Hombre lobo

Existen N ciudades y M caminos en la prefectura de Ibaraki, Japón. Las ciudades están numeradas de 0 a $N-1$ en orden creciente de su población. Cada camino conecta un par de ciudades distintas, y puede ser recorrido en ambas direcciones. Es posible ir de cualquier ciudad a cualquier otra usando uno o más de estos caminos.

Tú planeaste Q viajes, numerados de 0 a $Q-1$. El viaje i ($0 \leq i \leq Q-1$) es un viaje de la ciudad S_i a la ciudad E_i.

Tú eres un hombre lobo. Tienes dos formas: la **forma humana** y la **forma de lobo**. Al comienzo de cada viaje tienes la forma de humano. Al final de cada viaje, debes tener la forma de lobo. Durante el viaje tienes que **transformarte** (cambiar de forma humana a forma de lobo) exactamente una vez, y esto tiene que pasar cuando tú estés en alguna ciudad (posiblemente S_i o E_i).

Vivir como hombre lobo no es fácil. Debes evitar ciudades poco habitadas cuando estás en forma humana, y evitar ciudades altamente habitadas cuando estás en forma de lobo. Para cada viaje i ($0 \leq i \leq Q-1$), hay dos umbrales L_i y R_i ($0 \leq L_i \leq R_i \leq N-1$) que indican qué ciudades deben ser evitadas. Mas específicamente, debes evitar las ciudades $0, 1, \ldots, L_i-1$ cuando tienes la forma humana, y las ciudades $R_i+1, R_i+2, \ldots, N-1$ cuando tienes la forma de lobo. Esto significa que en el viaje i, tú puedes solamente transformarte en una de las ciudades $L_i, L_i+1, \ldots, R_i$.

Tu tarea es determinar, por cada viaje, si es posible viajr de la ciudad S_i a la ciudad E_i en una forma que satisfaga las restricciones mencionadas anteriormente. La ruta que tomes puede tener un tamaño arbitrario.

Detalles de implementación

Debes implementar la siguiente función:

```
int[] check_validity(int N, int[] X, int[] Y, int[] S, int[] E, int[]
 L, int[] R)
```

- `N`: el número de ciudades.
- `X` e `Y`: arreglos de tamaño M. Por cada j ($0 \leq j \leq M-1$), la ciudad `X[j]` está directamente conectada a la ciudad `Y[j]` por un camino.
- `S`, `E`, `L`, y `R`: arreglos de tamaño Q, representando los viajes.

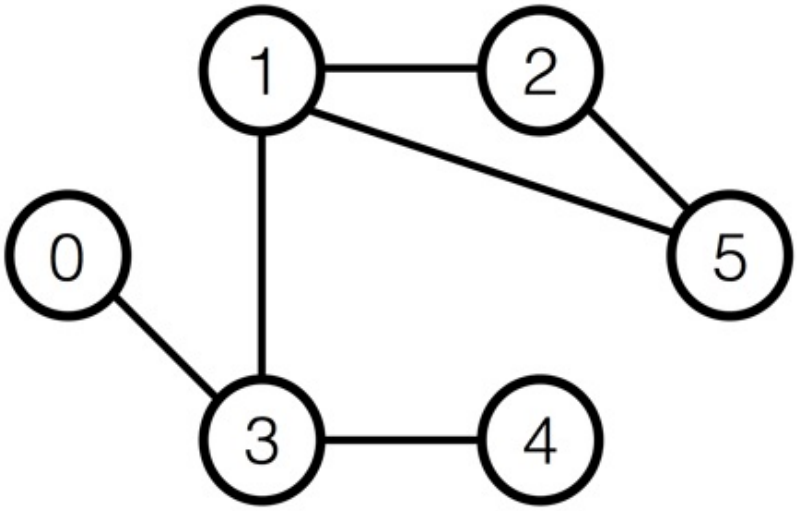
Note que los valores de M y Q son los tamaños de los arreglos, y pueden ser obtenidos comom es indicado en el aviso de implementación.

