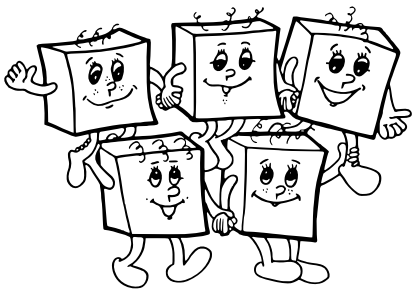
OLYMPIÁDA V INFORMATIKE

NA STREDNÝCH ŠKOLÁCH

http://oi.sk/



dvadsiaty ôsmy ročník
školský rok 2012/2013

zadania celoštátneho kola, deň 2
kategória A

Priebeh celoštátneho kola

Celoštátne kolo 28. ročníka Olympiády v informatike, kategórie A, sa koná v dňoch 20. – 23. marca 2013. Na riešenie úloh druhého, praktického dňa majú súťažiaci 4,5 hodiny čistého času. Akékoľvek pomôcky okrem písacích potrieb (napr. knihy, výpisy programov, kalkulačky) sú zakázané.

Čo má obsahovať riešenie úlohy?

- Skompilovateľný program v jazyku Pascal, C, alebo C++. Ak sa váš program nepodarí na našom testovacom počítači skompilovať, bude automaticky hodnotený 0 bodmi.

Hodnotenie riešení druhého (praktického) dňa

Za každú úlohu môžete získať od 0 do 15 bodov.

Po ukončení súťaže zoberieme pre každú úlohu váš posledný odovzdaný program a ten automaticky otestujeme na vopred pripravených testovacích vstupoch.

Testovanie na každom vstupe prebieha samostatne. Spustíme váš program a na štandardný vstup mu dáme konkrétne vstupné údaje. Hovoríme, že váš program daný vstup vyriešil, ak splní nasledujúce kritériá:

- Skončí skôr ako uplynie stanovený časový limit.
- Neprekročí stanovený pamäťový limit.
- Skončí korektne, nie chybou počas behu.
- Dáta, ktoré vypíše na štandardný výstup, tvoria korektný výstup, zodpovedajúci danému vstupu.
- Nebude používať žiadne funkcie zakázané kvôli bezpečnosti testovacieho systému.

Ku každej úlohe máme pripravených 15 sád testovacích vstupov. Sada vstupov pozostáva z jedného alebo viacerých testovacích vstupov. Za každú sadu vstupov, ktorej všetky vstupy (každý zvlášť) váš program správne vyrieši, získate jeden bod.

Sady vstupov sú navrhované tak, aby každé korektné riešenie získalo nejaké body, bez ohľadu na to, ako pomalé je. Bližšie informácie o testovacích dátach nájdete na konci zadania každej úlohy.

A-III-4 Špiónske voľby

Špiónska sieť v Absurdistane síce (v domácom kole OI) utrpela citeľné straty, medzi časom sa však opäť rozvinula do pôvodných rozmerov. Pripomeňme si, že kvôli lepšiemu utajeniu je táto sieť špiónov *riedka*: pre každú podmnožinu špiónov platí, že ak označíme jej veľkosť k, tak v nej bude existovať nanajvýš $3k$ dvojíc špiónov, ktorí sa poznajú. (Poznanie sa je vždy vzájomné.)

Onedlho majú prebehnúť voľby kráľa špiónov. Vo voľbách je päť kandidátov. Pre väčšie utajenie sú označení číslami 0, 1, 2, 3 a 4. Špiónov je n a majú čísla od 0 po $n-1$, vrátane.

Špiónske voľby sa vyhodnocujú inak ako tradičné. Najskôr si samozrejme každý špión vyberie práve jedného kandidáta, ktorého podporuje. Následne každý kandidát dostane toľko bodov, koľko existuje dvojíc špiónov takých, že sa poznajú a obaja daného kandidáta podporujú. Voľby samozrejme vyhrá ten kandidát, ktorý dostane najviac bodov.

