Klucze (Keys)

Architekt Tymoteusz zaprojektował nową grę. W tej grze jest n pokoi ponumerowanych od 0 do $n-1$. Początkowo w każdym pokoju znajduje się dokładnie jeden klucz. Każdy klucz ma typ będący liczbą całkowitą pomiędzy 0 a $n-1$ włącznie. Typ klucza w i-tym pokoju ($0 \leq i \leq n-1$) to $r[i]$. W różnych pokojach mogą być klucze o takim samym typie, tzn. wartości $r[i]$ niekoniecznie są parami różne.

Jest także m **dwukierunkowych** korytarzy w grze, ponumerowanych od 0 do $m-1$. Korytarz j ($0 \leq j \leq m-1$) łączy parę różnych pokoi $u[j]$ oraz $v[j]$. Para pokoi może być połączona wieloma korytarzami.

W grze bierze udział jeden gracz, który zbiera klucze oraz przemieszcza się między pokojami korzystając z korytarzy. Mówimy, że gracz **przechodzi** przez korytarz j, gdy używa tego korytarza w celu przemieszczenia się z pokoju $u[j]$ do pokoju $v[j]$, lub na odwrót. Gracz może przejść przez korytarz j, gdy zdobył wcześniej klucz o typie $c[j]$.

W dowolnym momencie gry, gracz jest w pewnym pokoju x oraz może wykonać jedną z dwóch akcji:

- zabranie klucza z pokoju x, którego typ jest $r[x]$ (o ile nie zabrał go wcześniej),
- przejście przez korytarz j, gdzie zachodzi $u[j] = x$ albo $v[j] = x$, jeżeli gracz zdobył wcześniej klucz o typie $c[j]$. Gracz **nigdy** nie pozbywa się klucza, jeżeli już go zabrał.

Gracz **zaczyna** grę w pewnym pokoju s nie posiadając żadnych kluczy. Z pokoju s można **dotrzeć** do pokoju t, jeżeli gracz zaczynający grę w pokoju s może wykonać pewną sekwencję akcji opisanych powyżej, żeby dostać się do pokoju t.

Dla każdego pokoju i ($0 \leq i \leq n-1$), niech liczba pokoi, do których można dotrzeć z pokoju i, wynosi $p[i]$. Tymoteusz chce wiedzieć jaki jest zbiór indeksów i osiągających minimalną wartość $p[i]$ dla $0 \leq i \leq n-1$.

Szczegóły implementacyjne

Masz zaimplementować poniższą funkcję:

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

- r: tablica o długości n. Dla każdego i ($0 \leq i \leq n-1$), klucz w i-tym pokoju jest typu $r[i]$.
- u, v: dwie tablice o długości m. Dla każdego j ($0 \leq j \leq m-1$), korytarz j łączy pokoje $u[j]$ oraz $v[j]$.
- c: tablica o długości m. Dla każdego j ($0 \leq j \leq m-1$), typ klucza potrzebnego do przejścia przez j-ty korytarz to $c[j]$.

- Ta funkcja powinna zwracać tablicę a o długości n. Dla każdego $0 \leq i \leq n-1$, wartość $a[i]$ powinna wynosić 1 jeżeli dla każdego j takiego, że $0 \leq j \leq n-1$, zachodzi $p[i] \leq p[j]$. W przeciwnym przypadku, wartość $a[i]$ powinna wynosić 0.

Przykłady

Przykład 1

Rozważmy poniższe wywołanie:

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

Jeżeli gracz zaczyna grę w pokoju 0, to może wykonać na przykład poniższą sekwencję akcji:

Aktualny pokój	Akcja
0	Zbierz klucz o typie 0
0	Przejdź przez korytarz 0 do pokoju 1
1	Zbierz klucz o typie 1
1	Przejdź przez korytarz 2 do pokoju 2
2	Przejdź przez korytarz 2 do pokoju 1
1	Przejdź przez korytarz 3 do pokoju 3

W takim razie z pokoju 0 można dotrzeć do pokoju 3. Podobnie, można skonstruować sekwencje ruchów wykazujące, że z pokoju 0 można dotrzeć do wszystkich pokoi, z czego wynika $p[0] = 4$. Poniższa tabelka przedstawia dla każdego pokoju startowego pokoje, do których można dotrzeć:

Pokój startowy i	Osiągalne pokoje	$p[i]$
0	$[0, 1, 2, 3]$	4
1	$[1, 2]$	2
2	$[1, 2]$	2
3	$[1, 2, 3]$	3

Najmniejsza wartość $p[i]$ dla wszystkich pokoi wynosi 2 oraz jest ona osiągalna tylko dla $i = 1$ albo $i = 2$. Dlatego to wywołanie powinno dać wynik $[0, 1, 1, 0]$.

Przykład 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])
```

Poniższa tabelka przedstawia dla każdego pokoju startowego pokoje, do których można dotrzeć:

Pokój startowy i	Osiągalne pokoje	$p[i]$
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

Najmniejsza wartość $p[i]$ dla wszystkich pokoi wynosi 2 oraz jest ona osiągalna tylko dla $i \in \{1, 2, 4, 6\}$. Dlatego to wywołanie powinno dać wynik $[0, 1, 1, 0, 1, 0, 1]$.

Przykład 3

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

Poniższa tabelka przedstawia dla każdego pokoju startowego pokoje, do których można dotrzeć:

Pokój startowy i	Osiągalne pokoje	$p[i]$
0	$[0, 1]$	2
1	$[0, 1]$	2
2	$[2]$	1

Najmniejsza wartość $p[i]$ dla wszystkich pokoi wynosi 2 oraz jest ona osiągalna tylko dla $i = 2$. Dlatego to wywołanie powinno dać wynik $[0, 0, 1]$.

Ograniczenia

- $2 \leq n \leq 300\,000$
- $1 \leq m \leq 300\,000$
- $0 \leq r[i] \leq n - 1$ dla każdego $0 \leq i \leq n - 1$
- $0 \leq u[j], v[j] \leq n - 1$ oraz $u[j] \neq v[j]$ dla każdego $0 \leq j \leq m - 1$

- $0 \leq c[j] \leq n-1$ dla każdego $0 \leq j \leq m-1$

Podzadania

1. (9 punktów) $c[j] = 0$ dla każdego $0 \leq j \leq m-1$ oraz $n, m \leq 200$
2. (11 punktów) $n, m \leq 200$
3. (17 punktów) $n, m \leq 2000$
4. (30 punktów) $c[j] \leq 29$ (dla każdego $0 \leq j \leq m-1$) oraz $r[i] \leq 29$ (dla każdego $0 \leq i \leq n-1$)
5. (33 punktów) Brak dodatkowych ograniczeń.

Przykładowa sprawdzaczka

Przykładowa sprawdzaczka wczytuje wejście w poniższym formacie:

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

Przykładowa sprawdzaczka wypisuje wynik zwracany przez wywołanie funkcji `find_reachable` w poniższym formacie:

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