Riešenia kategórie B

B-I-1 Hľadanie mín

Na reprezentáciu hracieho plánu použijeme dvojrozmerné pole. Pri načítaní vstupu si do neho zaznačíme polohu mín. Pri výpise výstupu potom pre každé políčko, ktoré neobsahuje mínu, prejdeme všetkých jeho susedov a zistíme počet tých, ktorí mínu obsahujú.

Šikovný trik, ako riešiť okraje hracieho plánu: namiesto toho, aby sme mali pole, ktorého riadky a stĺpce budú číslované od 1 po R, resp. od 1 po S, budeme mať pole v každom smere o políčko väčšie. Riadky teda budú mať čísla od 0 po $R+1$ a stĺpce od 0 po $S+1$. Nové políčka samozrejme žiadne míny neobsahujú.

Listing programu:

```
const MAX_ROZMER = 5000;

var miny : array[0..MAX_ROZMER+1, 0..MAX_ROZMER+1] of boolean;
    R, S, N, i, j, k, l, mr, ms, pocet : longint;

begin
   { nacitame rozmery }
   readln(R,S);

   { inicializujeme hraci plan: nikde nie je mina }
   for i:=0 to R+1 do for j:=0 to S+1 do miny[r,s]:=false;

   { nacitame jednotlive miny a zaznacime ich do pola }
   readln(N);
   for i:=1 to N do begin
      readln(mr,ms);
      miny[mr,ms] := true;
   end;

   { prechadzame cez policka a vyrabame vystup }
   for i:=1 to R do begin
      for j:=1 to S do begin
         if miny[i,j] then write('*') else begin
            pocet := 0;
            { prezrieme vsetkych susedov policka (i,j) }
            for k:=i-1 to i+1 do for l:=j-1 to j+1 do if miny[k,l] then inc(pocet);
            if pocet=0 then write('.') else write(pocet);
         end;
      end;
      writeln;
   end;
end.
```

Iné možné riešenie je zostrojovať výstup rovno pri načítaní vstupu. Na začiatku každé políčko inicializujeme na nulu. Vždy, keď načítame súradnice míny, zvýšime o 1 hodnoty na všetkých 8 susedných políčkach. Prítomnosť míny si môžeme zaznačiť napr. tak, že k hodnote políčka, kde mína leží, pripočítame 10.

B-I-2 Starosta

Najjednoduchšie riešenie tejto úlohy je načítať celý vstup do poľa a toto pole spracovať v dvoch prechodoch. V prvom prechode nájdeme všetkých Mórickovych známych. Na ich zapamätanie použijeme pole boolovských premenných, v ktorom si o každom dedinčanovi budeme pamätať, či sa s Mórickom pozná.

V druhom prechode vyplníme druhé pole boolovských premenných, v ktorom budeme mať o každom dedinčanovi uložené, či pozná nejakého Mórickovho známeho.

Listing programu:

```
const MAX_D = 1000000; MAX_Z = 5000000;

var vstup : array[1..MAX_Z, 1..2] of longint;
```

```
    znami1, znami2 : array[1..MAX_D] of boolean;
    D, Z, i, j : longint;

begin
   { nacitame vstup }
   read(D,Z);
   for i:=1 to Z do read(vstup[i,1],vstup[i,2]);

   { prejdeme vstup a zistime vsetkych Morickovych znamych }
   for i:=1 to D do znami1[i] := false;
   for i:=1 to Z do begin
      if vstup[i,1]=1 then znami1[ vstup[i,2] ] := true;
      if vstup[i,2]=1 then znami1[ vstup[i,1] ] := true;
   end;

   { znova prejdeme vstup a zistime vsetkych ich znamych }
   for i:=1 to D do znami2[i] := false;
   for i:=1 to Z do begin
      if znami1[ vstup[i,1] ] then znami2[ vstup[i,2] ] := true;
      if znami1[ vstup[i,2] ] then znami2[ vstup[i,1] ] := true;
   end;

   { vypiseme vsetkych, ktori nie su znami, v premennej j si pocitame, kolko ich bolo }
   j := 0;
   for i:=2 to D do
      if (not znami1[i]) and (not znami2[i]) then begin
         writeln(i);
         inc(j);
      end;
   if j=0 then writeln('starosta');
end.
```

Alternatívne riešenia

Predchádzajúce riešenie je bezkonkurenčne najjednoduchšie. Ukážeme si však aj iné možné prístupy, ktoré obsahujú viacero užitočných techník.

Základná myšlienka týchto riešení je jednoduchá: najskôr v cykle prejdeme všetkých Móríckovych známych a potom pomocou dvoch vnorených cyklov prejdeme pre každého známeho všetkých jeho známych. O každom z nich si zaznačíme, že už sme ho videli. Na záver už len prejdeme zoznam dedinčanov a vypíšeme všetkých, ktorých sme ani raz nevideli.

