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

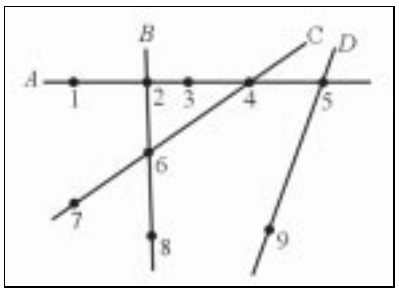

Fig. 2

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.

În caz de reparaţii, unele staţii de metrou pot fi temporar închise pentru transbordarea pasagerilor. În astfel de cazuri, nu întotdeauna pasagerul care pleacă din staţia *x*, poate să ajunge în oricare altă staţie *y*. De exemplu, dacă staţia 5 va fi închisă pentru transbordări (*fig. 2*), pasagerii care pleacă din staţia 1 nu mai pot ajunge în staţia 9. Un alt exemplu poate servi închiderea staţiei 2, care însă, nu împiedică pasagerii ce pleacă din staţia 1 să ajungă în oricare altă staţie de metrou.

Elaboraţi un program, care, cunoscînd planul metroului, staţia de plecare *x*, staţia de sosire *y* şi staţia închisă *z*, determină dacă pasagerul poate ajunge din *x* în *y*.

Date de intrare.

Fişierul text `METROU.IN` conţine pe prima linie numerele naturale *n*, *m*, *x*, *y*, *z* 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 cuvîntul `DA` dacă pasagerul poate ajunge din staţia *x* în staţia *y* şi `NU` în caz contrar.

Exemplu.

`METROU.IN`

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

`METROU.OUT`

```
DA
```

Restricţii. $2 \le n \le 26$, $3 \le m \le 250$, $x \ne y$, $x \ne z$, $y \ne z$. 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*.

Fie S mulțimea stațiilor de metrou la care se poate ajunge pornind din stația x. Această mulțime poate fi calculată iterativ după cum urmează:

1) Inițial includem în mulțimea S_0 toate stațiile de pe liniile pe care apare stația x.

2) Presupunem că la un anumit pas k avem deja calculată mulțimea S_{k-1}. Mulțimea S_k se obține adăugînd la mulțimea S_{k-1} toate stațiile de pe liniile pe care apare cel puțin o stație din mulțimea S_{k-1}. Evident, $S_0 \subseteq S_1 \subseteq S_2 \subseteq ... \subseteq S_k$, iar în mulțimile S_0, S_1, S_2, ..., S_k nu trebuie să apară stația z, care este închisă pentru transbordări.

3) Indiferent de configurația liniilor de metrou, numărul necesar de transbordări nu poate depăşi valoarea n-1. Prin urmare, mulțimile S_k trebuie calculate pentru k = 1, 2, 3 ş.a.m.d, în cel mai nefavorabil caz, pînă la $k = n$-1.

Pe calculator mulțimile S_0, S_1, S_2, ..., S_k pot fi reprezentate cu ajutorul unui singur vector (tablou) $T = \|t_j\|_m$, denumit vectorul stațiilor accesibile. La fiecare pas k, componentele t_j ale acestui vector au următoarea semnificație:

$$t_{ij} = \begin{cases} 1, & \text{daca statia } j \text{ apartine multimii } S_k; \\ 0, & \text{in caz contrar.} \end{cases}$$

În programul ce urmează liniile de tren şi vectorul stațiilor accesibile 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, z  : 3..mmax;
    T : array[1..mmax] of boolean;
```

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

```
Program Metro;
 { Clasele 9-10 }
const nmax=26;
      mmax=250;
type Linie = set of 1..mmax;
var L : array [1..26] of Linie;
```

```
    n : 2..nmax;
    m, x, y, z : 3..mmax;
    T : array[1..mmax] of boolean;

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, z);
  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);
  if T[y] then writeln(Iesire, 'DA')
          else Writeln(Iesire, 'NU');
  close(Iesire);
end; { Scrie }

procedure Accesibilitate;
 { Calculeaza vectorul T }
var i, j, k, q : integer;
begin
  { excludem statia z din toate liniile de metrou }
  for i:=1 to n do
    if (z in L[i]) then L[i]:=L[i]-[z];
  { initializam vectorul T }
  for j:=1 to m do T[j]:=false;
  { 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]:=true;
  { 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]) then
          for q:=1 to m do
            if (q in L[i]) then T[q]:=true;
end; { Accesibilitate }

begin
  Citeste;
  Accesibilitate;
  Scrie;
end.
```

Din analiza procedurii `Accesibilitate` se observă că instrucțiunea

```
if (q in L[i]) then T[q]:=true
```

din interiorul celor patru cicluri imbricate se va efectua de cel mult $n(n\text{-}1)m^2$ ori. Conform restricțiilor problemei, $n \leq 26$ şi $m \leq 250$. Prin urmare, în cazul cel mai nefavorabil, numărul necesar de operații este de ordinul 10^8, mărime comparabilă cu capacitatea de prelucrare a calculatoarelor personale.