Príklad. Majme štyroch špiónov: Pankráca, Serváca, Bonifáca a Medarda. Pankrác, Servác a Bonifác sa všetci navzájom poznajú. Navyše sa ešte poznajú Bonifác s Medardom.

Nech Pankrác a Servác volia kandidáta 0, zatiaľ čo Bonifác a Medard volia kandidáta 1. V tejto situácii by mal kandidát 0 jeden bod (za dvojicu Pankrác-Servác) a kandidát 1 tiež jeden bod (za dvojicu Bonifác-Medard).

Nech teraz Pankrác a Medard volia kandidáta 0, zatiaľ čo Servác a Bonifác volia kandidáta 1. V tejto situácii by mal kandidát 1 jeden bod, ale kandidát 0 by nemal žiaden (lebo Pankrác a Medard sa nepoznajú).

Ak by všetci štyria naši špióni volili toho istého kandidáta, ten by dostal štyri body.

Súťažná úloha

Prebieha predvolebná kampaň a špióni často menia názor na to, koho budú voliť. Vašou úlohou je napísať knižnicu, ktorá bude sledovať tieto zmeny názoru a priebežne informovať o počte bodov, ktorý majú jednotliví kandidáti.

Sieť špiónov sa počas behu vášho programu meniť nebude. Teda na začiatku sa dozviete, ktoré dvojice špiónov sa navzájom poznajú, a nik z nich počas behu programu nikoho nového nespozná.

Formát implementácie

Implementujete a odovzdávate (buď v C++ alebo v Pascale) knižnicu obsahujúcu tri funkcie: `spioni`, `zmen_nazor` a `pocet_bodov`.

Pri testovaní vašu knižnicu zlinkujeme s testovacím programom, ktorý sme pripravili my. Testovací program najskôr *jedenkrát* zavolá vašu funkciu `spioni`, čím vašej knižnici oznámi, ako vyzerá sieť špiónov a kto z nich na začiatku volí ktorého kandidáta. (Význam parametrov je popísaný nižšie.)

Následne bude testovací program *veľakrát v ľubovoľnom poradí* volať zvyšné dve funkcie. Volaním funkcie `zmen_nazor(s,v)` vašej knižnici náš program oznámi, že špión `s` odteraz volí kandidáta `v`. Pri volaní funkcie `pocet_bodov(p)` má vaša funkcia do poľa `p` vyplniť aktuálne počty bodov všetkých piatich kandidátov.

Hlavičky vašich funkcií v C++:

```
void spioni (int n, int m, int dvojice[][2], int voli[]);
void zmen_nazor (int spion, int voli);
void pocet_bodov (int pocet[5]);
```

Pomocné deklarácie a hlavičky vašich funkcií v Pascale:

```
type dvojica = array[0..1] of longint;
type vysledky = array[0..4] of longint;

procedure spioni (n, m : longint; dvojice : array of dvojica; voli: array of longint);
procedure zmen_nazor (spion, voli : longint);
procedure pocet_bodov (var pocet : vysledky);
```

Obmedzenia

- Pri volaní funkcie `spioni` bude platiť:
  - `n` je celkový počet špiónov (očíslovaných od 0 do $n-1$)
  - `m` je počet dvojíc, ktoré sa poznajú, platí $0 \leq m \leq 3n$ (lebo viac ich byť nevie)
  - `dvojice` je zoznam dvojíc, ktoré sa poznajú:
    pre každé i od 0 po $m-1$ platí, že sa poznajú špióni `dvojice[i][0]` a `dvojice[i][1]`
  - `voli` udáva kto koho na začiatku volí: pre každé i od 0 po $n-1$ špión i volí kandidáta `voli[i]`.
- Celkový počet volaní funkcií `zmen_nazor` a `pocet_bodov` neprevýši 10^6.
- V testovacích sadách 1, 2 a 3 platí $n \leq 100$.
- V testovacích sadách 4 a 5 platí $n \leq 100\,000$ a funkciu `pocet_bodov` náš program zavolá len raz.
- V testovacích sadách 6 až 10 platí $n \leq 100\,000$ a navyše platí nasledovná podmienka: Nech a a b je ľubovoľná dvojica špiónov, ktorí sa poznajú. Potom aspoň jeden zo špiónov a a b pozná najviac 5 iných špiónov (teda dokopy ich pozná najviac 6).
- V testovacích sadách 11 až 15 platí $n \leq 100\,000$.