Na to, aby bolo naše riešenie dostatočne rýchle, si potrebujeme informácie o dedinčanoch pamätať v nejakej vhodnejšej podobe – prechádzať celý zoznam známostí toľkokrát, koľko známych má Móricko, už je predsa len priveľa zbytočnej práce.

Matica susednosti

Pre vstupy, kde je počet dedinčanov D nanajvýš pár tisíc, si môžeme dovoliť použiť maticu susednosti – teda dvojrozmerné pole, kde si pre každú dvojicu dedinčanov zaznačíme, či sa poznajú.

Listing programu:

```
const MAX_D = 5000;
var pozna : array[1..MAX_D, 1..MAX_D] of boolean;
    OK : array[1..MAX_D] of boolean;
    D, Z, i, j, k : longint;

begin
   { nacitame vstup a vyplnime si maticu susednosti }
   read(D,Z);
   for i:=1 to D do for j:=1 to D do pozna[i,j] := false;
   for i:=1 to Z do begin
      read(j,k);
      pozna[j,k] := true;
      pozna[k,j] := true;
   end;

   { najdeme vsetkych Morickovych znamych a zaznacime ich do pola OK }
   OK[1] := true;
   for i:=2 to D do OK[i] := pozna[1,i];

   { pre kazdeho Morickovho znameho najdeme jeho znamych a zaznacime aj tych }
```

```
   for i:=2 to D do
      if pozna[1,i] then
         for j:=2 to D do
            if pozna[i,j] then
               OK[j] := true;

   { vypiseme vsetkych, ktori nie su OK, v premennej k si pocitame, kolko ich bolo }
   k := 0;
   for i:=2 to D do
      if not OK[i] then begin
         writeln(i);
         inc(k);
      end;
   if k=0 then writeln('starosta');
end.
```

Zoznamy známych

Ešte lepšie riešenie je vytvoriť si zoznamy známych: Pre každého dedinčana x budeme mať zoznam, v ktorom budú uložení všetci dedinčania, ktorých x pozná.

Oproti predchádzajúcemu riešeniu má toto hneď dve výhody. Za prvé, spotreba pamäte je omnoho nižšia – pamätáme si len údaje, ktoré sú na vstupe, neplýtvame pamäťou na dvojice, ktoré sa nepoznajú.

(Aby sme boli presnejší, riešenie používajúce maticu susednosti potrebuje pamäť priamo úmernú hodnote D^2, teda počtu všetkých dvojíc, zatiaľ čo pri tomto riešení bude množstvo pamäte lineárne od Z, teda počtu dvojíc, ktoré sa poznajú.)

Za druhé, zlepší nám to aj časovú zložitosť – aj pre Móricka, aj pre každého jeho známeho budeme vedieť rovno vymenovať všetkých jeho známych, nebudeme musieť zbytočne kontrolovať všetkých dedinčanov.

Jeden možný spôsob, ako zoznamy známych uložiť, je použiť dátovú štruktúru *spájaný zoznam*.

Listing programu:

```
const MAX_D = 1000000;
type pznamy = ^tznamy;
     tznamy = record
        kto : longint;
        dalsi : pznamy;
     end;

var znami : array[1..MAX_D] of pznamy;
    p, q : pznamy;
    OK : array[1..MAX_D] of boolean;
    D, Z, i, j, k : longint;

begin
   { nacitame vstup a vyplnime si zoznamy znamych }
   read(D,Z);
   for i:=1 to D do znami[i] := nil;
   for i:=1 to Z do begin
      read(j,k);
      p := new(pznamy);  p^.kto := k;  p^.dalsi := znami[j];  znami[j] := p;
      p := new(pznamy);  p^.kto := j;  p^.dalsi := znami[k];  znami[k] := p;
   end;

   { najdeme vsetkych Morickovych znamych a zaznacime ich do pola OK }
   OK[1] := true;
   for i:=2 to D do OK[i] := false;
   p := znami[1];
   while p <> nil do begin
      OK[ p^.kto ] := true;
      p := p^.dalsi;
   end;

   { pre kazdeho znameho prejdeme vsetkych jeho znamych a zaznacime aj tych }
   p := znami[1];
   while p <> nil do begin
      q := znami[ p^.kto ];
      while q <> nil do begin
         OK[ q^.kto ] := true;
         q := q^.dalsi;
      end;
      p := p^.dalsi;
   end;
```

```
   { vypiseme vsetkych, ktori nie su OK, v premennej k si pocitame, kolko ich bolo }
   k := 0;
   for i:=2 to D do
      if not OK[i] then begin
         writeln(i);
         inc(k);
      end;
   if k=0 then writeln('starosta');
end.
```

Zoznamy známych v obyčajnom poli

V predchádzajúcom riešení sme použili spájané zoznamy, implementované pomocou ukazovateľov (pointrov). Vystačíme si však aj bez nich – rovnako efektívne riešenie sa dá naprogramovať aj s obyčajnými poľami. Moderné programovacie jazyky ponúkajú polia, ktorých veľkosť vieme nastaviť počas behu programu. (Napr. vo FreePascale/Delphi môžeme na nastavenie dĺžky poľa použiť `SetLength`.)

