# 62. ročník Matematické olympiády – 2012/2013

*Úlohy ústředního kola kategorie P – 2. soutěžní den*

---

Na řešení úloh máte 4,5 hodiny čistého času. Při soutěži je zakázáno používat jakékoliv pomůcky kromě psacích potřeb a přiděleného počítače (tzn. knihy, kalkulačky, mobily, apod.).

Řešením každého příkladu je zdrojový kód programu zapsaný v programovacím jazyce Pascal, C nebo C++. Po skončení soutěže bude pro každou úlohu váš poslední odevzdaný program automaticky otestován pomocí předem připravených 15 sad testovacích vstupních dat. Každá sada je tvořena jedním nebo několika testovacími vstupy. Jeden bod získáte za každou sadu vstupů, kterou váš program celou správně vyřeší – tzn. pro každý testovací vstup z této sady dá program správný výsledek ve stanoveném časovém a paměťovém limitu. Za každou úlohu tedy můžete získat 0 až 15 bodů.

Sady vstupů jsou navrženy tak, aby každé korektní řešení získalo nějaké body, i když je založeno na neefektivním algoritmu. Časové a paměťové limity zajišťují, že plný počet bodů získá pouze efektivní řešení. Podrobnější informace o testovacích datech najdete na konci zadání každé úlohy.

*obraťte list*

## P-III-4 Volby

I přes vaši pomoc v domácím kole síť špiónů v království krále Filipa stále existuje, a dokonce připravuje povstání. K jeho provedení už zbývá jen dohodnout se, kdo bude novým králem. Brzy se proto budou konat špiónské volby.

Špiónské volby se trochu liší od těch běžných. Stejně jako v běžných volbách je několik kandidátů na nového krále (v těch aktuálních je jich pět, známých pod přezdívkami 0, 1, 2, 3 a 4) a každý ze špiónů si vybere jednoho z nich. Zcela jinak se ale určuje, kdo volby vyhraje. Připomeňme, že každý špión zná typicky jen několik málo dalších špionů – v každé podmnožině špiónů $X$ je nanejvýš $3 \cdot |X|$ dvojic špiónů, kteří se znají. Každý kandidát $k$ dostane tolik bodů, kolik existuje dvojic špionů, kteří se navzájem znají a oba si zvolili kandidáta $k$. Vyhrává kandidát, který získá nejvíce bodů.

Například, nechť špióni jsou Pankrác, Servác, Bonifác a Medard, kde se navzájem znají Pankrác se Servácem, Servác s Bonifácem, Pankrác s Bonifácem a Bonifác s Medardem. Jestliže Pankrác, Servác a Medard volí kandidáta 0 a Bonifác kandidáta 1, pak kandidát 0 získá 1 bod (za dvojici Pankrác, Servác) a kandidát 1 získá 0 bodů. Kdyby ale kandidáta 0 volili Pankrác, Servác a Bonifác, zatímco Medard by volil kandidáta 1, pak by kandidát 0 získal 3 body (za dvojice Pankrác, Servác; Servác, Bonifác; Pankrác, Bonifác) a kandidát 1 by získal 0 bodů.

### Soutěžní úloha

Špióni kují pikle a jimi preferovaní kandidáti se neustále mění. Napište program, který by veřejnosti pomohl orientovat se v komplikované volební situaci.

### Popis vstupu a výstupu

Vytvořte knihovnu `volby.c`, `volby.cpp` nebo `volby.pas` implementující následující procedury:

```
/* V C či C++ */
void spioni(int n, int m, int znaji_se[][2], int voli[]);
void zmen_nazor(int spion, int voli);
void pocet_bodu(int pocet[5]);
```

```
{ V Pascalu }
type dvojice = array [0..1] of longint;
type vysledky = array [0..4] of longint;
procedure spioni(n, m: longint; znaji_se: array of dvojice; voli: array of longint);
procedure zmen_nazor(spion, voli: longint);
procedure pocet_bodu(var pocet: vysledky);
```

Tato knihovna bude ve vyhodnocovači slinkována s hlavním programem, který nejprve zavolá proceduru `spioni` s parametry $n \geq 1$ (počet špiónů), $m$ (počet dvojic špiónů, kteří se navzájem znají, kde $0 \leq m \leq 3n$), `znaji_se` (seznam $m$ dvojic špiónů, kteří se navzájem znají) a pole `voli`, kde `voli[i]` je číslo kandidáta, kterého volí špión číslo $i$. Špióni jsou očíslováni od 0 do $n-1$ a kandidáti od 0 do 4.

