Time limit: 2.0s **Memory limit:** 512M

Timothy the architect has designed a new escape game. In this game, there are n rooms numbered from 0 to n-1. Initially, each room contains exactly one key. Each key has a type, which is an integer between 0 and n-1, inclusive. The type of the key in room i $(0 \le i \le n-1)$ is r[i]. Note that multiple rooms may contain keys of the same type, i.e., the values r[i] are not necessarily distinct.

There are also m bidirectional connectors in the game, numbered from 0 to m-1. Connector j ($0 \le j \le m-1$) connects a pair of different rooms u[j] and v[j]. A pair of rooms can be connected by multiple connectors.

The game is played by a single player who collects the keys and moves between the rooms by traversing the connectors. We say that the player **traverses** connector j when they use this connector to move from room u[j] to room v[j], or vice versa. The player can only traverse connector j if they have collected a key of type c[j] before.

At any point during the game, the player is in some room x and can perform two types of actions:

- collect the key in room x, whose type is r[x] (unless they have collected it already),
- traverse a connector j, where either u[j] = x or v[j] = x, if the player has collected a key of type c[j] beforehand. Note that the player **never** discards a key they have collected.

The player **starts** the game in some room s not carrying any keys. A room t is **reachable** from a room s, if the player who starts the game in room s can perform some sequence of actions described above, and reach room t.

For each room i $(0 \le i \le n-1)$, denote the number of rooms reachable from room i as p[i]. Timothy would like to know the set of indices i that attain the minimum value of p[i] across $0 \le i \le n-1$.

Implementation Details

You are to implement the following procedure:

```
std::vector<int> find_reachable(std::vector<int> r, std::vector<int> u, std::vector<int> v,
std::vector<int> c)
```

- r: an array of length n. For each i $(0 \le i \le n-1)$, the key in room i is of type r[i].
- u, v: two arrays of length m. For each j ($0 \le j \le m-1$), connector j connects rooms u[j] and v[j].
- c: an array of length m. For each j $(0 \le j \le m-1)$, the type of key needed to traverse connector j is c[j].
- This procedure should return an array a of length n. For each $0 \le i \le n-1$, the value of a[i] should be 1 if for every j such that $0 \le j \le n-1$, $p[i] \le p[j]$. Otherwise, the value of a[i] should be 0.

Examples

Example 1

Consider the following call:

If the player starts the game in room 0, they can perform the following sequence of actions:

Current room	Action
0	Collect key of type 0
0	Traverse connector 0 to room 1
1	Collect key of type 1
1	Traverse connector 2 to room 2
2	Traverse connector 2 to room 1
1	Traverse connector 3 to room 3

Hence room 3 is reachable from room 0. Similarly, we can construct sequences showing that all rooms are reachable from room 0, which implies p[0] = 4. The table below shows the reachable rooms for all starting rooms:

Starting room i	Reachable rooms	p[i]
0	[0, 1, 2, 3]	4
1	[1,2]	2
2	[1,2]	2
3	[1, 2, 3]	3

The smallest value of p[i] across all rooms is 2, and this is attained for i=1 or i=2. Therefore, this procedure should return [0,1,1,0].

Example 2

The table below shows the reachable rooms:

Starting room i	Reachable rooms	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

The smallest value of p[i] across all rooms is 2, and this is attained for $i \in \{1, 2, 4, 6\}$. Therefore, this procedure should return [0, 1, 1, 0, 1, 0, 1].

Example 3

```
find_reachable({0, 0, 0}, {0}, {1}, {0})
```

The table below shows the reachable rooms:

Starting room i	Reachable rooms	p[i]
0	[0,1]	2
1	[0,1]	2
2	[2]	1

The smallest value of p[i] across all rooms is 1, and this is attained when i=2. Therefore, this procedure should return [0,0,1].

Constraints

- $2 \le n \le 300\,000$
- $1 \le m \le 300\,000$
- $\bullet \quad 0 \leq r[i] \leq n-1 \text{ for all } 0 \leq i \leq n-1$
- $0 \leq u[j], v[j] \leq n-1$ and u[j]
 eq v[j] for all $0 \leq j \leq m-1$
- $\bullet \quad 0 \leq c[j] \leq n-1 \text{ for all } 0 \leq j \leq m-1$

Subtasks

- 1. (9 points) c[j]=0 for all $0\leq j\leq m-1$ and $n,m\leq 200$
- 2. (11 points) $n,m \leq 200$
- 3. (17 points) $n, m \leq 2000$

- 4. (30 points) $c[j] \leq 29$ (for all $0 \leq j \leq m-1$) and $r[j] \leq 29$ (for all $0 \leq i \leq n-1$)
- 5. (33 points) No additional constraints.

Sample Grader

The sample grader reads the input in the following format:

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

The sample grader prints the return value of find_reachable in the following format:

• line 1: $a[0] \ a[1] \ \dots \ a[n-1]$

Attachment Package

The sample grader and sample test cases are available here: keys.zip.