Keys

O arquiteto Timothy desenhou um novo jogo de fuga. Neste jogo, há n salas numeradas de 0 a $n-1$. Inicialmente, cada sala contém exatamente uma chave. Cada chave tem um tipo, que é um inteiro entre 0 e $n-1$, inclusive. O tipo da chave na sala i ($0 \leq i \leq n-1$) é $r[i]$. Nota que múltiplas salas podem conter chaves do mesmo tipo, isto é, os valores $r[i]$ não são necessariamente distintos.

Há ainda m conectores **bidirecionais** no jogo, numerados de 0 a $m-1$. O conector j ($0 \leq j \leq m-1$) liga um par de salas distintas $u[j]$ e $v[j]$. Um par de salas pode ser ligado por múltiplos conectores.

O jogo é jogado por um único jogador que coleciona as chaves e move-se entre as salas atravessando os conectores. Dizemos que o jogador **atravessa** o conector j quando usa este conector para se movimentar da sala $u[j]$ para a sala $v[j]$, ou vice-versa. O jogador pode apenas atravessar o conector j se tiver colecionado a chave do tipo $c[j]$ antes.

A qualquer altura durante o jogo, o jogador está numa certa sala x e pode fazer um de dois tipos de ações:

- colecionar a chave na sala x, cujo tipo é $r[x]$ (caso ainda não a tenha colecionado antes),
- atravessar o conector j, onde ou $u[j] = x$ ou $v[j] = x$, apenas se o jogador tiver colecionado a chave do tipo $c[j]$ antes. Nota que o jogador **nunca** descarta uma chave que já colecionou.

O jogador **começa** o jogo numa certa sala s sem ter nenhuma chave. Uma sala t é **atingível** a partir de uma sala s se o jogador quando começa o jogo na sala s consegue fazer alguma sequência de ações como descrito em cima e chegar à sala t.

Para cada sala i ($0 \leq i \leq n-1$), dizemos que o número de salas atingíveis a partir da sala i é $p[i]$. O Timothy quer saber o conjunto de índices i que atingem o valor mínimo de $p[i]$ para $0 \leq i \leq n-1$.

Detalhes de Implementação

Deves implementar a seguinte função:

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

- r: um array de comprimento n. Para cada i ($0 \leq i \leq n-1$), a chave na sala i é do tipo $r[i]$.

- u, v: dois arrays de comprimento m. Para cada j ($0 \leq j \leq m-1$), o conector j liga as salas $u[j]$ e $v[j]$.
- c: um array de comprimento m. Para cada j ($0 \leq j \leq m-1$), o tipo de chave necessário para atravessar o conector j é $c[j]$.
- Esta função deve returnar um array a de comprimento n. Para cada $0 \leq i \leq n-1$, o valor de $a[i]$ deve ser 1 para cada j tal que $0 \leq j \leq n-1$, $p[i] \leq p[j]$. Caso contrário, o valor de $a[i]$ deve ser 0.

Exemplos

Exemplo 1

Considera a seguinte chamada:

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

Se o jogador começar o jogo na sala 0, pode fazer a seguinte sequência de ações:

Sala atual	Ação
0	Coleciona uma chave do tipo 0
0	Atravessa o conector 0 para a sala 1
1	Coleciona uma chave do tipo 1
1	Atravessa o conector 2 para a sala 2
2	Atravessa o conector 2 para a sala 1
1	Atravessa o conector 3 para a sala 3

Logo a sala 3 é atingível da sala 0. De forma similar, podemos construir sequências que mostram que todas as salas são atingíveis desde a sala 0, o que implica que $p[0] = 4$. A tabela abaixo mostra as salas atingíveis para cada sala inicial:

Sala inicial i	Salas atingíveis	$p[i]$
0	$[0, 1, 2, 3]$	4
1	$[1, 2]$	2
2	$[1, 2]$	2
3	$[1, 2, 3]$	3

O menor valor de $p[i]$ entre todas as salas é 2 e este é atingível para $i = 1$ ou $i = 2$. Logo, esta função deve retornar $[0, 1, 1, 0]$.

Exemplo 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])
```

A tabela abaixo mostra todas as salas atingíveis:

Sala inicial i	Salas atingíveis	$p[i]$
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

O menor valor de $p[i]$ entre todas as salas é 2 e este é atingível para $i \in \{1, 2, 4, 6\}$. Logo, esta função deve retornar $[0, 1, 1, 0, 1, 0, 1]$.

Example 3

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

A tabela abaixo mostra todas as salas atingíveis:

Sala inicial i	Salas atingíveis	$p[i]$
0	$[0, 1]$	2
1	$[0, 1]$	2
2	$[2]$	1

O menor valor de $p[i]$ entre todas as salas é 2 e este é atingível quando $i = 2$. Logo, esta função deve retornar $[0, 0, 1]$.

Restrições

- $2 \leq n \leq 300\,000$
- $1 \leq m \leq 300\,000$

- $0 \leq r[i] \leq n-1$ para todo $0 \leq i \leq n-1$
- $0 \leq u[j], v[j] \leq n-1$ e $u[j] \neq v[j]$ para todo $0 \leq j \leq m-1$
- $0 \leq c[j] \leq n-1$ para todo $0 \leq j \leq m-1$

Subtarefas

1. (9 pontos) $c[j] = 0$ para todo $0 \leq j \leq m-1$ e $n, m \leq 200$
2. (11 pontos) $n, m \leq 200$
3. (17 pontos) $n, m \leq 2000$
4. (30 pontos) $c[j] \leq 29$ (para todo $0 \leq j \leq m-1$) e $r[i] \leq 29$ (para todo $0 \leq i \leq n-1$)
5. (33 pontos) Sem restrições adicionais.

Avaliador Exemplo

O avaliador exemplo lê o input no seguinte formato:

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

O avaliador exemplo escreve o valor de retorno de `find_reachable` no seguinte formato:

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