Escape room

L'architetto Timothy ha costruito una nuova *escape room* composta da n stanze numerate da 0 a $n-1$. All'interno della stanza i è presente una chiave di tipo $r[i]$ (un intero tra 0 e $n-1$, inclusi). Nota che più stanze possono contenere chiavi dello stesso tipo, quindi i valori di $r[i]$ non sono necessariamente distinti.

Sono inoltre presenti m passaggi **bidirezionali**, numerati da 0 a $m-1$. Il passaggio j collega una coppia di stanze distinte $u[j]$ e $v[j]$, ed è bloccato da una serratura di tipo $c[j]$. Una stessa coppia di stanze può essere collegata da più passaggi.

Ti muovi tra le stanze, raccogliendo le chiavi e **attraversando** i passaggi, cioè usando un passaggio j per andare dalla stanza $u[j]$ alla stanza $v[j]$, o viceversa. Per farlo devi aver già raccolto una chiave di tipo $c[j]$.

Durante il gioco, se ti trovi nella stanza x puoi fare due azioni:

- raccogliere la chiave di tipo $r[x]$ lì presente, se non l'hai già raccolta,
- attraversare un passaggio j, se $u[j] = x$ oppure $v[j] = x$, e hai già raccolto una chiave di tipo $c[j]$.

Nota che **non scarti mai** le chiavi raccolte.

Inizi senza chiavi da una qualche stanza s. Una stanza t è **raggiungibile** da s se, facendo una qualche sequenza di azioni a partire da s, puoi raggiungere t.

Per ogni stanza i, sia $p[i]$ il numero di stanze raggiungibili partendo da i. Trova l'insieme delle stanze che raggiungono il minor numero di altre stanze, cioè gli indici i per cui $p[i]$ è minimo.

Note di implementazione

Devi implementare la seguente funzione:

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

- r: array di lunghezza n, dove $r[i]$ è il tipo della chiave nella stanza i ($0 \le i \le n-1$).
- u, v: array di lunghezza m, dove il passaggio j ($0 \le j \le m-1$) collega le stanze $u[j]$ e $v[j]$.
- c: array di lunghezza m, dove il passaggio j ($0 \le j \le m-1$) richiede una chiave di tipo $c[j]$.
- Questa funzione deve restituire un array a di n elementi, per cui $a[i]$ ($0 \le i \le n-1$) deve essere 1 se $p[i]$ è minimo ($p[i] \le p[j]$ per ogni $0 \le j \le n-1$); altrimenti deve essere 0.

Esempi

Esempio 1

Considera la seguente chiamata:

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

Se il giocatore inizia dalla stanza 0, può fare le seguente sequenza di azioni:

Stanza attuale	Azione
0	Raccogliere la chiave 0
0	Attraversare il passaggio 0 fino alla stanza 1
1	Raccogliere la chiave 1
1	Attraversare il passaggio 2 fino alla stanza 2
2	Attraversare il passaggio 2 fino alla stanza 1
1	Attraversare il passaggio 3 fino alla stanza 3

Quindi la stanza 3 è raggiungibile dalla stanza 0. Dalla stanza 0 è possibile raggiungere ogni altra stanza, quindi $p[0] = 4$. Questa tabella mostra quali sono le stanze raggiungibili:

Stanza iniziale i	Stanze raggiungibili	$p[i]$
0	$[0, 1, 2, 3]$	4
1	$[1, 2]$	2
2	$[1, 2]$	2
3	$[1, 2, 3]$	3

Il valore minimo di $p[i]$ è 2, e questo corrisponde agli indici $i = 1$ e $i = 2$. Quindi la funzione deve restituire $[0, 1, 1, 0]$.

Esempio 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])
```

Questa tabella mostra le stanze raggiungibili:

Stanza iniziale i	Stanze raggiungibili	$p[i]$
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

Il valore minimo di $p[i]$ è 2, e questo corrisponde alle stanze $i \in \{1, 2, 4, 6\}$. Quindi la funzione deve restituire $[0, 1, 1, 0, 1, 0, 1]$.

Esempio 3

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

Questa tabella mostra le stanze raggiungibili:

Stanza iniziale i	Stanze raggiungibili	$p[i]$
0	$[0, 1]$	2
1	$[0, 1]$	2
2	$[2]$	1

Il valore minimo di $p[i]$ è 1 corrispondente ad $i = 2$, quindi devi restituire $[0, 0, 1]$.

Assunzioni

- $2 \leq n \leq 300\,000$.
- $1 \leq m \leq 300\,000$.
- $0 \leq r[i] \leq n - 1$ per ogni $0 \leq i \leq n - 1$.
- $0 \leq u[j], v[j] \leq n - 1$ e $u[j] \neq v[j]$ per ogni $0 \leq j \leq m - 1$.
- $0 \leq c[j] \leq n - 1$ per ogni $0 \leq j \leq m - 1$.

Subtask

1. (9 points) $c[j] = 0$ per ogni $0 \leq j \leq m - 1$ e $n, m \leq 200$.
2. (11 points) $n, m \leq 200$.
3. (17 points) $n, m \leq 2000$.
4. (30 points) $c[j] \leq 29$ (per ogni $0 \leq j \leq m - 1$) e $r[i] \leq 29$ (per ogni $0 \leq i \leq n - 1$).
5. (33 points) Nessuna limitazione aggiuntiva.

Grader di esempio

Il grader di esempio legge l'input nel seguente formato:

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

Il grader di esempio stampa l'output nel seguente formato:

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