Metroul

Metroul din Londra reprezintă un sistem complex, care transportă zilnic milioane de pasageri (*fig. 1*). Din punctul de vedere al pasagerului, metroul poate fi tratat ca o mulțime de linii de tren şi o mulțime de staţii. În scopuri didactice, vom nota liniile de tren prin literele mari *A*, *B*, *C*, *D* ş.a.m.d. ale alfabetului latin, în total *n* linii, iar staţiile – prin numerele naturale 1, 2, 3, ..., în total *m* staţii (*fig. 2*).

Fig. 1

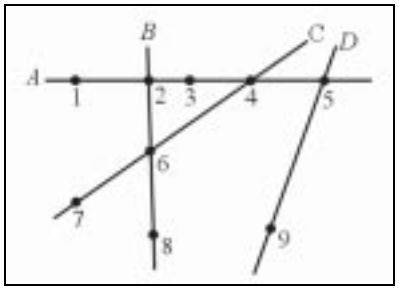

Fig. 2

Liniile de tren şi staţiile respective au fost proiectate în aşa fel, încît pasagerul, care pleacă din orice staţie *x*, să poată ajunge în oricare altă staţie *y*. Evident, în cazurile în care staţiile se află pe linii diferite, pasagerul este nevoit să facă una sau mai multe transbordări, schimbînd trenul în staţiile în care se întîlnesc două sau mai multe linii de tren.

De exemplu, pentru a ajunge din staţia 1 în staţia 8 (*fig. 2*), pasagerul poate să facă o singură transbordare în staţia 2 sau două transbordări – prima în staţia 4 şi a doua în staţia 6.

Elaboraţi un program, care, cunoscînd planul metroului, staţia de plecare *x* şi staţia de sosire *y*, calculează numărul minim de transbordări.

Date de intrare.

Fişierul text `METROU.IN` conţine pe prima linie numerele naturale *n*, *m*, *x*, *y* separate prin spaţiu. Fiecare din următoarele *n* linii ale fişierului conţine numere de staţii separate prin spaţiu. Linia a 2-a a fişierului de intrare conţine numerele de staţii ale liniei de tren *A*, linia a treia a fişierului de intrare conţine numerele de staţii ale liniei de tren *B* ş.a.m.d.

Date de ieşire.

Fişierul text `METROU.OUT` va conţine pe o singură linie numărul minim de transbordări.

Exemplu.

`METROU.IN`

```
4 9 1 8
1 2 3 4 5
2 6 8
7 6 4
5 9
```

`METROU.OUT`

```
1
```

Restricţii. $2 \le n \le 26$, $3 \le m \le 250$, $x \ne y$. Timpul de execuţie nu va depăşi 5 secunde. Fişierul sursă va avea denumirea `METROU.PAS`, `METROU.C` sau `METROU.CPP`.

Rezolvare

Introducem în studiu mulțimile L_1, L_2, ..., L_n, fiecare mulțime reprezentînd cîte o linie de tren. De exemplu, în cazul *fig. 2*, avem:

$L_1 = \{1, 2, 3, 4, 5\}$ – linia *A*;
$L_2 = \{ 2, 6, 8\}$ – linia *B*;
$L_3 = \{7, 6, 4\}$ – linia *C*;
$L_4 = \{ 5, 9\}$ – linia *D*.

De asemenea, vom utiliza următoarele notații:

S – mulțimea stațiilor de metrou;

S_k – mulțimea stațiilor de metrou la care se ajunge, pornind din stația *x*, prin exact *k* transbordări.

Mulțimile S_0, S_1, S_2, ..., S_k pot fi calculate iterativ, după cum urmează:

1) în mulțimea S_0 se includ toate stațiile de pe liniile pe care apare stația *x*;

2) în mulțimea S_1 se includ toate stațiile de pe liniile pe care apare cel puțin o stație din mulțimea S_0, dar care nu au fost deja incluse în S_0;

3) în general, în mulțimea S_k se includ toate stațiile de pe liniile pe care apare cel puțin o stație din mulțimea S_{k-1}, dar care nu au fost deja incluse în mulțimile S_0, S_1, S_2, ..., S_{k-1}.

Întrucît numărul minim de transbordări nu poate depăși valoarea *n*-1, calculele recurente ale mulțimilor S_0, S_1, S_2, ..., S_k se vor termina cînd $k = n - 1$.

Pe calculator mulțimile S_0, S_1, S_2, ..., S_k pot fi reprezentate cu ajutorul unui singur vector $T = \|t_j\|_m$, denumit vectorul transbordărilor. Componentele t_j ale acestui vector pot fi calculate iterativ după cum urmează:

- inițial pentru toate componentele stabilim $t_j = -1$, fapt ce semnifică $S_0 = S_1 = S_2 = ... = S_k = \varnothing$;
- în continuare, pentru toate stațiile *j* din mulțimea S_k (*k* va lua consecutiv valorile 0, 1, 2 ş.a.m.d.) stabilim $t_j = k$.

În programul ce urmează liniile de tren şi vectorul transbordărilor sînt reprezentate prin structurile de date

```
const nmax=26;
      mmax=250;
type Linie = set of 1..mmax;
var L : array [1..26] of Linie;
    n : 2..nmax;
    m, x, y  : 3..mmax;
    T : array[1..mmax] of integer;
```

iar apartenența stațiilor la anumite linii de metrou se verifică cu ajutorul operatorului **in**.

```
Program Metro;
 { Clasele 10-12 }
const nmax=26;
      mmax=250;
```

```
type Linie = set of 1..mmax;
var L : array [1..26] of Linie;
    n : 2..nmax;
    m, x, y  : 3..mmax;
    T : array[1..mmax] of integer;

procedure Citeste;
 { Citirea datelor de intrare }
var i, j : integer;
    Intrare : text;
begin
  assign(Intrare, 'METRO.IN');
  reset(Intrare);
  readln(Intrare, n, m, x, y);
  for i:=1 to n do
    begin
      L[i]:=[];
      while not eoln(Intrare) do
        begin
         read(Intrare, j);
         L[i]:=L[i]+[j];
        end; { while }
        readln(Intrare);
    end; { for }
  close(Intrare);
end; { Citeste }

procedure Scrie;
 { Scrierea datelor in fisierul de iesire }
var i, j, k : integer;
    Iesire : text;
begin
  assign(Iesire, 'METRO.OUT');
  rewrite(Iesire);
  writeln(Iesire, T[y]);
  close(Iesire);
end; { Scrie }

procedure Transbordari;
 { Calculeaza vectorul T }
var i, j, k, q : integer;
begin
  { initializam vectorul T }
  for j:=1 to m do T[j]:=-1;
  { calculam multimea S0 }
  for i:=1 to n do
    if (x in L[i]) then
      for j:=1 to m do
        if (j in L[i]) then T[j]:=0;
  { calculam multimile Sk }
```

```
  for k:=1 to n-1 do
    for i:=1 to n do
      for j:=1 to m do
        if ((j in L[i]) and (T[j]=k-1)) then
          for q:=1 to m do
            if ((q in L[i]) and (T[q]=-1)) then T[q]:=k;
end; { Transbordari }

begin
  Citeste;
  Transbordari;
  Scrie;
end.
```

Din analiza procedurii `Transbordari` se observă că complexitatea temporală a algoritmului respectiv este de ordinul $O(n^2m^2)$. Conform restricţiilor problemei, $n \leq 26$ şi $m \leq 250$. Prin urmare, în cazul cel mai nefavorabil, numărul necesar de operaţii este de ordinul 10^8, mărime comparabilă cu capacitatea de prelucrare a calculatoarelor personale.