recenzia: Michal Forišek
Slovenská komisia Olympiády v informatike
Vydal: IUVENTA – Slovenský inštitút mládeže, Bratislava 2017