Keys

Arhitectul Timothy a inventat un nou joc de evadare. În acest joc sunt n camere numerotate de la 0 la $n-1$. Inițial, fiecare cameră conține exact o cheie. Fiecare cheie are asociat un tip care este un număr întreg între 0 și $n-1$, inclusiv. Tipul cheii din camera i ($0 \leq i \leq n-1$) este $r[i]$. Atenție: este posibil ca mai multe camere să conțină chei de același tip, adică, valorile $r[i]$ nu sunt neapărat distincte.

De asemenea, în acest joc sunt și m conectori **bidirecționali**, numerotați de la 0 la $m-1$. Conectorul j ($0 \leq j \leq m-1$) conectează o pereche de camere distincte $u[j]$ și $v[j]$. Atenție: o pereche de camere pot fi conectate prin mai mulți conectori.

Jocul este jucat de o singură persoană care colectează chei și se mută între camere prin traversarea conectorilor. Spunem că jucătorul **traversează** conectorul j când acesta folosește conectorul pentru a se muta din camera $u[j]$ în camera $v[j]$, sau viceversa. Jucătorul poate traversa conectorul j numai dacă acesta a colectat deja o cheie de tipul $c[j]$.

În orice moment de timp, jucătorul este într-o cameră x și poate efectua două tipuri de acțiuni:

- colectează cheia din camera x, al cărui tip este $r[x]$ (numai dacă acesta nu a colectat-o deja),
- traversează un conector j, unde fie $u[j] = x$ sau $v[j] = x$, dacă jucătorul a colectat deja o cheie de tipul $c[j]$. Atenție: jucătorul nu pierde **niciodată** o cheie pe care a colectat-o deja.

Jucătorul **începe** jocul în camera s fără a avea vreo cheie în posesie. O cameră t este **accesibilă** din camera s dacă jucătorul care începe jocul în camera s poate efectua o secvență de acțiuni descrise mai sus ca să ajungă în camera t.

Pentru fiecare cameră i ($0 \leq i \leq n-1$), notăm numărul de camere accesibile din camera i cu $p[i]$. Timothy ar vrea să afle mulțimea indicilor i pentru care valoarea $p[i]$ este egală cu minimul tuturor valorilor $p[i]$ pentru oricare $0 \leq i \leq n-1$.

Detalii de Implementare

Trebuie să implementați următoarea procedură:

```
int[] find_reachable(int[] r, int[] u, int[] v, int[] c)
```

- r: un tablou unidimensional de lungime n. Pentru fiecare i ($0 \leq i \leq n-1$), cheia din camera i este de tipul $r[i]$.
- u, v: două tablouri unidimensionale de lungime m fiecare. Pentru fiecare j ($0 \leq j \leq m-1$), conectorul j conectează camerele $u[j]$ și $v[j]$.

- c: un tablou unidimensional de lungime m. Pentru fiecare j ($0 \leq j \leq m - 1$), tipul cheii necesare pentru a traversa conectorul j este $c[j]$.
- Această procedură trebuie să returneze un tablou unidimensional a de lungime n. Pentru fiecare $0 \leq i \leq n - 1$, valoarea lui $a[i]$ este 1 dacă pentru fiecare j astfel încât $0 \leq j \leq n - 1$, avem $p[i] \leq p[j]$. Altfel, valoarea lui $a[i]$ este 0.

Exemple

Exemplul 1

Să considerăm următorul apel:

```
find_reachable([0, 1, 1, 2],
               [0, 0, 1, 1, 3], [1, 2, 2, 3, 1], [0, 0, 1, 0, 2])
```

Dacă jucătorul începe jocul în camera 0, se poate aplica următorul șir de acțiuni:

Camera curentă	Acțiunea
0	Culege cheia de tip 0
0	Traversează conectorul 0 către camera 1
1	Culege cheia de tip 1
1	Traversează conectorul 2 către camera 2
2	Traversează conectorul 2 către camera 1
1	Traversează conectorul 3 către camera 3

Deci se poate ajunge în camera 3 plecând din camera 0. Similar, putem construi șiruri de acțiuni care arată că se poate ajunge la orice cameră plecând din camera 0 deci $p[0] = 4$. Tabelul de mai jos arată, pentru fiecare cameră de plecare, în ce camere se poate ajunge.