Takéto riešenie bude fungovať v dvoch fázach. V prvej načítame celý vstup do pomocného poľa a pre každého dedinčana spočítame jeho *stupeň* – teda počet ľudí, s ktorými sa pozná. Potom pre každého dedinčana vyrobíme pole príslušnej dĺžky a následne všetky tieto polia vyplníme potrebnými údajmi.

(V C++ máme k dispozícii dátovú štruktúru `vector`, pomocou ktorej je riešenie ešte pohodlnejšie – nepotrebujeme poznať veľkosť poľa vopred, stačí postupne na jeho koniec pridávať nové a nové prvky.)

Listing programu:

```
const MAX_D = 1000000; MAX_Z = 5000000;

var vstup : array[1..MAX_Z, 1..2] of longint;
    stupne : array[1..MAX_D] of longint;
    znami : array[1..MAX_D] of array of longint;
    OK : array[1..MAX_D] of boolean;
    D, Z, i, j : longint;

begin
   { nacitame vstup a spocitame stupne }
   read(D,Z);
   for i:=1 to D do stupne[i] := 0;
   for i:=1 to Z do begin
      read(vstup[i,1],vstup[i,2]);
      inc( stupne[ vstup[i,1] ] );
      inc( stupne[ vstup[i,2] ] );
   end;

   { nastavime velkosti poli a vyplnime ich }
   for i:=1 to D do setlength( znami[i], stupne[i] );
   for i:=1 to D do stupne[i]:=0;
   for i:=1 to Z do begin
      inc( stupne[ vstup[i,1] ] );
      znami[ vstup[i,1] ][ stupne[vstup[i,1]] ] := vstup[i,2];
      inc( stupne[ vstup[i,2] ] );
      znami[ vstup[i,2] ][ stupne[vstup[i,2]] ] := vstup[i,1];
   end;

   { najdeme vsetkych Morickovych znamych a zaznacime ich do pola OK }
   OK[1] := true;
   for i:=2 to D do OK[i] := false;
   for i:=1 to stupne[1] do OK[ znami[1,i] ] := true;

   { pre kazdeho znameho prejdeme vsetkych jeho znamych a zaznacime aj tych }
   for i:=1 to stupne[1] do
      for j:=1 to stupne[znami[1,i]] do
         OK[ znami[znami[1,i],j] ] := true;

   { vypiseme vsetkych, ktori nie su OK, v premennej k si pocitame, kolko ich bolo }
   k := 0;
   for i:=2 to D do
      if not OK[i] then begin
         writeln(i);
         inc(k);
      end;
   if k=0 then writeln('starosta');
end.
```

B-I-3 Poriadok na polici

Optimálne usporiadanie šestice zo zadania

Na polici ležali zväzky v poradí 4, 2, 1, 6, 5, 3. Mali sme ich usporiadať použitím najmenšieho možného počtu výmen susedných dvojíc a dokázať, že naše riešenie je najlepšie možné.

Jeden optimálny postup vyzerá nasledovne:

- Knihu 1 vymeníme s knihou 2 a následne s knihou 4. Dostávame poradie 1, 4, 2, 6, 5, 3.
- Knihu 2 vymeníme s knihou 4. Dostávame 1, 2, 4, 6, 5, 3.
- Knihu 3 vymeníme s knihou 5, knihou 6 a nakoniec aj s knihou 4. Dostávame 1, 2, 3, 4, 6, 5.
- Vymeníme knihy 5 a 6 a sme hotoví.

Dokopy sme spravili sedem výmen.

Prečo to na menej výmen nejde?

Všimnime si napríklad knihy 1 a 2. Na začiatku je kniha 1 napravo od knihy 2, na konci však už musí byť od nej naľavo. No a jediný spôsob, ako sa tam dostane, je ten, že ich niekedy počas riešenia vymeníme.

To isté platí napríklad aj pre knihy 3 a 6. V ľubovoľnom riešení musíme niekedy tieto dve knihy vymeniť, aby sme dostali knihu 3 naľavo od knihy 6.

Vyzbrojení týmto pozorovaním si už ľahko všimneme, že vo vyššie uvedenom riešení to platilo pre všetkých 7 výmen, ktoré sme spravili. Vždy sme menili dvojicu kníh takú, že kniha, ktorú sme presúvali sprava doľava, mala od tej druhej menšie číslo. Hocijaké iné riešenie musí obsahovať každú z týchto 7 výmen, preto lepšie riešenie neexistuje.

