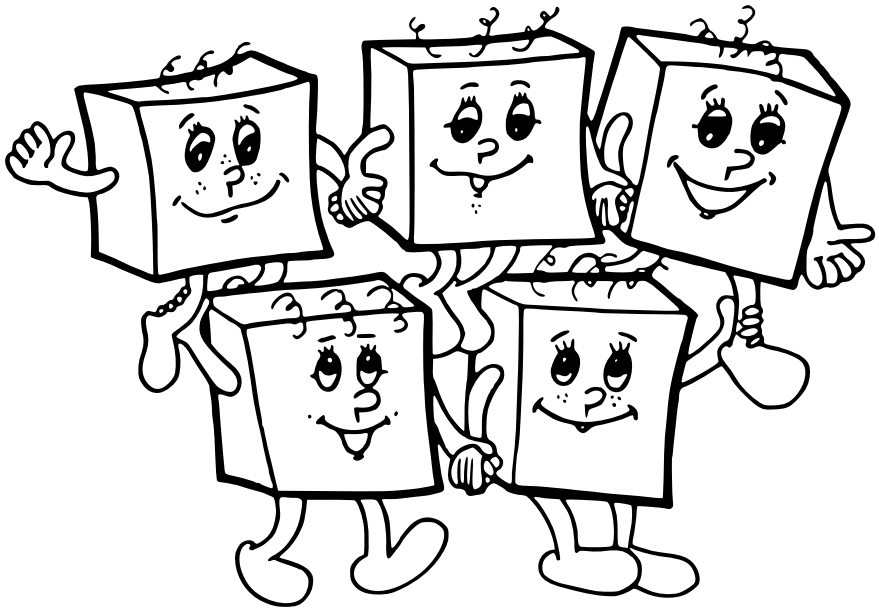
OLYMPIÁDA V INFORMATIKE
NA STREDNÝCH ŠKOLÁCH



dvadsiaty druhý (a zároveň prvý) **ročník**
školský rok 2006/07

riešenia domáceho kola
kategória B

- **Olympiáda v informatike** je od tohto školského roku samostatnou súťažou. Predchádzajúcich 21 ročníkov tejto súťaže prebiehalo pod názvom **Matematická olympiáda, kategória P** (programovanie).
- Oficiálnu **webstránku** súťaže nájdete na `http://www.ksp.sk/oi/`.
- Novinkou je zavedenie **dvoch kategórií**. Nová kategória B je určená pre mladších riešiteľov. Podrobnosti nájdete v pravidlách.

Riešenia kategórie B

Tento materiál má pomôcť učiteľom na školách pri príprave konzultácií a pracovných seminárov pre riešiteľov súťaže, členom krajských výborov MO slúži ako podklad pre opravovanie úloh domáceho kola MO kategórie P. **Študentom možno tieto komentáre poskytnúť až po termíne stanovenom pre odovzdanie riešení úloh domáceho kola MO kategórie P** ako informáciu, ako bolo treba úlohy správne riešiť a pre ich odbornú prípravu na účasť v krajskom kole súťaže.

B-I-1 Pestovanie mrkvičiek

Nech $A = [x_1, y_1]$ a $B = [x_2, y_2]$ sú body zo vstupu. Ak $x_1 = x_2$ alebo $y_1 = y_2$, úloha je ľahká.

Teraz bez ujmy na všeobecnosti predpokladajme, že $x_1 < x_2$. Nech $\delta_x = x_2 - x_1$ a $\delta_y = |y_1 - y_2|$. Nech $C = [x_3, y_3]$ je bod, ktorý leží na úsečke spájajúcej body $[x_1, y_1]$ a $[x_2, y_2]$ (rôzny od A). Označme $\delta'_x = x_3 - x_1$ a $\delta'_y = |y_1 - y_3|$. Potom z podobnosti trojuholníkov ABB' a ACC' (kde $B' = [x_2, y_1]$ a $C' = [x_3, y_1]$) plynie:

$$\frac{\delta_x}{\delta_y} = \frac{\delta'_x}{\delta'_y}$$

Nás zaujímajú také body C, pre ktoré δ'_x a δ'_y sú celé čísla. Chceme teda vedieť, koľko existuje takých celočíselných dvojíc a, b ($a \leq \delta_x$, $b \leq \delta_y$), že $\frac{a}{b} = \frac{\delta_x}{\delta_y}$. Dva zlomky sa rovnajú práve vtedy, ak sa po ich úprave na základný tvar rovnajú čitatele a menovatele. Ak d je najväčší spoločný deliteľ čísel δ_x a δ_y, tak zlomok $\frac{\delta_x}{\delta_y}$ má základný tvar $\frac{\delta_x/d}{\delta_y/d}$.