Písanie a testovanie riešení

Dostanete od nás súbor `spioni.cc`, resp. `spioni.pas`. Tento súbor obsahuje kostru knižnice, ktorú máte implementovať. Vo vašom riešení stačí do neho doplniť implementácie funkcií `spioni`, `zmen_nazor` a `pocet_bodov`. Môžete si samozrejme definovať ďalšie pomocné funkcie a lokálne premenné. (V C++ odporúčame všetky globálne premenné deklarovať ako `static`.)

Ďalej dostanete od nás súbor `Makefile` a súbor `main.cc`, resp. `main.pas`. Súbor `main.*` obsahuje ukážkový hlavný program, ktorý môžete použiť na testovanie vášho riešenia. Súbor `Makefile` obsahuje inštrukcie na ich skompilovanie. Stačí na príkazovom riadku napísať `make` a spustia sa správne príkazy, ktoré, ak všetko dobre zafunguje, vyrobia spustiteľný súbor `main`.

Tento súbor očakáva na štandardnom vstupe najskôr celé čísla n a m, potom zoznam m dvojíc čísel špiónov, ktorí sa poznajú, potom zoznam n čísel kandidátov, ktorých na začiatku volia jednotliví špióni. S týmito parametrami zavolá náš program vašu funkciu `spioni`.

Následne môžete na štandardný vstup písať medzerami oddelené príkazy tvaru „`N s v`“ a „`P`“. Po prečítaní príkazu prvého typu náš program zavolá `zmen_nazor(s,v)`, po prečítaní príkazu druhého typu náš program zavolá `pocet_bodov` a vypíše na štandardný výstup jeden riadok obsahujúci vrátené hodnoty.

Príklad

vstup

```
4 4
0 1   1 2   2 0   2 3
0 0 1 0
P
N 2 0
P
N 3 1
P
N 2 1
P
```

výstup

```
pocet = 1, 0, 0, 0, 0
pocet = 4, 0, 0, 0, 0
pocet = 3, 0, 0, 0, 0
pocet = 1, 1, 0, 0, 0
```

Vľavo je príklad údajov, ktoré by ste mohli poslať na štandardný vstup skompilovaného programu.

Sieť špiónov je tá z príkladu v zadaní (0 je Pankrác, 1 Servác, 2 Bonifác, 3 Medard). Na začiatku sa zavolá vaša funkcia `spioni` *s popisom tejto siete špiónov a informáciou, že na začiatku Bonifác volí kandidáta 1 a ostatní volia kandidáta 0.*

Následne sa 4× zavolá vaša funkcia `pocet_bodov` *a medzi jednotlivými jej volaniami sa zavolá* `zmen_nazor(2,0)`, `zmen_nazor(3,1)` *a* `zmen_nazor(2,1)`. *Teda postupne sa Bonifác rozhodne voliť kandidáta 0, Medard kandidáta 1, a aj Bonifác kandidáta 1.*

A-III-5 Uhlopriečky

Usamec a Maru majú obaja voľnú hodinu, a tak si dlhú chvíľu krátia zábavnou hrou. Aby nedošlo k nejakým nedorozumeniam, radšej vám povieme, akou.

