Weerwolf

Er zijn N steden en M wegen in Ibaraki Prefecture, Japan. De steden zijn genummerd van 0 tot en met $N-1$ in oplopende volgorde van inwoneraantal. Elke weg verbindt een paar van verschillende steden en kan in beide richtingen gebruikt worden. Je kunt van elke stad naar elke andere stad reizen over een of meer wegen.

Je hebt Q trips gepland, genummerd van 0 tot en met $Q-1$. Trip i ($0 \leq i \leq Q-1$) is om van stad S_i naar stad E_i te reizen.

Je bent een weerwolf. Je hebt twee gedaanten: **mens** en **wolf**. Aan het begin van elke trip ben je in de menselijke gedaante. Aan het einde van iedere trip moet je wolf zijn. Gedurende elke trip moet je precies één keer **transformeren** (gedaante verwisselen van mens naar wolf) en dit kan alleen maar in een stad gebeuren (mogelijk S_i of E_i).

Het leven van een weerwolf is niet makkelijk. Je moet steden met weinig inwoners vermijden als je mens bent en steden met veel inwoners vermijden als je wolf bent.

Voor elke trip i ($0 \leq i \leq Q-1$) zijn er twee drempels L_i en R_i met $0 \leq L_i \leq R_i \leq N-1$ die aangeven welke steden je moet vermijden. Steden $0, 1, \ldots, L_i-1$ moet je vermijden als je mens bent, en de steden $R_i+1, R_i+2, \ldots, N-1$ moet je vermijden als je wolf bent. Dit betekent dus dat je in trip i alleen kunt transformeren in een van de steden $L_i, L_i+1, \ldots, R_i$.

Bepaal voor elke trip of het mogelijk is om te reizen van stad S_i naar stad E_i, op een wijze die voldoet aan de gegeven voorwaarden. De lengte van de route maakt niet uit.

Implementatiedetails

Implementeer de volgende functie:

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

- `N`: het aantal steden.
- `X` en `Y`: arrays van lengte M. Voor elke j ($0 \leq j \leq M-1$), stad `X[j]` is middels een weg rechtsteeks verbonden met stad `Y[j]`.
- `S`, `E`, `L`, en `R`: arrays van lengte Q, die de trips weergeven.

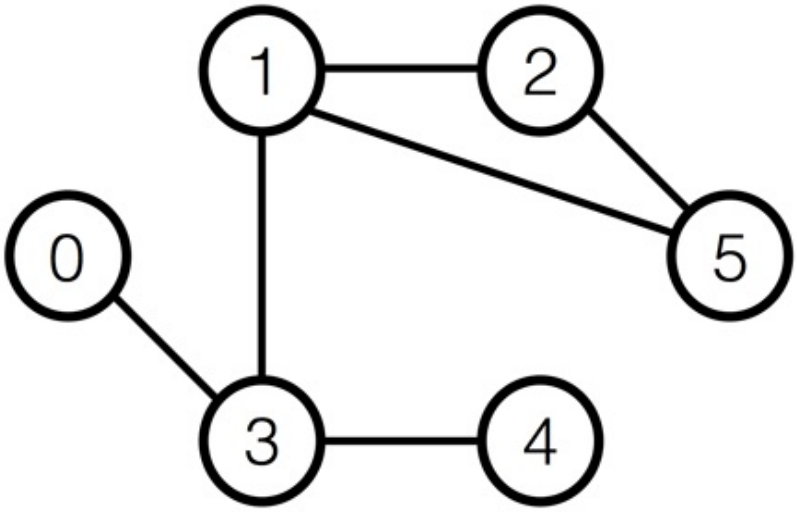
Merk op dat de waardes van M en Q de lengtes van de arrays aangeven en bepaald

kunnen worden op de wijze die staat aangegeven in de implementatieopmerkingen. De functie `check_validity` wordt voor elke test case precies een keer aangeroepen . Deze functie moet een array A opleveren van lengte Q, dat bestaat uit integers. De waarde van A_i $(0 \leq i \leq Q-1)$ moet `1` zijn als trip i mogelijk is onder bovenstaande voorwaarden en anders `0`.

Voorbeeld

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

De grader doet de volgende aanroep `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])`.



Bij trip 0 kun je als volgt reizen van stad 4 naar stad 2:

- Begin in stad 4 (als mens)
- Reis naar stad 3 (als mens)
- Reis naar stad 1 (als mens)
- Verander in een weerwolf (als wolf)
- Reis naar stad 2 (als wolf)

Voor reizen 1 en 2 is het niet mogelijk te reizen tussen de gegeven steden.

Je programma moet dus als oplossing retourneren: $[1, 0, 0]$.

De bestanden `sample-01-in.txt` en `sample-01-out.txt` in de ZIP-file horen bij dit voorbeeld. Je vindt ook een ander paar voorbeelden van invoer en uitvoer in de ZIP-file.

Randvoorwaarden

- $2 \leq N \leq 200\,000$
- $N - 1 \leq M \leq 400\,000$

- $1 \leq Q \leq 200\,000$
- Voor elke $0 \leq j \leq M - 1$
  - $0 \leq X_j \leq N - 1$
  - $0 \leq Y_j \leq N - 1$
  - $X_j \neq Y_j$
- Je kan reizen van elke stad naar elke andere stad door wegen te gebruiken.
- Elk paar steden is direct verbonden met maximaal één weg. In andere woorden, voor elke $0 \leq j < k \leq M - 1$, $(X_j, Y_j) \neq (X_k, Y_k)$ en $(Y_j, X_j) \neq (X_k, Y_k)$.
- Voor elke $0 \leq i \leq Q - 1$
  - $0 \leq L_i \leq S_i \leq N - 1$
  - $0 \leq E_i \leq R_i \leq N - 1$
  - $S_i \neq E_i$
  - $L_i \leq R_i$

Subtaken

1. (7 punten) $N \leq 100$, $M \leq 200$, $Q \leq 100$
2. (8 punten) $N \leq 3\,000$, $M \leq 6\,000$, $Q \leq 3\,000$
3. (34 points) $M = N - 1$ en elke stad is verbonden met hoogstens 2 wegen (de steden liggen op een lijn).
4. (51 punten) Geen aanvullende voorwaarden

Voorbeeldgrader

De voorbeeldgrader leest de invoer in het volgende formaat:

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

De voorbeeldgrader drukt de uitvoer van `check_validity` af in het volgende formaat:

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