La función `check_validity` es llamada exactamente una vez por cada caso de prueba. Esta función debe retornar un arreglo A de enteros de tamaño Q. El valor de A_i ($0 \leq i \leq Q-1$) debe ser `1` si el viaje i es posible mientras se satisfagan las anteriormente mencionadas condiciones, o `0` de otra forma.

Ejemplo

Sean $N = 6$, $M = 6$, $Q = 3$, $X = [5, 1, 1, 3, 3, 5]$, $Y = [1, 2, 3, 4, 0, 2]$, $S = [4, 4, 5]$, $E = [2, 2, 4]$, $L = [1, 2, 3]$, and $R = [2, 2, 4]$.

El evaluador llama `check_validity(6, [5, 1, 1, 3, 3, 5], [1, 2, 3, 4, 0, 2], [4, 4, 5], [2, 2, 4], [1, 2, 3], [2, 2, 4])`.



Para el viaje 0, tú puedes viajar de la ciudad 4 a la ciudad 2 de la siguiente manera:

- Empezar en la ciudad 4(estás en forma humana).
- Moverte a la ciudad 3(estás en forma humana).
- Moverte a la ciudad 1(estás en forma humana).
- Transormarte en forma de lobo(estás en forma de lobo).
- Moverte a la ciudad 2(estás en forma de lobo).

Para los viajes 1 y 2, no puedes viajar entre las ciudades dadas.

Por lo tanto, tu programa debería retornar $[1, 0, 0]$.

Los archivos `sample-01-in.txt` y `sample-01-out.txt` en el paquete comprimido adjunto corresponde a este ejemplo. Otros ejemplos de entrada/salida también están disponibles en el paquete.

Restricciones

- $2 \leq N \leq 200\,000$
- $N - 1 \leq M \leq 400\,000$
- $1 \leq Q \leq 200\,000$
- Por cada $0 \leq j \leq M - 1$
  - $0 \leq X_j \leq N - 1$
  - $0 \leq Y_j \leq N - 1$
  - $X_j \neq Y_j$
- Puedes viajar de cualquier ciudad a cualquier ciudad usando los caminos.
- Cada par de ciudades están directamente conectadas por al menos un camino. En otras palabras, para todo $0 \leq j < k \leq M - 1$, $(X_j, Y_j) \neq (X_k, Y_k)$ y $(Y_j, X_j) \neq (X_k, Y_k)$.
- Cada par de ciudades están directamente conectadas a lo mucho un camino. En otras palabras, para todo $0 \leq j < k \leq M - 1$, $(X_j, Y_j) \neq (X_k, Y_k)$ and $(Y_j, X_j) \neq (X_k, Y_k)$.
- Por cada $0 \leq i \leq Q - 1$
  - $0 \leq L_i \leq S_i \leq N - 1$
  - $0 \leq E_i \leq R_i \leq N - 1$
  - $S_i \neq E_i$
  - $L_i \leq R_i$

Subtareas

1. (7 puntos) $N \leq 100$, $M \leq 200$, $Q \leq 100$
2. (8 puntos) $N \leq 3\,000$, $M \leq 6\,000$, $Q \leq 3\,000$
3. (34 puntos) $M = N - 1$ y ninguna ciudad está directamente conectada a más de dos ciudades (las ciudades están conectadas en una línea)
4. (51 puntos) Sin restricciones adicionales.

Evaluador simple

El evaluador simple lee las entradas en el siguiente formato:

- línea 1: N M Q
- línea $2 + j$ ($0 \leq j \leq M - 1$): X_j Y_j
- línea $2 + M + i$ ($0 \leq i \leq Q - 1$): S_i E_i L_i R_i

El evaluador simple imprime el valor de retorno de `check_validity` en el siguiente formato:

- línea $1 + i$ ($0 \leq i \leq Q - 1$): A_i