鍵 (Keys)

Timothy は新しい脱出ゲームを設計した. このゲームでは n 個の部屋が使用され, 部屋には 0 から $n-1$ までの番号が付けられている. はじめ, それぞれの部屋にはちょうど 1 個の鍵がある. それぞれの鍵にはタイプが定められており, それは 0 以上 $n-1$ 以下の整数で表される. 部屋 i ($0 \leq i \leq n-1$) にある鍵のタイプは $r[i]$ である. 複数の部屋に同じタイプの鍵があるかもしれない. 言い換えるなら, $r[i]$ は互いに異なっているとは限らない.

また, ゲームでは m 本の **双方向の** 通路も使用され, 通路には 0 から $m-1$ までの番号が付けられている. 通路 j ($0 \leq j \leq m-1$) は 2 つの異なる部屋 $u[j]$ と $v[j]$ をつないでいる . 複数の通路が同じ組み合わせの部屋同士をつないでいるかもしれない.

このゲームは一人用である. プレイヤーは部屋で鍵を手に入れながら通路を通行していく. プレイヤーが通路 j を **通行する** とは, プレイヤーが通路 j を使用し, 部屋 $u[j]$ から部屋 $v[j]$ に移動するか, 部屋 $v[j]$ から部屋 $u[j]$ に移動することを指す. ただし, プレイヤーが通路 j を通行できるのは, 事前にタイプ $c[j]$ の鍵をすでに手に入れていた場合に限られる.

ゲーム中, ある部屋 x にいるプレイヤーは以下のいずれかの行動をすることができる.

- 部屋 x にあるタイプ $r[x]$ の鍵を手に入れる (ただし, その部屋の鍵をまだ手に入れていない場合に限る).
- $u[j] = x$ もしくは $v[j] = x$ を満たし, タイプ $c[j]$ の鍵をすでに手に入れているような j について, 通路 j を通行する. 注意として, 一度手に入れた鍵を **失うことはない** .

プレイヤーはある部屋 s から鍵を一つも持たない状態でゲームを **開始する** . ある部屋 t が部屋 s から **到達可能** であるとは, プレイヤーが部屋 s からゲームを開始したとき, ある一連の行動をすることで, 部屋 t に移動できるということである.

それぞれの部屋 i ($0 \leq i \leq n-1$) について, 部屋 i から到達可能な部屋の数を $p[i]$ とする. Timothy は $0 \leq i \leq n-1$ の中で $p[i]$ が最小となる i をすべて知りたい.

実装の詳細

あなたは以下の関数を実装しなさい.

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

- r: 長さ n の配列である. それぞれの i ($0 \leq i \leq n-1$) について, 部屋 i にある鍵のタイプは $r[i]$ である.
- u, v: それぞれ長さ m の配列である. それぞれの j ($0 \leq j \leq m-1$) について, 通路 j は部屋 $u[j]$ と部屋 $v[j]$ をつないでいる.

- c: 長さ m の配列である. それぞれの j ($0 \leqq j \leqq m-1$) について, 通路 j を通行するためにはタイプ $c[j]$ の鍵が必要である.
- この関数は長さ n の配列 a を戻り値とする必要がある. それぞれの i ($0 \leqq i \leqq n-1$) について, すべての j ($0 \leqq j \leqq n-1$) が $p[i] \leqq p[j]$ を満たすならば, $a[i] = 1$ とせよ. そうでない場合, $a[i] = 0$ とせよ.

入出力例

入出力例 1

以下の関数呼び出しを考える.

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

プレイヤーが部屋 0 からゲームを開始したとき, 以下のように行動をすることができる.

プレイヤーがいる部屋	行動
0	タイプ 0 の鍵を手に入れる
0	通路 0 を通行し, 部屋 1 に移動する
1	タイプ 1 の鍵を手に入れる
1	通路 2 を通行し, 部屋 2 に移動する
2	通路 2 を通行し, 部屋 1 に移動する
1	通路 3 を通行し, 部屋 3 に移動する

以上より, 部屋 3 は部屋 0 から到達可能である. 同様に, すべての部屋について部屋 0 から到達可能であることが示せるため, $p[0] = 4$ となる. 以下の表はすべての部屋について, その部屋からゲームを開始したときに到達可能な部屋を示している.

ゲームを開始した部屋 i	到達可能な部屋	$p[i]$
0	$[0, 1, 2, 3]$	4
1	$[1, 2]$	2
2	$[1, 2]$	2
3	$[1, 2, 3]$	3

すべての部屋の中で最小の $p[i]$ は 2 であり, $p[i] = 2$ となるような i は $1, 2$ である. そのため, この関数は $[0, 1, 1, 0]$ を戻り値とする必要がある.

入出力例 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])
```

以下の表は到達可能な部屋を表す.

ゲームを開始した部屋 i	到達可能な部屋	$p[i]$
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

すべての部屋の中で最小の $p[i]$ は 2 であり, $p[i] = 2$ となるような i は $1, 2, 4, 6$ である. そのため, この関数は $[0, 1, 1, 0, 1, 0, 1]$ を戻り値とする必要がある.

入出力例 3

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

以下の表は到達可能な部屋を表す.

ゲームを開始した部屋 i	到達可能な部屋	$p[i]$
0	$[0, 1]$	2
1	$[0, 1]$	2
2	$[2]$	1

すべての部屋の中で最小の $p[i]$ は 1 であり, $p[i] = 1$ となるような i は 2 である. そのため, この関数は $[0, 0, 1]$ を戻り値とする必要がある.

制約

- $2 \leqq n \leqq 300\,000$
- $1 \leqq m \leqq 300\,000$
- $0 \leqq r[i] \leqq n - 1$ ($0 \leqq i \leqq n - 1$)
- $0 \leqq u[j], v[j] \leqq n - 1$, $u[j] \neq v[j]$ ($0 \leqq j \leqq m - 1$)

- $0 \leqq c[j] \leqq n-1$ ($0 \leqq j \leqq m-1$)

小課題

1. (9 点) $n, m \leqq 200$, $c[j] = 0$ ($0 \leqq j \leqq m-1$)
2. (11 点) $n, m \leqq 200$
3. (17 点) $n, m \leqq 2000$
4. (30 点) $c[j] \leqq 29$ ($0 \leqq j \leqq m-1$), $r[i] \leqq 29$ ($0 \leqq i \leqq n-1$)
5. (33 点) 追加の制約はない.

採点プログラムのサンプル

採点プログラムのサンプルは入力を以下の形式で読み込む:

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

採点プログラムのサンプルは以下の形式で `find_reachable` の戻り値を表示する:

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