Hra začína tým, že Maru zoberie čistý list papiera a čiernou farbou naň nakreslí pravidelný n-uholník. Potom doň, opäť čiernou farbou, dokreslí $n-3$ uhlopriečok. Tie však nesmie kresliť len tak, ako sa jej zachce. Musí dodržať dve pravidlá:

1. Žiadne dve uhlopriečky sa nesmú pretínať. (Rozmyslite si, že takýmto dokreslením uhlopriečok Maru určite rozdelí n-uholník na presne $n-2$ trojuholníkov, nič iné nemôže vzniknúť.)
2. Maru musí vyberať uhlopriečky tak, aby mal každý z dotyčných $n-2$ trojuholníkov aspoň jednu stranu spoločnú s pôvodným n-uholníkom.

Keď Maru dokreslí uhlopriečky, začína sa samotná hra. V tej Usamec a Maru striedavo ťahajú (začína Usamec). Ťah spočíva v tom, že si hráč vyberie čiernu úsečku (buď jednu z uhlopriečok, alebo jednu zo strán n-uholníka) a prefarbí ju na červeno. Hru vyhráva hráč, ktorý ako prvý vytvorí červený trojuholník.

Súťažná úloha

Vašou úlohou je napísať knižnicu, ktorá bude túto hru hrať namiesto Usamca (prvého hráča). Body dostanete, ak vaša knižnica hru vyhrá. V niektorých hrách bude Maru ťahať náhodne, vo väčšine však bude hrať optimálne – teda ak spravíte chybu, prehráte.

Formát implementácie

Implementujete a odovzdávate (buď v C++ alebo v Pascale) knižnicu obsahujúcu tri funkcie: `hraci_plan`, `tah_usamca` a `tah_maru`.

Pri testovaní vašu knižnicu zlinkujeme s testovacím programom, ktorý sme pripravili my. Testovací program najskôr *jedenkrát* zavolá vašu funkciu `hraci_plan`, čím vašej knižnici oznámi, ako vyzerá hrací plán, ktorý nakreslila Maru – teda sa dozviete počet vrcholov n a zoznam nakreslených uhlopriečok.

Následne bude testovací program hrať proti vašej knižnici vyššie popísanú hru, pričom vaša knižnica hrá za Usamca, zatiaľ čo testovací program hrá za Maru. Testovací program bude teda *dokola volať striedavo* vašu funkciu `tah_usamca` a vašu funkciu `tah_maru`.

Pri každom volaní vašej funkcie `tah_usamca` musí táto vrátiť ťah, ktorý chcete spraviť – teda čísla dvoch vrcholov spojených čiernou hranou, ktorú chcete prefarbiť na červenú. (Na poradí čísel vrcholov nezáleží.) Pri každom volaní vašej funkcie `tah_maru` vám v jej parametri testovací program oznámi dve čísla vrcholov určujúce hranu, ktorú na červeno prefarbila Maru.

Akonáhle jeden z hráčov vytvorí červený trojuholník, testovací program podá príslušnú správu a skončí. Ak sa pri niektorom volaní vašej funkcie `tah_usamca` pokúsite spraviť neplatný ťah, testovací program tiež okamžite skončí a automaticky prehrávate. (Neplatný ťah môže byť použitie neexistujúceho čísla vrcholu, použitie dvoch čísel vrcholov ktoré nie sú spojené vôbec, alebo takých dvoch, ktoré už sú spojené červenou hranou.)

Hlavičky vašich funkcií v C++:

```
void hraci_plan (int n, int uhlopriecky[][2]);
void tah_usamca (int hrana[2]);
void tah_maru   (int hrana[2]);
```

Pomocné deklarácie a hlavičky vašich funkcií v Pascale:

```
type dvojica = array[0..1] of longint;

procedure hraci_plan (n : longint; uhlopriecky : array of dvojica);
procedure tah_usamca (var hrana : dvojica);
procedure tah_maru   (hrana : dvojica);
```

