Ачкычтар

Архитектор Тимофей жаңы "качуу" оюнун жасады. Бул оюнда 0дөн $(n-1)$ге чейинки n бөлмө бар. Башында, ар бир бөлмөдө бирден ачкыч бар. Ар бир ачкычтын түрү бар, ал $0..n-1$ ичинде. i-бөлмөдөгү ($0 \leq i \leq n-1$) ачкычтын түрү $r[i]$. Бир нече бөлмөдө бир эле түрдөгү ачкычтар болушу мүмкүн, башкача айтканда $r[i]$ мааниси бирдей болушу мүмкүн.

Ошондой эле оюнда 0дөн $(m-1)$ге чейинки m **эки багыттуу** туташтыргыч бар. j-туташтыргыч ($0 \leq j \leq m-1$) $u[j]$-бөлмөнү жана $v[j]$-бөлмөнү бириктирет. Бир жуп бөлмөнү бир нече туташтыргыч аркылуу туташтырса болот.

Оюнду ачкычтарды чогулткан жалгыз оюнчу ойнойт жана туташтыргычтарды аралап бөлмөлөрдүн ортосунда жүрөт. Эгерде оюнчу $u[j]$-бөлмөдөн $v[j]$-бөлмөгө өтүү үчүн, же тескерисинче өтүү үчүн туташтыргычты колдонсо, анда оюнчу ал j-туташтыргычты **өтөт** деп айтабыз. Эгерде оюнчу $c[j]$-түрүндөгү ачкычты мурда чогултса гана анда оюнчу j-туташтыргычты өтө алат.

Оюндун жүрүшүндө каалаган учурда оюнчу x-бөлмөдө болот жана төмөндөгү эки аракетти жасай алат:

- ачкычты x-бөлмөгө чогулта алат, анын түрү $r[x]$ (эгерде ал буга чейин чогултушпаса),
- j-туташтыргычты өтө алат, ($u[j] = x$ же $v[j] = x$), эгерде оюнчу алдын ала $c[j]$-түрүндөгү ачкычты чогултуп алса. Оюнчу **эч качан** чогулткан ачкычты таштабай тургандыгын эске алыңыз.

Оюнчу оюнду кандайдыр бир s-бөлмөдө ачкычсыз **баштайт** . Эгерде оюнчу s-бөлмөдөн оюнду баштап жогоруда айтылган аракеттердин ырааттуулугун аткарып, t-бөлмөгө жете алса, анда t-бөлмө s бөлмөдөн **жетилүүчү** болот.

Ар бир i-бөлмө үчүн ($0 \leq i \leq n-1$), i-бөлмөдөн жетилүүчү бөлмөлөрдүн санын $p[i]$ менен деп белгилеңиз. $0 \leq i \leq n-1$ аралыгында $p[i]$нин минималдык маанисине жеткен i индекстердин көптүгүн Тимофей билгиси келет.

Implementation Details

You are to implement the following procedure:

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

- r: an array of length n. For each i ($0 \leq i \leq n-1$), the key in room i is of type $r[i]$.
- u, v: two arrays of length m. For each j ($0 \leq j \leq m-1$), connector j connects rooms $u[j]$ and $v[j]$.

- c: an array of length m. For each j ($0 \leq j \leq m-1$), the type of key needed to traverse connector j is $c[j]$.
- This procedure should return an array s of length n. For each $0 \leq i \leq n-1$, the value of $s[i]$ should be 1 if for every j such that $0 \leq j \leq n-1$, $p[i] \leq p[j]$. Otherwise, the value of $s[i]$ should be 0.

Examples

Example 1

Consider the following call:

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

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

```
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])
```

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

- $2 \leq n \leq 300\,000$
- $1 \leq m \leq 300\,000$
- $0 \leq r[i] \leq n - 1$ for all $0 \leq i \leq n - 1$
- $0 \leq u[j], v[j] \leq n - 1$ and $u[j] \neq v[j]$ for all $0 \leq j \leq m - 1$

- $0 \le c[j] \le n-1$ for all $0 \le j \le m-1$

Subtasks

1. (9 points) $c[j] = 0$ for all $0 \le j \le m-1$ and $n, m \le 200$
2. (11 points) $n, m \le 200$
3. (17 points) $n, m \le 2000$
4. (30 points) $c[j] \le 29$ (for all $0 \le j \le m-1$) and $r[i] \le 29$ (for all $0 \le i \le n-1$)
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 $3+j$ ($0 \le j \le m-1$): $u[j]$ $v[j]$ $c[j]$

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

- line 1: $s[0]$ $s[1]$ $\ldots$ $s[n-1]$