Poté hlavní program opakovaně a v libovolném pořadí volá vaše procedury `zmen_nazor` a `pocet_bodu`. Celkový počet volání procedur `zmen_nazor` a `po-`

`cet_bodu` bude nejvýše 1 000 000. Jejich význam je tento: Od okamžiku zavolání procedury `zmen_nazor(s, k)` volí špión $s$ kandidáta $k$. Procedura `pocet_bodu(pocet)` vyplní do pole `pocet` výsledky voleb dle aktuálních preferencí špiónů, pro $i = 0, \ldots, 4$ tedy do položky `pocet[i]` vyplní aktuální počet hlasů kandidáta $i$ dle pravidel špiónských voleb.

V adresáři `/mo/templates/volby/` máte k dispozici ukázkové soubory `volby.{c,cpp,pas}`, které obsahují základní kostru pro požadovanou knihovnu. Ve vašem řešení stačí doplnit do nich definice procedur `spioni`, `zmen_nazor` a `pocet_bodu`. Můžete samozřejmě definovat i další pomocné procedury, funkce a proměnné (v C a C++ je doporučujeme deklarovat `static`).

Dále máte k dispozici soubory `volby-main.{c,cpp,pas}` obsahující ukázkový hlavní program, který můžete používat k testování svého řešení. Příkaz `compile volby` automaticky slinkuje vaše řešení s odpovídajícím hlavním programem a vytvoří spustitelný soubor `volby`. Ten očekává na standardním vstupu nejprve celá čísla $n$ a $m$, následovaná $m$ dvojicemi čísel špiónů, kteří se navzájem znají, a $n$ čísly, udávajícími kandidáty, které špióni na začátku volí. Dále ze standardního vstupu čte posloupnost příkazů, které mohou být: `N s v` – zavolá `zmen_nazor(s, v)`, nebo `P` – zavolá `pocet_bodu` a vypíše vrácené hodnoty.

**Hodnocení**

Ve vstupech, za něž lze získat až 2 body, bude $n \leq 100\,000$ a procedura `pocet_bodu` volána pouze jednou. Ve vstupech, za něž lze získat další 2 body, bude $n \leq 100$. Dalších 5 bodů lze získat za vstupy, v nichž $n \leq 100\,000$ a pro každou dvojici špiónů $a, b$, kteří se znají, platí, že buď $a$ nebo $b$ zná celkem nejvýše 6 špionů. Ostatní vstupy (za něž lze získat dalších 6 bodů) splňují $n \leq 100\,000$.

**Příklad**

| *Vstup:* | *Výstup:* |
|---|---|
| `spioni(4, 4,`<br>`  {{0,1},{1,2},{2,0},{2,3}},`<br>`  {0,0,1,0});` | |
| `pocet_bodu(pocet);` | `pocet = {1, 0, 0, 0, 0}` |
| `zmen_nazor(2, 0);` | |
| `pocet_bodu(pocet);` | `pocet = {4, 0, 0, 0, 0}` |
| `zmen_nazor(3, 1);` | |
| `pocet_bodu(pocet);` | `pocet = {3, 0, 0, 0, 0}` |
| `zmen_nazor(2, 1);` | |
| `pocet_bodu(pocet);` | `pocet = {1, 1, 0, 0, 0}` |

## P-III-5 Gamesa

Princ Honzík hraje se svou kamarádkou Mařenkou následující hru. Mařenka nejprve nakreslí herní plán: začne s pravidelným $n$-úhelníkem a dokreslí do něj $n-3$ úhlopříček rovnými a neprotínajícími se hranami, čímž jeho vnitřek rozdělí na $n-2$ trojúhelníků. Podle pravidel navíc tyto úhlopříčky musí volit tak, aby každý ze vzniklých trojúhelníků sdílel alespoň jednu hranu s počátečním $n$-úhelníkem.

Honzík a Mařenka pak střídavě barví hrany herního plánu, a to jak hrany $n$-úhelníka, tak úhlopříčky. Honzík začíná a oba k barvení používají červenou barvu. Vyhrává ten, kdo první vytvoří červený trojúhelník.

### Soutěžní úloha

Napište program, který pomůže Honzíkovi vyhrát. Ve svém řešení můžete předpokládat, že pro Mařenčin herní plán má Honzík vyhrávající strategii.

### Popis vstupu a výstupu

Vytvořte knihovnu implementující následující procedury:

```
/* V C či C++ */
void herni_plan(int n, int uhlopricky[][2]);
void tah_honzika(int hrana[2]);
void tah_marenky(int hrana[2]);
```

```
type dvojice = array [0..1] of longint;
procedure herni_plan(n: longint; uhlopricky: array of dvojice);
procedure tah_honzika(var hrana: dvojice);
procedure tah_marenky(hrana: dvojice);
```

Tato knihovna bude ve vyhodnocovači slinkována s hlavním programem simulujícím chování Mařenky. Ten nejprve zavolá proceduru `herni_plan`, jejíž parametry určují počet hran $n \geq 3$ hracího plánu a jeho úhlopříčky. Vrcholy hracího plánu jsou číslovány od 1 do $n$ a Mařenka nakreslí úhlopříčky mezi vrcholy `uhlopricky[i][0]` a `uhlopricky[i][1]` pro $i = 0, \ldots, n-4$.

