

# Keys

Timoteo el arquitecto diseñó un nuevo juego de escape. En el juego hay  $n$  salones, numerados de  $0$  a  $n - 1$  inclusive. Al principio del juego en cada salón hay exactamente una llave. Cada llave tiene un tipo, que es un entero entre  $0$  y  $n - 1$  inclusive. La llave que está en el salón  $i$  ( $0 \leq i \leq n - 1$ ) tiene tipo  $r[i]$ . Notar que puede haber llaves del mismo tipo en salones distintos, es decir los valores  $r[i]$  no son necesariamente distintos.

Hay también  $m$  pasillos **bidireccionales** en el juego, que se numeran del  $0$  al  $m - 1$ . El pasillo  $j$  ( $0 \leq j \leq m - 1$ ) conecta dos salones diferentes,  $u[j]$  y  $v[j]$ . Puede haber más de un pasillo entre los mismos dos salones.

Hay un solo jugador en el juego, que agarra llaves y se mueve entre los salones a través de los pasillos. Decimos que el jugador **atraviesa** un pasillo  $j$  cuando usa este pasillo para moverse del salón  $u[j]$  al salón  $v[j]$ , o vice versa. Para poder atravesar el pasillo  $j$ , el jugador debe primero haber agarrado alguna llave de tipo  $c[j]$ .

Si el jugador está en el salón  $x$ , puede hacer dos cosas:

- agarrar la llave que está en el salón  $x$ , que es de tipo  $r[x]$  (a menos que la haya agarrado ya)
- atravesar el pasillo  $j$  con  $u[j] = x$  o  $v[j] = x$ , siempre y cuando el jugador ya haya agarrado una llave de tipo  $c[j]$ . Notar que esto **no** gasta la llave: una vez que el jugador agarra una llave la tiene disponible para siempre.

El jugador **arranca** el juego en un salón  $s$ , sin tener ninguna llave. Un salón  $t$  es **alcanzable** si el jugador puede llegar al salón  $t$  haciendo una serie de acciones permitidas.

Para cada salón  $i$  ( $0 \leq i \leq n - 1$ ), sea  $p[i]$  la cantidad de salones alcanzables desde el salón  $i$ . Timoteo quiere saber que conjunto de índices  $i$  entre ( $0 \leq i \leq n - 1$ ) minimizan el valor de  $p[i]$ .

## Detalles de Implementación

Tenés que implementar la siguiente función:

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

- $r$ : es un vector de longitud  $n$ . Para cada  $i$  ( $0 \leq i \leq n - 1$ ), la llave en el salón  $i$  es de tipo  $r[i]$ .
- $u, v$ : son dos vectores de longitud  $m$ . Para cada  $j$  ( $0 \leq j \leq m - 1$ ), el pasillo  $j$  va entre los salones  $u[j]$  y  $v[j]$ .
- $c$ : vector de longitud  $m$ . Para cada  $j$  ( $0 \leq j \leq m - 1$ ),  $c[j]$  es el tipo de llave que se necesita para atravesar el pasillo  $j$ .

- La función debe devolver un vector  $a$  de longitud  $n$ . Para cada  $0 \leq i \leq n - 1$ , el valor  $a[i]$  debe ser 1 si para cada  $j$  tal que  $0 \leq j \leq n - 1$  se cumple  $p[i] \leq p[j]$ . Caso contrario, el valor  $a[i]$  debe ser 0.

## Ejemplos

### Ejemplo 1

Considera la siguiente llamada:

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

Si el jugador arranca en el salón 0, puede hacer la siguiente secuencia de acciones:

Salón actual	Acción
0	Agarrar una llave de tipo 0
0	Atravesar el pasillo 0 hasta el salón 1
1	Agarrar una llave de tipo 1
1	Atravesar el pasillo 2 hasta el salón 2
2	Atravesar el pasillo 2 hasta el salón 1
1	Atravesar el pasillo 3 hasta el salón 3

Por lo tanto, el salón 3 es alcanzable desde el salón 0. Similarmente, podemos construir sucesiones de acciones que muestran que todos los salones son alcanzables desde el salón 0, por lo que  $p[0] = 4$ .

En la tabla de abajo se muestra qué salones son alcanzables desde cada salón.

Salón inicial $i$	Salones alcanzables	$p[i]$
0	[0, 1, 2, 3]	4
1	[1, 2]	2
2	[1, 2]	2
3	[1, 2, 3]	3

El menor valor de  $p[i]$  sobre todos los salones es 2, y se alcanza para  $i = 1$  y  $i = 2$ . La función debe entonces devolver  $[0, 1, 1, 0]$ .

### Ejemplo 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])
```

En la tabla de abajo se muestra qué salones son alcanzables desde cada salón.

Salón inicial $i$	Salones alcanzables	$p[i]$
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

El menor valor de  $p[i]$  sobre todos los salones es 2, y se alcanza para  $i \in \{1, 2, 4, 6\}$ . La función debe entonces devolver [0, 1, 1, 0, 1, 0, 1].

### Ejemplo 3

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

En la tabla de abajo se muestra qué salones son alcanzables desde cada salón.

Salón inicial $i$	Salones alcanzables	$p[i]$
0	[0, 1]	2
1	[0, 1]	2
2	[2]	1

El menor valor de  $p[i]$  sobre todos los salones es 1, y se alcanza para  $i = 2$ . La función debe entonces devolver [0, 0, 1].

## Restricciones

- $2 \leq n \leq 300\,000$
- $1 \leq m \leq 300\,000$
- $0 \leq r[i] \leq n - 1$  para todo  $0 \leq i \leq n - 1$
- $0 \leq u[j], v[j] \leq n - 1$  y  $u[j] \neq v[j]$  para todo  $0 \leq j \leq m - 1$

- $0 \leq c[j] \leq n - 1$  para todo  $0 \leq j \leq m - 1$

## Subtareas

1. (9 puntos)  $c[j] = 0$  para todo  $0 \leq j \leq m - 1$  y  $n, m \leq 200$
2. (11 puntos)  $n, m \leq 200$
3. (17 puntos)  $n, m \leq 2000$
4. (30 puntos)  $c[j] \leq 29$  (para todo  $0 \leq j \leq m - 1$ ) y  $r[i] \leq 29$  (para todo  $0 \leq i \leq n - 1$ )
5. (33 puntos) Sin restricciones adicionales.

## Evaluador Local

El evaluador local lee la input en el siguiente formato

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

El evaluador local imprime el vector que devuelve `find_reachable` en el siguiente formato

- línea 1:  $a[0] \ a[1] \ \dots \ a[n - 1]$