Obmedzenia

- V každom z použitých testovacích vstupov na začiatku hry existuje vyhrávajúca stratégia pre Usamca.
- Pri volaní funkcie `hraci_plan` bude platiť:
  - `n` je počet vrcholov mnohouholníka (platí $n \geq 3$, vrcholy sú očíslované 1 až n po obvode)
  - pre každé i od 0 po $n-4$ sú vrcholy `uhlopriecky[i][0]` a `uhlopriecky[i][1]` spojené hranou
- V testovacích sadách 1 až 3 hrá Maru náhodne: v každom ťahu si vyberie náhodnú čiernu hranu a tú prefarbí na červeno. V jednotlivých sadách platí $n \leq 100$, $n \leq 10\,000$ a $n \leq 100\,000$.
- V testovacích sadách 4 až 15 Maru hrá optimálne – akonáhle spravíte zlý ťah, hru už nevyhráte.
- V testovacích sadách 4 až 6 platí $n \leq 10$.
- V testovacích sadách 7 až 10 platí $n \leq 100\,000$ a navyše majú úplne všetky uhlopriečky, ktoré Maru na začiatku nakreslila, spoločný vrchol.
- V testovacích sadách 11 až 15 platí $n \leq 100\,000$.

Písanie a testovanie riešení

Dostanete od nás súbor `uhlopriecky.cc`, resp. `uhlopriecky.pas`. Tento súbor obsahuje kostru knižnice, ktorú máte implementovať. Vo vašom riešení stačí do neho doplniť implementácie funkcií `hraci_plan`, `tah_usamca` a `tah_maru`. Môžete si samozrejme definovať ďalšie pomocné funkcie a lokálne premenné. (V C++ odporúčame všetky globálne premenné deklarovať ako `static`.)

Ďalej dostanete od nás súbor `Makefile` a súbor `main.cc`, resp. `main.pas`. Súbor `main.*` obsahuje ukážkový hlavný program, ktorý môžete použiť na testovanie vášho riešenia. Súbor `Makefile` obsahuje inštrukcie na ich skompilovanie. Stačí na príkazovom riadku napísať `make` a spustia sa správne príkazy, ktoré, ak všetko dobre zafunguje, vyrobia spustiteľný súbor `main`.

Tento program očakáva ako parameter názov vstupného súboru. Z toho načíta popis hracieho plánu: najskôr číslo n a následne $n-3$ dvojíc čísel vrcholov spojených uhlopriečkou. Následne bude tento program s vami komunikovať cez štandardný vstup a výstup, čím vám umožní ručne hrať proti vašej knižnici. Vždy najskôr zavolá funkciu `tah_usamca`, jej výstup vypíše na štandardný výstup, zo štandardného vstupu načíta váš ťah (t.j. ťah Maru) a ten volaním funkcie `tah_maru` oznámi vašej knižnici.

Príklad

Vstupný súbor `vstup.txt` obsahuje nasledovné údaje:

```
6
1 3    3 6    6 4
```

Tomuto zodpovedá hrací plán na obrázku vpravo.
Keď spustíme „`./main vstup.txt`“, môžeme sa zahrať. Jeden možný priebeh hry:

```
volanie tah_usamca vrátilo:  3 1
užívateľ zadal z klávesnice: 4 6
volanie tah_usamca vrátilo:  3 4
užívateľ zadal z klávesnice: 1 6
volanie tah_usamca vrátilo:  3 6
Usamec vyhral
```

Po ťahu „3 6“ hra končí, keďže vznikol červený trojuholník. (Dokonca dva: 136 aj 346.) Poznamenajme, že ani jeden z hráčov nehral v tomto príklade optimálne.

DVADSIATY ÔSMY ROČNÍK OLYMPIÁDY V INFORMATIKE

Príprava úloh: Michal Anderle, Vladimír Boža, Michal Forišek, Ján Hozza, Marek Špano
Recenzia: Michal Forišek
Slovenská komisia Olympiády v informatike
Vydal: IUVENTA – Slovenský inštitút mládeže, Bratislava 2013