Camera de plecare i	Camerele în care se poate ajunge	$p[i]$
0	$[0, 1, 2, 3]$	4
1	$[1, 2]$	2
2	$[1, 2]$	2
3	$[1, 2, 3]$	3

Cea mai mică valoare pentru $p[i]$ pentru toate camerele este 2, și se obține pentru $i = 1$ sau $i = 2$. De aceea, acest apel ar trebui să returneze $[0, 1, 1, 0]$.

Exemplul 2

```
find_reachable([0, 1, 1, 2, 2, 1, 2],
               [0, 0, 1, 1, 2, 3, 3, 4, 4, 5],
               [1, 2, 2, 3, 3, 4, 5, 5, 6, 6],
               [0, 0, 1, 0, 0, 1, 2, 0, 2, 1])
```

Tabelul de mai jos descrie camerele în care se poate ajunge:

Camera de plecare i	Camerele în care se poate ajunge	$p[i]$
0	$[0, 1, 2, 3, 4, 5, 6]$	7
1	$[1, 2]$	2
2	$[1, 2]$	2
3	$[3, 4, 5, 6]$	4
4	$[4, 6]$	2
5	$[3, 4, 5, 6]$	4
6	$[4, 6]$	2

Cea mai mică valoare pentru $p[i]$ pentru toate camerele este 2, și se obține pentru $i \in \{1, 2, 4, 6\}$. De aceea, acest apel ar trebui să returneze $[0, 1, 1, 0, 1, 0, 1]$.

Example 3

```
find_reachable([0, 0, 0], [0], [1], [0])
```

Tabelul de mai jos descrie camerele în care se poate ajunge:

Camera de plecare i	Camerele în care se poate ajunge	$p[i]$
0	$[0, 1]$	2
1	$[0, 1]$	2
2	$[2]$	1

Cea mai mică valoare pentru $p[i]$ pentru toate camerele este 1, și se obține pentru $i = 2$. De aceea, acest apel ar trebui să returneze $[0, 0, 1]$.

Restricții

- $2 \leq n \leq 300\,000$
- $1 \leq m \leq 300\,000$
- $0 \leq r[i] \leq n - 1$ pentru oricare $0 \leq i \leq n - 1$
- $0 \leq u[j], v[j] \leq n - 1$ și $u[j] \neq v[j]$ pentru oricare $0 \leq j \leq m - 1$

- $0 \leq c[j] \leq n - 1$ pentru oricare $0 \leq j \leq m - 1$

Subtasks-uri

1. (9 puncte) $c[j] = 0$ pentru oricare $0 \leq j \leq m - 1$ și $n, m \leq 200$
2. (11 puncte) $n, m \leq 200$
3. (17 puncte) $n, m \leq 2000$
4. (30 puncte) $c[j] \leq 29$ (pentru oricare $0 \leq j \leq m - 1$) și $r[i] \leq 29$ (pentru oricare $0 \leq i \leq n - 1$)
5. (33 puncte) Fără restricții suplimentare.

Exemplul de Grader

Grader-ul citește datele de intrare în următorul format:

- linia 1: n m
- linia 2: $r[0]$ $r[1]$ $\ldots$ $r[n-1]$
- linia $3 + j$ ($0 \leq j \leq m - 1$): $u[j]$ $v[j]$ $c[j]$

Grader-ul afișează valoare returnată de `find_reachable` în următorul format:

- linia 1: $a[0]$ $a[1]$ $\ldots$ $a[n-1]$