Hľadáme teda také celé čísla a, b, ktoré sa po vydelení nejakým celým číslom k rovnajú δ_x/d a δ_y/d. To sú práve čísla $(\delta_x/d, \delta_y/d)$, $(2\delta_x/d, 2\delta_y/d)$, $\ldots$, $((d-1)\delta_x/d, (d-1)\delta_y/d)$, (δ_x, δ_y), čo je dokopy d dvojíc. Spolu s bodom A (ktorý sme na začiatku úvahy vylúčili) máme $d+1$ bodov s celočíselnými súradnicami.

Číslo d vypočítame štandardným Euklidovým algoritmom. Všimnite si, že v našej implementácií, ak jeden zo vstupných parametrov pre funkciu `nsd` je 0, tak výsledkom tejto funkcie je to druhé číslo, teda prípad $x_1 = x_2$, resp. $y_1 = y_2$ nemusíme uvažovať ako špeciálny.

Listing programu:

```
var x1,y1:longint;
    x2,y2:longint;

function nsd(a,b:longint):longint;
var c:longint;
begin
  while (b<>0) do begin
    c:=a mod b; a:=b; b:=c;
  end;
  nsd:=a;
end;

begin
  readln(x1,y1);
  readln(x2,y2);
  writeln(nsd(abs(x2-x1),abs(y2-y1))+1);
end.
```

B-I-2 Hladný Samo

Táto úloha patri medzi tzv. ťažké úlohy v tom zmysle, že pre ňu ešte nikto nevymyslel efektívny algoritmus. Náše riešenie teda bude len skúšať všetky možnosti.

Ide teda o to, ako vygenerovať všetky možné podmnožiny potravín, pre jednu podmnožinu overiť, či v nej nie sú 2 také potraviny, ktoré sa nemôžu jesť spolu a vybrať nejakú s najväčším počtom prvkov. Generovať všetky možné podmnožiny sa dá rôzne. Pomerne jednoduché riešenie je postupne prechádzať čísla 0, 1, ..., $2^N - 1$, kde N je počet potravín, a reprezentovať ich binárny zápis ako podmnožinu (i-ty bit hovorí, či tam i-ta potravina je alebo nie).

Prehľadnejšie a efektívnejšie je generovať podmnožiny rekurzívne. Pre každú potravinu sa rozhodneme, či ju zjeme alebo nie a pre obe možnosti sa rekurzívne spustíme na nasledujúcu potravinu. Výhoda je tá, že môžeme hneď overiť, či danú potravinu môžeme v danom momente zjesť (zisťujeme, či potravina, ktorú

chceme zjesť sa môže jesť spolu s doteraz zjedenými) a teda nemusíme to kontrolovať, až keď máme vygenerovanú celú podmnožinu.

Na konci už len zistíme, či sme našli väčšiu podmnožinu ako doteraz nájdenú. Časová zložitosť tohto algoritmu je $O(N2^N)$.

Listing programu:

```
const MAXN=25;

var N,M,a,b,i,j:integer;
    mnoz:array[1..MAXN] of integer;
    maxmnoz:array[1..MAXN] of integer;
    dvojice:array [1..MAXN,1..MAXN] of boolean;
    poc,max:integer;

procedure skus(k:integer);
var i:integer;
    ok:boolean;
begin
  if (k=N+1) then begin
    if (poc>max) then begin
      max:=poc;
      for i:=1 to poc do maxmnoz[i]:=mnoz[i];
    end;
  end else begin
    skus(k+1);
    ok:=true;
    for i:=1 to poc do
      if dvojice[k,mnoz[i]] then
        ok:=false;
    if ok then begin
      inc(poc);
      mnoz[poc]:=k;
      skus(k+1);
      dec(poc);
    end;
  end;
end;

begin
  readln(N,M);
  for i:=1 to N do for j:=1 to N do dvojice[i,j]:=false;

  for i:=1 to M do begin
    readln(a,b);
    dvojice[a,b]:=true; dvojice[b,a]:=true;
  end;

  skus(1);
  writeln(max);
  for i:=1 to max do write(maxmnoz[i],' ');
  writeln;
end.
```

B-I-3 Družstvá

Vezmime prvého hráča a bez ujmy na všeobecnosti ho priraďme do 1. družstva. Všetci hráči, ktorí nemôžu s ním hrať v družstve musia automaticky hrať v 2. družstve. Všetci, ktorí nemôžu hrať s týmito, musia hrať v 1. družstve, atď.

