المستذئب

يوجد N مدينة و M طريق في ولاية ابرقي، اليابان. المدن مرقمة من 0 إلى $N-1$ تصاعدياً بحسب تعدادهم السكاني. كل طريق يصل بين اثنين من المدن المختلفة, ويمكن عبوره بالاتجاهين. يمكنك العبور من أي مدينة إلى أي مدينة باستخدام طريق واحد أو أكثر من هذه الطرق. لقد قمت بالتخطيط لQ رحلة ، مرقمة من 0 إلى $Q-1$. الرحلة رقم i ($0 \leq i \leq Q-1$) هي العبور من المدينة S_i إلى المدينة E_i.

لقد كنت مستذئباً. لديك شكلين: **الشكل الانساني** و **والشكل المستذئب**. في بداية كل رحلة انت تكون بالشكل الانساني. في نهاية كل رحلة يجب ان تكون بالشكل المستذئب. خلال الرحلة يجب عليك **التحول** (التبدل من الشكل الانساني إلى الشكل المستذئب) مرة واحدة تماماً. يمكنك التحول فقط عندما تكون في أحد المدن (ربما S_i أو E_i). الحياة كمستذئب ليست بالسهلة. يجب عليك تجنب المدن قليلة التعداد السكاني عندما تكون بالشكل الانساني, وتجنب المدن كثيرة التعداد السكاني عندما تكون بالشكل المستذئب.

من اجل كل رحلة i ($0 \leq i \leq Q-1$), يوجد رقمان L_i وR_i ($0 \leq L_i \leq R_i \leq N-1$) يعبران عن المدن التي يجب تجنبها. بشكل أدق, يجب عليك تجنب المدن $0, 1, \ldots, L_i-1$ عندما تكون بالشكل الانساني, ويجب عليك تجنب المدن $R_i+1, R_i+2, \ldots, N-1$ عندما تكون بالشكل المستذئب. هذا يعني في الرحلة i, يمكنك التحول فقط في المدن $L_i, L_i+1, \ldots, R_i$.

المطلوب هو ان تحدد من أجل كل رحلة فيما اذا كان ممكنا العبور من المدينة S_i إلى المدينة E_i بحيث تحقق الشروط المذكورة سابقاً. الطريق المستخدم يمكن ان يكون بأي طول ممكن.

تفاصيل البرمجة

يجب عليك برمجة التابع التالي:

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

- N: عدد المدن.
- X و Y: مصفوفتان بطول M. لأجل كل j ($0 \leq j \leq M-1$), المدينة X[j] موصولة بشكل مباشر إلى المدينة Y[j] عبر طريق.
- S و E و L و R: مصفوفات بطول Q تمثل الرحلات.

لاحظ ان M وQ هما اطوال المصفوفات , ويمكن استخراجهم بحسب الملاحظات البرمجية

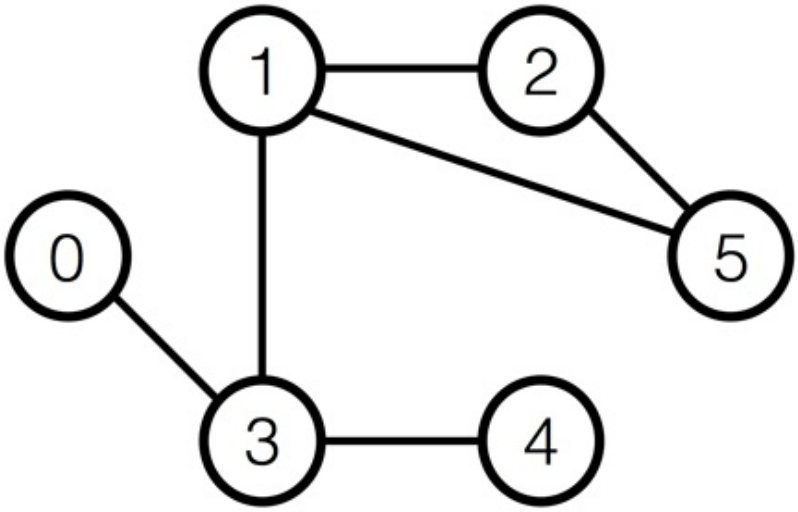
التابع check_validity يتم استدعاؤه مرة واحدة تماما لأجل كل حالة اختبار. يجب على هذا التابع أن يعيد مصفوفة A من الاعداد الصحيحة بطول Q. قيمة A_i ($0 \leq i \leq Q-1$) يجب ان تكون 1 اذا كانت الرحلة i ممكنة بحسب الشروط المذكورة سابقاً, او 0 بعكس ذلك.

مثال

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

المقيم سيستدعي

```
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])
```



من اجل الرحلة 0, يمكنك العبور من المدينة 4 إلى المدينة 2 كالتالي:

- ابدأ في المدينة 4 (انت في الشكل الانساني)
- تحرك إلى المدينة 3 (انت في الشكل الانساني)
- تحرك إلى المدينة 1 (انت في الشكل الانساني)
- حول نفسك إلى الشكل المستذئب (انت في الشكل المستذئب)
- تحرك إلى المدينة 2 (انت في الشكل المستذئب)

من اجل الرحلتين 1 و 2 لايمكنك العبور بين المدينتين المعطاتين.

لذلك برنامجك عليه اعادة $[1, 0, 0]$.

الملفات `sample-01-in.txt` و `sample-01-out.txt` في الحزمة المضغوطة المرفقة تحوي هذا المثال. هذه الحزمة تحوي مثالاً اخر من ملفات الدخل والخرج

القيود

- $2 \leq N \leq 200\,000$
- $N - 1 \leq M \leq 400\,000$
- $1 \leq Q \leq 200\,000$
- لأجل كل $0 \leq j \leq M - 1$
  - $0 \leq X_j \leq N - 1$

    - $0 \leq Y_j \leq N - 1$
    - $X_j \neq Y_j$
- يمكنك العبور من أي مدينة إلى أي مدينة باستخدام الطرق الموجودة.
- كل زوج من المدن هما متصلان عن طريق طريق مباشر واحد كحد اقصى. بعبارة أخرى، من اجل كل $0 \leq j < k \leq M - 1$ يكون $(X_j, Y_j) \neq (X_k, Y_k)$ و $(Y_j, X_j) \neq (X_k, Y_k)$.
- من اجل كل $0 \leq i \leq Q - 1$
    - $0 \leq L_i \leq S_i \leq N - 1$
    - $0 \leq E_i \leq R_i \leq N - 1$
    - $S_i \neq E_i$
    - $L_i \leq R_i$

المسائل الجزئية

1. (points 7) $N \leq 100$, $M \leq 200$, $Q \leq 100$
2. (points 8) $N \leq 3\,000$, $M \leq 6\,000$, $Q \leq 3\,000$
3. (points 34) $M = N - 1$ وكل مدينة هي مجاورة لـ 2 من الطرق كحد اقصى (المدن متصلة على شكل خط)
4. (points 51) لايوجد قيود اضافية

مقيم الاختبار

The sample grader reads the input in the following format:

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

The sample grader prints the return value of `check_validity` in the following format:

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