Poznámka: V odbornej literatúre sa dvojica kníh s vlastnosťou „kniha s väčším číslom je naľavo od knihy s menším číslom“ volá *inverzia*. Počet inverzii je zaujímavou mierou toho, nakoľko je postupnosť čísel neutriedená.

Usporiadanie veľa kníh

Budúci mesiac vyjde nové vydanie magickej encyklopédie. To už nebude mať 6 zväzkov, ale N. Našou úlohou je dokázať, že knihovníkovi bude určite stačiť spraviť nanajvýš $N(N-1)/2$ výmen na to, aby ich usporiadal.

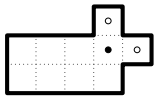
Toto je jednoduché – stačí, ak bude knihovník postupovať rovnako, ako my pri riešení prvej podúlohy. Najskôr postupnými výmenami presunie na prvú pozíciu knihu číslo 1, potom za ňu knihu číslo 2, a tak ďalej.

Nech by bola na začiatku kniha 1 hocikde, určite nám stačí nanajvýš $N-1$ výmen na to, aby sme ju dostali na začiatok. Pre knihu 2 nám už bude stačiť nanajvýš $N-2$ výmen – prvá pozícia je totiž už obsadená knihou 1. A tak ďalej. Celkový počet výmen teda vieme zhora ohraničiť hodnotou

$$(N-1)+(N-2)+\cdots+2+1=\frac{N(N-1)}{2}.$$

B-I-4 Dláždenie

Podúloha a)

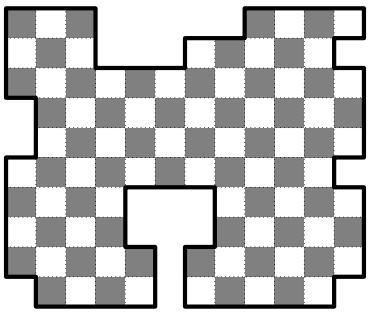


Mali sme zostrojiť miestnosť tvorenú presne 10 štvorčekmi, ktorá sa nebude dať vydláždiť. Jedna takáto miestnosť je na obrázku vpravo.

Ľahko dokážeme, že sa naozaj nedá vydláždiť. Všimnime si jedno z políčok označených prázdnym krúžkom. Jediný spôsob, ako ho pokryť dlaždicou, pokryje aj políčko označené plným krúžkom. Potom už ale nemáme ako pokryť druhé políčko označené prázdnym krúžkom.

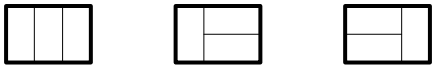
Podúloha b)

Útvar v zadaní sa vydláždiť nedá. Aby sme to dokázali, stačí si ho ofarbiť šachovnicovým vzorom. Zjavne každá dlaždica vždy pokryje jedno čierne a jedno biele políčko. Náš útvar však obsahuje 48 čiernych a 50 bielych políčok. To znamená, že nech budeme prikladať dlaždice ako len chceme, vždy nám ostanú aspoň dve biele políčka nepokryté.



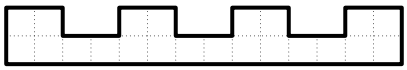
Podúloha c)

V tejto podúlohe sme mali nájsť miestnosť, ktorá bude mať presne 3 možné dláždenia. Najjednoduchšou takouto miestnosťou je obdĺžnik rozmerov 2×3. Na nasledujúcom obrázku sú všetky jeho možné dláždenia.



Podúloha d)

Existuje veľa miestností, ktoré majú 16 možných dláždení. V našom riešení začneme tým, že si všimneme, že štvorec 2×2 má dve možné dláždenia. Teraz použijeme jednoduchý trik: Ak máme dve miestnosti, z ktorých prvá má A možných dláždení a druhá B a vhodne ich spojíme do jednej väčšej miestnosti, dostaneme miestnosť s AB rôznymi dláždeniami.



Miestnosť na obrázku vznikla takýmto spojením štyroch štvorcov. Je zjavné, že má práve 16 dláždení: v každom zo štvorcov, ktoré obsahuje, máme na výber z dvoch možností, „spájacie“ dlaždice medzi štvorcami sú určené jednoznačne.

SLOVENSKÁ KOMISIA OLYMPIÁDY V INFORMATIKE

DVADSIATY ŠIESTY ROČNÍK OLYMPIÁDY V INFORMATIKE

Vydala IUVENTA s finančnou podporou Ministerstva školstva SR
Náklad: 400 výtlačkov
Zodpovedný redaktor: Michal Forišek
Sadzba programom LaTeX