Werewolf

În prefectura Ibaraki din Japonia există N orașe și M drumuri. Orașele sunt numerotate de la 0 la $N-1$ în ordine crescătoare a populației. Fiecare drum conectează o pereche de orașe distincte și poate fi traversat în ambele direcții. Puteți călători din orice oraș în orice alt oraș utilizând unul sau mai multe din aceste drumuri.

Planificați Q călătorii, numerotate de la 0 la $Q-1$. Călătoria i ($0 \leq i \leq Q-1$) este o călătorie din orașul S_i către orașul E_i.

Sunteți un vârcolac. Puteți lua două forme: **forma unui om** și **forma unui lup**. La începutul fiecărei călătorii aveți forma unui om. La sfârșitul fiecărei călătorii trebuie să aveți forma unui lup. Pe parcursul călătoriei trebuie să vă **transformați** (să vă schimbați din forma unui om în forma unui lup) exact o dată. Vă puteți transforma doar când sunteți într-un oraș (posibil S_i sau E_i).

Viața de vârcolac nu este una ușoară. Trebuie să evitați orașele mai puțin populate atunci când aveți forma unui om și orașele mai populate atunci când aveți forma unui lup. Pentru orice călătorie $i (0 \leq i \leq Q-1)$, există doi întregi L_i și R_i ($0 \leq L_i \leq R_i \leq N-1$), care indică orașele ce trebuie evitate. Mai exact, pentru călătoria i, trebuie să evitați orașele $0, 1, \ldots, L_i-1$ când aveți forma unui om și să evitați orașele $R_i+1, R_i+2, \ldots, N-1$ când aveți forma unui lup. Aceasta înseamnă că, în călătoria i, vă veți transforma în unul din orașele $L_i, L_i+1, \ldots, R_i$.

Sarcina dumneavoastră este să determinați pentru fiecare călătorie dacă este posibil să ajungeți din orașul S_i în orașul E_i, respectând condițiile descrise mai sus. Drumul ales poate fi de orice lungime.

Detalii de implementare

Trebuie să implementați următoarea funcție:

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

- `N`: numărul de orașe.
- `X` și `Y`: tablouri unidimensionale de dimensiune M. Pentru fiecare j ($0 \leq j \leq M-1$), orașul `X[j]` este conectat printr-un drum direct cu orașul `Y[j]`.
- `S`, `E`, `L`, și `R`: tablouri unidimensionale de dimensiune Q, reprezentând călătoriile.

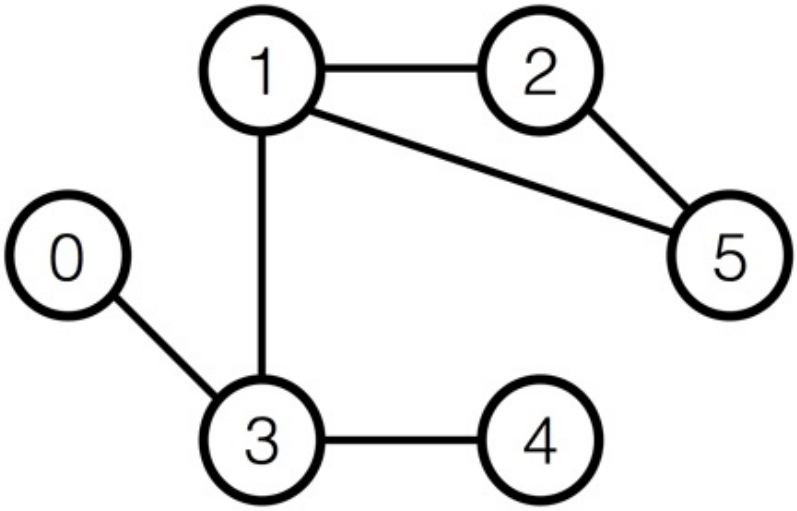
Luați la cunoștință că valorile M și Q reprezintă dimensiunea tablourilor și pot fi obținute după cum este indicat în Observațiile de implementare.

Funcția `check_validity` este apelată o singură dată pentru fiecare test. Această funcție trebuie să întoarcă un tablou unidimensional A conținând Q numere întregi. Valoarea lui A_i ($0 \leq i \leq Q-1$) trebuie să fie `1` dacă călătoria este posibilă satisfăcând constrângerile menționate anterior, sau `0` în caz contrar.

Exemplu

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

Grader-ul apelează `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])`.



Pentru călătoria 0, puteți traversa din orașul 4 către orașul 2 în felul următor:

- Porniți din orașul 4 (aveți forma unui om)
- Mergeți către orașul 3 (aveți forma unui om)
- Mergeți către orașul 1 (aveți forma unui om)
- Transformați-vă în forma unui lup (aveți forma unui lup)
- Mergeți către orașul 2 (aveți forma unui lup)

Pentru călătoriile 1 și 2, nu puteți traversa între orașele indicate.

Prin urmare, programul dumneavoastră va întoarce $[1, 0, 0]$.

Fișierele `sample-01-in.txt` și `sample-01-out.txt` din pachetul anexat sub forma de arhivă corespund acestui exemplu. Un alt exemplu intrare/ieșire, este de asemenea disponibil în pachet.

Constrângeri

- $2 \leq N \leq 200\,000$
- $N - 1 \leq M \leq 400\,000$
- $1 \leq Q \leq 200\,000$
- Pentru fiecare $0 \leq j \leq M - 1$
  - $0 \leq X_j \leq N - 1$
  - $0 \leq Y_j \leq N - 1$
  - $X_j \neq Y_j$
- Puteți călători între oricare două orașe distincte utilizând drumurile.
- Fiecare pereche de orașe este direct conectată prin cel mult un drum. În alte cuvinte, oricare ar fi $0 \leq j < k \leq M - 1$, $(X_j, Y_j) \neq (X_k, Y_k)$ și $(Y_j, X_j) \neq (X_k, Y_k)$.
- Pentru fiecare $0 \leq i \leq Q - 1$
  - $0 \leq L_i \leq S_i \leq N - 1$
  - $0 \leq E_i \leq R_i \leq N - 1$
  - $S_i \neq E_i$
  - $L_i \leq R_i$

Subtask-uri

1. (7 puncte) $N \leq 100$, $M \leq 200$, $Q \leq 100$
2. (8 puncte) $N \leq 3\,000$, $M \leq 6\,000$, $Q \leq 3\,000$
3. (34 puncte) $M = N - 1$ și orice oraș este incident cu cel mult 2 drumuri (orașele sunt conectate într-o linie)
4. (51 puncte) Fără constrângeri adiționale

Grader-ul local

Grader-ul local citește datele de intrare în următoarea formă:

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

Grader-ul local afișează valoarea întoarsă de `check_validity` în următoarea formă:

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