Teraz, buď sa nám podarí hráčov rozdeliť (ostane nám ešte rozdeliť hráčov, ktorí sú nezávislí od týchto), alebo budeme chcieť nejakého hráča dať do oboch družstiev. V takom prípade rozdelenie hráčov neexistuje. Funguje to len kvôli tomu, že máme len 2 družstvá. Ak by sme chceli zistiť, či sa dajú hráči rozdeliť do 3 družstiev, už to nie je také jednoduché.

Rýchlosť algoritmu závisí od zvolenej reprezentácie vzťahov medzi hráčmi a od použitého systému priradzovania. Vzťahy si môžeme zapisovať do booleovského poľa $N \times N$, kde $A[i, j]$ je `true` práve vtedy keď i a j nemôžu hrať spolu. Pri takejto reprezentácií je časová a pamäťová zložitosť $O(N^2)$.

Listing programu:

```
const MAXN=100;

var N,M,a,b,i,j:integer;
    dvojice:array [1..MAXN,1..MAXN] of boolean;
    druzstvo:array [1..MAXN] of integer;
    nedasa:boolean;

procedure rozdel(v:integer);
var i:integer;
begin
  for i:=1 to N do
    if dvojice[v][i] then begin
      if druzstvo[i]=0 then begin
        druzstvo[i]:=3-druzstvo[v];
        rozdel(i);
      end else if druzstvo[i]=druzstvo[v] then
        nedasa:=true;
    end;
end;

begin
  readln(N,M);
  for i:=1 to N do for j:=1 to N do dvojice[i,j]:=false;
  for i:=1 to N do druzstvo[i]:=0;

  for i:=1 to M do begin
```

```
    readln(a,b);
    dvojice[a,b]:=true; dvojice[b,a]:=true;
  end;

  nedasa:=false;
  for i:=1 to N do
    if druzstvo[i]=0 then begin
      druzstvo[i]:=1;
      rozdel(i);
    end;

  if nedasa then
    writeln('Nie')
  else begin
    writeln('Ano');
    for i:=1 to N do
      if druzstvo[i]=1 then write(i,' ');
    writeln;
  end;
end.
```

B-I-4 Assembler

a) Vstup načítame na dvakrát. Pri prvom prechode si riadky očíslujeme a čísla riadkov, na ktorých sa vyskytujú návestia si zapamätáme. Pri druhom prechode programom si zapisujeme inštrukcie, pričom návestia nahradzujeme číslami riadkov, kde sa vyskytuje ich definícia.

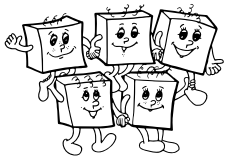
   Teraz už len odsimulujeme program na danom vstupe. Budeme si pamätať tzv. Instruction Pointer (IP), čo je číslo inštrukcie (v našom prípade riadku), kde bude vykonávanie programu pokračovať. V prípade jednoduchej inštrukcie len príslušne upravíme registre (a príznak **Z**). V prípade inštrukcií **jmp**, **jz** a **jnz** len upravíme IP na príslušné číslo riadku.

b) Postupne budeme od registra R_1 odčítavať hodnotu R_2. To budeme robiť tak, že vždy od oboch odčítame 1 a ku R_0 pričítame 1. Ak sa nám podarí odčítať celé R_2, tak v R_0 máme pôvodné R_2, takže to len prelejeme do R_2 a pokračujeme znovu. Ak sa nám to nepodarí (tzn. ak budeme chcieť od R_1 odčítať 1, ale už nebude z čoho), tak v R_0 je zapísané, koľko sa nám podarilo odčítať, čo je akurát zvyšok po delení.

Na zisťovanie, či je už v R_2 nula sme využili to, že inštrukcia **dec** tiež nastavuje príznak **Z**, ak po jej vykonaní je $R_2 = 0$.

```
c1: test R1,R1
    jz koniec
    dec R1
    inc R0
    dec R2
    jz c2
    jmp c1
c2: test R0,R0
    jz c1
    dec R0
    inc R2
    jmp c2
koniec:
```

Olympiáda v Informatike



2006/07
domáce kolo
riešenia kategórie B

SLOVENSKÁ KOMISIA OLYMPIÁDY V INFORMATIKE

DVADSIATY DRUHÝ ROČNÍK OLYMPIÁDY V INFORMATIKE

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