Poté hlavní program střídavě volá procedury `tah_honzika` a `tah_marenky`. V proceduře `tah_honzika` si vyberete hranu, kterou budete chtít obarvit, a vrátíte ji v poli hrana – obarvíte tedy čáru mezi vrcholy `hrana[0]` a `hrana[1]`, což musí být buď jedna z hran mnohoúhelníka nebo jedna z úhlopříček na hracím plánu. Tato čára také nesměla být již předtím obarvena. Na pořadí čísel vrcholů hrany nezáleží, chcete-li tedy obarvit hranu 1–$n$, můžete zadat buď `hrana[0]=1` a `hrana[1]=n`, nebo `hrana[0]=n` a `hrana[1]=1`.

Volání procedury `tah_marenky` vás informuje o tahu Mařenky, tedy o tom, že obarvila čáru mezi vrcholy `hrana[0]` a `hrana[1]`.

Jakmile jeden z hráčů obarví všechny hrany v některém z trojúhelníků, hra končí a hlavní program již dále žádné procedury z vaší knihovny nevolá. Pokud některý z vašich tahů odporuje pravidlům (daná hrana není v herním plánu či již byla obarvena), automaticky prohrajete.

V adresáři `/mo/templates/gamesa/` máte k dispozici soubory `gamesa.c`, `gamesa.cpp` a `gamesa.pas`, které obsahují základní kostru pro požadavanou knihovnu.

Ve vašem řešení stačí do nich doplnit definice procedur `herni_plan`, `tah_honzika` a `tah_marenky`. Můžete samozřejmě definovat i další pomocné procedury, funkce a proměnné (v C a C++ je doporučujeme deklarovat `static`).

Dále máte k dispozici soubory `gamesa-main.{c,cpp,pas}` obsahující ukázkový hlavní program, který můžete používat k testování svého řešení. Příkaz `compile gamesa` automaticky slinkuje vaše řešení s odpovídajícím hlavním programem a vytvoří spustitelný soubor `gamesa`. Ten jako argument dostane soubor popisující herní plán a poté vám umožní hrát proti vaší knihovně – zobrazuje tedy, co vaše knihovna odpovídá na volání `tah_honzika`, a ze standardního vstupu čte dvojice, s nimiž má volat `tah_marenky`. Soubor popisující hrací plán obsahuje na prvním řádku číslo $n$ a na následujících $n-3$ řádcích dvojice čísel, určující vybrané úhlopříčky.

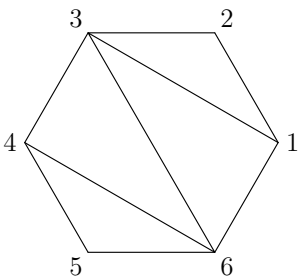
**Hodnocení**

Ve vstupech, za něž lze získat až 3 body, bude $n \leq 100\,000$ a Mařenka používá následující strategii: pokud jsou již v některém trojúhelníku dvě obarvené hrany, pak obarví třetí a vyhraje, jinak obarvní náhodnou ještě neobarvenou hranu.

Ve všech ostatních vstupech Mařenka používá optimální strategii, zahraje-li tedy Honzík chybně, Mařenka vyhraje. Ve vstupech, za něž lze získat další 3 body, bude $n \leq 10$. Další 4 body lze získat za vstupy, v nichž $n \leq 100\,000$ a všechny Mařenkou vybrané úhlopříčky sousedí s jedním vrcholem. Ostatní vstupy (za něž lze získat dalších 5 bodů) splňují $n \leq 100\,000$.

**Příklad**

Hlavní program nejprve zavolá funkci `herni_plan(6, {{1,3},{3,6},{6,4})`, čímž vašemu programu oznámí, že se hraje na následujícím plánu:




Poté střídavě volá procedury `tah_honzika` a `tah_marenky`, například takto:

| *Vstup:* | *Výstup:* |
|---|---|
| `tah_honzika(tah)` | `tah = {3,1}` |
| `tah_marenky({4,6});` | |
| `tah_honzika(tah)` | `tah = {3,4}` |
| `tah_marenky({1,6});` | |
| `tah_honzika(tah)` | `tah = {3,6}` |

Po svém posledním tahu Honzík vyhrál, neboť obarvil trojúhelník 346 (a 136). Poznamenejme, že žádný z hráčů v tomto příkladu nepoužíval optimální strategii.