Lupo mannaro

Nella prefettura di Ibaraki, in Giappone, ci sono N città ed M strade. Le città sono numerate da 0 a $N-1$ in ordine crescente di popolazione. Ciascuna strada connette una coppia di città distinte, e può essere percorsa in entrambi i sensi. È possibile spostarsi da una qualsiasi città ad una qualsiasi altra città utilizzando una o più strade.

Per le tue escursioni hai pianificato Q viaggi, numerati da 0 a $Q-1$, per cui il viaggio i-esimo ($0 \leq i \leq Q-1$) consiste nello spostarsi dalla città S_i alla città E_i.

A complicare le cose però c'è il fatto che tu sei un lupo mannaro: puoi quindi assumere **forma umana** o **forma lupo**. All'inizio di ciascun viaggio sei in forma umana, mentre alla fine dello stesso dovrai essere in forma lupo. In qualche momento durante il viaggio dovrai quindi **trasformarti** (cambiare dalla forma umana alla forma lupo). La trasformazione può avvenire una ed una sola volta, e può avvenire solo quando ti trovi in una città (eventualmente, anche S_i o E_i).

La vita da lupo mannaro non è facile. È infatti consigliabile evitare di visitare città *poco popolate* quando si è in forma umana e, allo stesso modo, è bene non trovarsi in città *molto popolate* quando si è in forma lupo.

Nell'i-esimo viaggio ($0 \leq i \leq Q-1$), dovrai tenere conto di due "soglie" indicate con L_i e R_i ($0 \leq L_i \leq R_i \leq N-1$) che indicano quali città dovrai evitare. Più precisamente dovrai evitare le città $0, 1, \ldots, L_i-1$ quando sarai in forma umana, e le città $R_i+1, R_i+2, \ldots, N-1$ quando sarai in forma lupo. Questo implica che nel viaggio i ti potrai trasformare solo quando ti troverai su una tra le città $L_i, L_i+1, \ldots, R_i$.

Determina, per ciascun viaggio, se sia possibile o no spostarsi dalla città di partenza S_i verso quella di arrivo E_i in modo tale che i suddetti vincoli siano rispettati (il percorso che scegli può essere lungo quanto vuoi).

Dettagli implementativi

Devi implementare la seguente funzione:

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

- `N`: il numero di città.

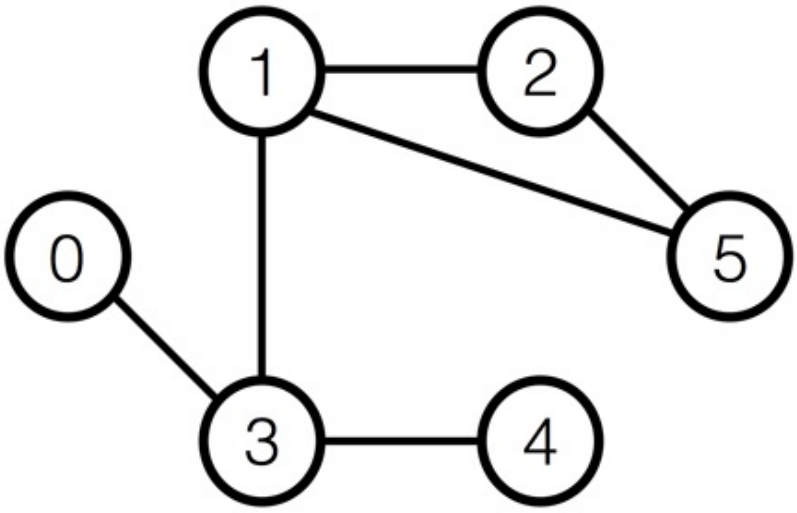
- `X` e `Y`: array di lunghezza M. Per ogni j ($0 \leq j \leq M-1$), esiste una strada che connette le città `X[j]` e `Y[j]`.
- `S`, `E`, `L`, e `R`: array di lunghezza Q, rappresentano i viaggi che hai pianificato.

Nota che le lunghezze M e Q non vengono fornite come parametri perché si possono ottenere (come indicato nelle "Note di implementazione") dagli array stessi.

La funzione `check_validity` viene chiamata esattamente una volta per ciascun caso di test. La funzione dovrà restituire un array A formato da Q numeri interi. Il valore di A_i ($0 \leq i \leq Q-1$) dovrà essere `1` qualora il viaggio i sia fattibile con i suddetti vincoli, oppure `0` se non dovesse essere fattibile.

Esempio

Siano $N = 6$, $M = 6$, $Q = 3$, $X = [5, 1, 1, 3, 3, 5]$, $Y = [1, 2, 3, 4, 0, 2]$, $S = [4, 4, 5]$, $E = [2, 2, 4]$, $L = [1, 2, 3]$, e $R = [2, 2, 4]$.
Il grader chiama `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])`.



Per il viaggio 0, ti puoi spostare dalla città 4 alla città 2 in questo modo:

- Comincia dalla città 4 (in forma umana)
- Spostati nella città 3 (in forma umana)
- Spostati nella città 1 (in forma umana)
- Esegui la trasformazione (in forma lupo)
- Spostati nella città 2 (in forma lupo)

Per i viaggi 1 e 2, non puoi spostarti tra le città pianificate. Il tuo programma dovrà quindi restituire $[1, 0, 0]$.

I file `sample-01-in.txt` e `sample-01-out.txt` nell'archivio compresso allegato al problema corrispondono all'esempio appena spiegato. Questo archivio contiene anche un input/output di esempio aggiuntivo.

Assunzioni

- $2 \leq N \leq 200\,000$
- $N - 1 \leq M \leq 400\,000$
- $1 \leq Q \leq 200\,000$
- Per ogni $0 \leq j \leq M - 1$
  - $0 \leq X_j \leq N - 1$
  - $0 \leq Y_j \leq N - 1$
  - $X_j \neq Y_j$
- È possibile spostarsi da qualsiasi città verso qualsiasi altra città con le strade fornite.
- Ogni coppia di città è direttamente connessa da al più una strada. In altre parole, per ogni $0 \leq j < k \leq M - 1$, $(X_j, Y_j) \neq (X_k, Y_k)$ e $(Y_j, X_j) \neq (X_k, Y_k)$.
- Per ogni $0 \leq i \leq Q - 1$
  - $0 \leq L_i \leq S_i$
  - $0 \leq E_i \leq R_i \leq N - 1$
  - $S_i \neq E_i$ ($0 \leq i \leq Q - 1$)
  - $0 \leq L_i \leq R_i \leq N - 1$

Subtask

1. (7 punti) $N \leq 100$, $M \leq 200$, $Q \leq 100$
2. (8 punti) $N \leq 3\,000$, $M \leq 6\,000$, $Q \leq 3\,000$
3. (34 punti) $M = N - 1$ ed ogni città è connessa ad al più 2 strade (le città formano una "linea")
4. (51 punti) Nessuna limitazione aggiuntiva

Grader di esempio

Il grader di esempio legge l'input nel seguente formato:

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

Il grader di esempio stampa il valore restituito da `check_validity` nel seguente formato:

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