Werwolf

In Japans Präfektur Ibaraki gibt es N Städte und M Strassen. Die Städte sind aufsteigend nach Einwohnerzahl mit den Zahlen 0 bis $N-1$ durchnummeriert. Jede Strasse verbindet zwei verschiedene Städte und kann in beide Richtungen genutzt werden. Mittels (möglicherweise mehrerer) dieser Strassen kann man von jeder Stadt aus zu jeder anderen Stadt gelangen.

Du hast Q Reisen geplant, welche von 0 bis $Q-1$ durchnummeriert sind. Dabei führt die i-te Reise ($0 \leq i \leq Q-1$) von Stadt S_i zu Stadt E_i.

Du bist ein Werwolf. Du hast zwei Gestalten: die **Menschengestalt** und die **Wolfsgestalt**. Zu Beginn jeder Reise bist du in Menschengestalt. Am Ende jeder Reise musst du in Wolfsgestalt sein. Während jeder Reise musst du dich genau einmal **verwandeln** (von Menschen- in Wolfsgestalt). Du kannst dich nur verwandeln, wenn du dich in einer Stadt (möglicherweise S_i oder E_i) aufhältst.

Das Leben als Werwolf ist nicht einfach. In Menschengestalt musst du schwachbevölkerte und in Wolfsgestalt vielbevölkerte Städte meiden. Für jede Reise i existieren Grenzwerte L_i und R_i mit $0 \leq L_i \leq R_i \leq N-1$. Dies bedeutet, dass du auf der i-ten Reise die Städte $0, 1, \ldots, L_i - 1$ meiden musst, solange du in Menschengestalt bist, und die Städte $R_i + 1, R_i + 2, \ldots, N-1$ meiden musst, sobald du in Wolfsgestalt bist. Auf Reise i kannst du dich also nur in einer der Städte $L_i, L_i + 1, \ldots, R_i$ verwandeln.

Deine Aufgabe ist es, für jede Reise zu ermitteln, ob du unter Einhaltung obiger Bedingungen von Stadt S_i zu Stadt E_i gelangen kannst. Dabei darf deine Reise beliebig lang sein.

Implementierungshinweise

Implementiere die folgende Funktion:

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

- `N`: Die Anzahl der Städte
- `X` und `Y`: Arrays der Länge M. Für jedes j ($0 \leq j \leq M-1$) verbindet eine Strasse die Stadt `X[j]` direkt mit Stadt `Y[j]`.

- `S`, `E`, `L` und `R`: Arrays der Länge Q, welche die Reisen darstellen.

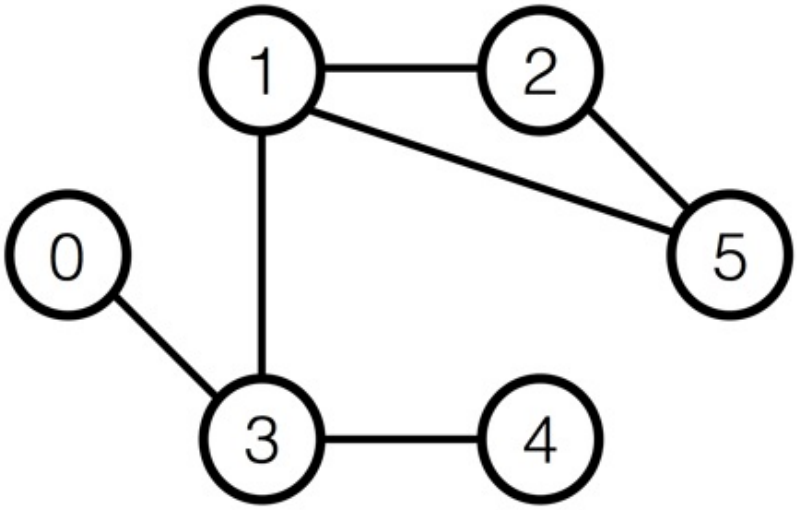
Beachte, dass die Werte für M und Q die Längen der entsprechenden Arrays sind und folglich wie in den Implementierungshinweisen beschrieben erhalten werden können.

Die Funktion `check_validity` wird pro Testfall genau einmal aufgerufen. Diese soll ein Array A der Länge Q von Ganzzahlen zurückgeben. Der Wert von A_i ($0 \leq i \leq Q-1$) muss dabei `1` lauten, wenn Reise i unter obigen Bedingungen möglich ist, sonst `0`.

Beispiel

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

Der Grader macht einen Aufruf `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])`.



Auf Reise 0 kannst du von Stadt 4 wie folgt zu Stadt 2 gelangen:

- Beginne in Stadt 4 (Du bist in Menschengestalt)
- Reise zu Stadt 3 (Du bist in Menschengestalt)
- Reise zu Stadt 1 (Du bist in Menschengestalt)
- Verwandle dich in eine Wolfsgestalt (Du bist in Wolfsgestalt)
- Reise zu Stadt 2 (Du bist in Wolfsgestalt)

Bei den Reisen 1 und 2 ist es nicht möglich, von Start zu Ziel zu gelangen.

Demnach sollte dein Programm $[1, 0, 0]$ zurückgeben.

Die Dateien `sample-01-in.txt` und `sample-01-out.txt` im Zip-Archiv unter *Attachments* entsprechen diesem Beispiel. Dieses Archiv enthält auch ein weiteres Ein-/Ausgabepaar.

Limits

- $2 \leq N \leq 200\,000$
- $N - 1 \leq M \leq 400\,000$
- $1 \leq Q \leq 200\,000$
- Für jedes $0 \leq j \leq M - 1$ gilt:
  - $0 \leq X_j \leq N - 1$
  - $0 \leq Y_j \leq N - 1$
  - $X_j \neq Y_j$
- Man kann entlang der Strassen von jeder Stadt zu jeder anderen Stadt gelangen.
- Zwischen je zwei Städten verläuft höchstens eine direkte Strasse. Mit anderen Worten: für alle $0 \leq j < k \leq M - 1$ gilt $(X_j, Y_j) \neq (X_k, Y_k)$ und $(Y_j, X_j) \neq (X_k, Y_k)$.
- Für alle $0 \leq i \leq Q - 1$ gilt:
  - $0 \leq L_i \leq S_i \leq N - 1$
  - $0 \leq E_i \leq R_i \leq N - 1$
  - $S_i \neq E_i$
  - $L_i \leq R_i$

Teilaufgaben

1. (7 Punkte) $N \leq 100$, $M \leq 200$, $Q \leq 100$
2. (8 Punkte) $N \leq 3\,000$, $M \leq 6\,000$, $Q \leq 3\,000$
3. (34 Punkte) $M = N - 1$ und jede Stadt ist mit höchstens zwei anderen Städten direkt verbunden (die Städte sind wie auf einer Perlenschnur aufgereiht).
4. (51 Punkte) Keine weiteren Einschränkungen.

Beispielgrader

Der Beispielgrader liest Eingaben in folgendem Format:

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

Der Beispielgrader gibt den Rückgabewert von `check_validity` in folgendem Format aus:

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