Clefs (Keys)

Timothy a concu un nouvel escape game. Dans ce jeu, il y a n salles numérotées de 0 à $n-1$. Initialement, chaque salle contient exactement une clef. Chaque clef a un type, qui est un entier compris entre 0 et $n-1$ inclus. Le type de la clef dans la salle i ($0 \leq i \leq n-1$) est $r[i]$. Notez que plusieurs salles peuvent contenir des clefs du même type, c'est à dire que les valeurs $r[i]$ ne sont pas nécessairement toutes distinctes.

Il y a aussi m connecteurs **bidirectionnels** dans le jeu, numérotés de 0 à $m-1$. Le connecteur j ($0 \leq j \leq m-1$) connecte une paire de salles distinctes $u[j]$ et $v[j]$. Une paire de salles peut être connectée par plusieurs connecteurs.

Le jeu est joué par un unique joueur qui ramasse les clefs et se déplace de salle en salle en empruntant les connecteurs. On dit que le joueur **emprunte** le connecteur j lorsqu'il utilise ce connecteur pour se déplacer de la salle $u[j]$ à la salle $v[j]$, ou vice versa. Le joueur ne peut emprunter le connecteur j que s'il a ramassé une clef de type $c[j]$ auparavant.

À tout instant du jeu, le joueur est dans une certaine salle x et peut effectuer deux types d'actions :

- ramasser une clef dans la salle x, dont le type est $r[x]$ (sauf s'il a déjà cette clef),
- emprunter un connecteur j, tel que $u[j] = x$ ou $v[j] = x$, si le joueur a déjà collecté une clef de type $c[j]$ auparavant. Notez que le joueur ne jette **jamais** une clef qu'il a ramassée.

Le joueur **commence** le jeu dans une salle s sans aucune clef. Une salle t est **accessible** depuis une salle s, si le joueur commençant le jeu dans la salle s peut effectuer une série d'actions décrit ci-dessus et atteindre la salle t.

Pour chaque salle i ($0 \leq i \leq n-1$), on note $p[i]$ le nombre de salles accessibles depuis la salle i. Timothy aimerait connaître l'ensemble des indices i qui atteigne la valeur minimale de $p[i]$ pour $0 \leq i \leq n-1$.

Détails d'implémentation

Vous devez implémenter la fonction suivante :

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

- r : un tableau de taille n. Pour chaque i ($0 \leq i \leq n-1$), la clef de la salle i est de type $r[i]$.
- u, v : deux tableaux de taille m. Pour chaque j ($0 \leq j \leq m-1$), le connecteur j relie les salles $u[j]$ et $v[j]$.

- c : un tableau de taille m. Pour chaque j ($0 \leq j \leq m-1$), le type de clef requis pour emprunter le connecteur j est $c[j]$.
- Cette fonction doit renvoyer un tableau a de taille n. Pour chaque $0 \leq i \leq n-1$, la valeur de $a[i]$ doit être 1 si pour tout j tel que $0 \leq j \leq n-1$, $p[i] \leq p[j]$. Sinon, la valeur de $a[i]$ doit être 0.

Exemples

Exemple 1

Considérez l'appel suivant :

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

Si le joueur commence dans la salle 0, il peut effectuer les actions suivantes :

Salle actuelle	Action
0	Ramasser une clef de type 0
0	Emprunter le connecteur 0 vers la salle 1
1	Ramasser une clef de type 1
1	Emprunter le connecteur 2 vers la salle 2
2	Emprunter le connecteur 2 vers la salle 1
1	Emprunter le connecteur 3 vers la salle 3

Par conséquent, la salle 3 est accessible depuis la salle 0. De manière similaire, il est possible de construire des séquences montrant que toutes les salles sont accessibles depuis la salle 0 et donc $p[0] = 4$. La table ci-dessous donne le nombre de salles accessibles pour chaque salle de départ :

Salle de départ i	Salles accessibles	$p[i]$
0	$[0, 1, 2, 3]$	4
1	$[1, 2]$	2
2	$[1, 2]$	2
3	$[1, 2, 3]$	3

La plus petite valeur de $p[i]$ est 2, et est atteinte pour $i = 1$ or $i = 2$. Par conséquent, la fonction doit renvoyer $[0, 1, 1, 0]$.

Exemple 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])
```

La table ci-dessous donne les salles accessibles :

Salle de départ i	Salles accessibles	$p[i]$
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

La plus petite valeur de $p[i]$ est 2, et est atteinte pour $i \in \{1, 2, 4, 6\}$. Par conséquent, la fonction doit renvoyer $[0, 1, 1, 0, 1, 0, 1]$.

Exemple 3

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

La table ci-dessous donne les salles accessibles :

Salle de départ i	Salles accessibles	$p[i]$
0	$[0, 1]$	2
1	$[0, 1]$	2
2	$[2]$	1

La plus petite valeur de $p[i]$ est 1, et est atteinte pour $i = 2$. Par conséquent, la fonction doit renvoyer $[0, 0, 1]$.

Contraintes

- $2 \leq n \leq 300\,000$
- $1 \leq m \leq 300\,000$
- $0 \leq r[i] \leq n - 1$ pour tout $0 \leq i \leq n - 1$
- $0 \leq u[j], v[j] \leq n - 1$ et $u[j] \neq v[j]$ pour tout $0 \leq j \leq m - 1$

- $0 \leq c[j] \leq n-1$ pour tout $0 \leq j \leq m-1$

Sous-tâches

1. (9 points) $c[j] = 0$ pour tout $0 \leq j \leq m-1$ et $n, m \leq 200$
2. (11 points) $n, m \leq 200$
3. (17 points) $n, m \leq 2000$
4. (30 points) $c[j] \leq 29$ (pour tout $0 \leq j \leq m-1$) et $r[i] \leq 29$ (pour tout $0 \leq i \leq n-1$)
5. (33 points) Pas de contrainte supplémentaire.

Évaluateur d'exemple

L'évaluateur d'exemple lit l'entrée au format suivant :

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

L'évaluateur d'exemple affiche la valeur renvoyée par `find_reachable` au format suivant :

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