گرگ‌نما

در استان ایباراکی ژاپن، N شهر و M جاده وجود دارد. شهرها به ترتیب صعودی جمعیتشان از 0 تا $N-1$ شماره‌گذاری شده‌اند. هر جاده دو شهر متمایز را به هم متصل می‌کند که از هر دو طرف قابل استفاده است. شما می‌توانید از طریق این جاده‌ها از هر شهر به هر شهر دیگر سفر کنید.

شما Q سفر برنامه‌ریزی کرده‌اید که از 0 تا $Q-1$ شماره‌گذاری شده‌اند. سفر iاُم ($0 \leq i \leq Q-1$) از شهر S_i به شهر E_i است.

شما یک گرگ‌نما هستید که دو چهره دارید: **چهره‌ی انسانی** و **چهره‌ی گرگی**. در ابتدای هر سفر شما در چهره‌ی انسانی خود هستید. در پایان هر سفر، شما باید در چهره‌ی گرگی باشید. در طول سفر شما باید دقیقا یک بار در یکی از شهرها **تغییر چهره** دهید (یعنی از چهره‌ی انسانی به چهره‌ی گرگی درآیید). این تغییر چهره می‌تواند در S_i یا E_i نیز رخ دهد.

زندگی در چهره‌ی گرگی چندان آسان نیست. شما باید در چهره‌ی انسانی خود از شهرهای کم‌جمعیت اجتناب کنید. همچنین وقتی در چهره‌ی گرگی هستید، باید از شهرهای پرجمعیت اجتناب کنید. برای هر سفر i دو عدد آستانه‌ی L_i و R_i با شرط $0 \leq L_i \leq R_i \leq N-1$ وجود دارد که شهرهایی که باید اجتناب شوند را مشخص می‌کنند. به طور مشخص، شما باید در چهره‌ی انسانی از شهرهای $0, 1, \ldots, L_i-1$ و در چهره‌ی گرگی از شهرهای $R_i+1, R_i+2, \ldots, N-1$ اجتناب کنید. این بدین معنا است که در سفر i شما باید دقیقاً در یکی از شهرهای $L_i, L_i+1, \ldots, R_i$ تغییر چهره دهید.

برای هر سفر، شما باید تعیین کنید که آیا مسیری از شهر S_i به E_i وجود دارد که شرایط فوق را ارضا کند یا خیر. مسیری که انتخاب می‌کنید می‌تواند هر طول دلخواهی داشته باشد.

جزئیات پیاده‌سازی

شما باید تابع زیر را پیاده‌سازی کنید:

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

- N: تعداد شهرها.
- X و Y: آرایه‌هایی به طول M. برای هر j ($0 \leq j \leq M-1$)، شهر X[j] با یک جاده به شهر Y[j] متصل شده است.
- L، E، S و R: آرایه‌هایی به طول Q که سفرها را مشخص می‌کنند. توجه داشته باشید که مقادیر M و Q طول آرایه‌ها هستند و می‌توان آن‌ها را همان‌طور که در نکات پیاده‌سازی ذکر شده به دست آورد.

تابع check_validity دقیقاً یک بار برای هر مورد آزمون فراخوانی می‌شود. این تابع باید یک آرایه‌ی A به طول Q از اعداد صحیح برگرداند. مقدار A_i ($0 \leq i \leq Q-1$) باید برابر 1 باشد اگر سفر i با شرایط مذکور ممکن باشد،

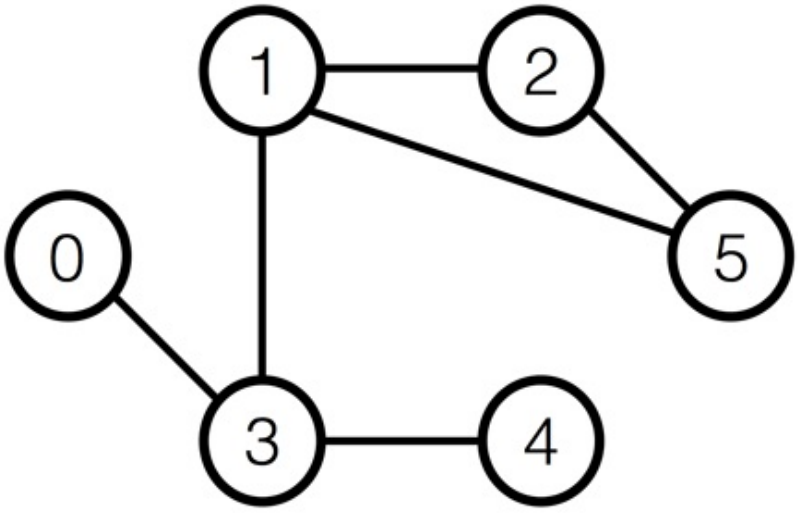
در غیر این صورت مقدار A_i باید برابر 0 باشد.

مثال

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

ارزیاب تابع را به صورت زیر فراخوانی می‌کند:

```
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]).
```



برای سفر 0، شما می‌توانید از شهر 4 به شهر 2 به شکل زیر حرکت کنید:

- از شهر 4 شروع کنید (در چهره‌ی انسانی)
- به شهر 3 حرکت کنید (در چهره‌ی انسانی)
- به شهر 1 حرکت کنید (در چهره‌ی انسانی)
- به چهره‌ی گرگی تغییر چهره دهید.
- به شهر 2 حرکت کنید (در چهره‌ی گرگی)

برای سفرهای 1 و 2 شما نمی‌توانید بین دو شهر داده‌شده سفر کنید.

بنابراین، برنامه باید مقدار $[1, 0, 0]$ را برگرداند.

فایل‌های `sample-01-in.txt` و `sample-01-out.txt` در بسته‌ی فشرده‌ی پیوست مربوط با این مثال هستند. این بسته شامل یک جفت فایل ورودی/خروجی نمونه‌ی دیگر نیز هست.

محدودیت‌ها

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

- به ازای هر $0 \le j \le M - 1$
  - $0 \le X_j \le N - 1$
  - $0 \le Y_j \le N - 1$
  - $X_j \ne Y_j$
- شما با استفاده از جاده‌ها می‌توانید از هر شهر به هر شهر دیگر سفر کنید.
- بین هر دو شهر حداکثر یک جاده وجود دارد. به عبارت دیگر، به ازای هر $0 \le j < k \le M - 1$، $(X_j, Y_j) \ne (X_k, Y_k)$ و $(Y_j, X_j) \ne (X_k, Y_k)$.
- به ازای هر $0 \le i \le Q - 1$
  - $0 \le L_i \le S_i \le N - 1$
  - $0 \le E_i \le R_i \le N - 1$
  - $S_i \ne E_i$
  - $L_i \le R_i$

زیرمسئله‌ها

1. (۷ نمره) $N \le 100$، $M \le 200$، $Q \le 100$
2. (۸ نمره) $N \le 3\,000$، $M \le 6\,000$، $Q \le 3\,000$
3. (۳۴ نمره) $M = N - 1$ و هر شهر حداکثر به دو جاده متصل است (یعنی شهرها در یک خط به هم وصل هستند)
4. (۵۱ نمره) بدون محدودیت اضافی

ارزیاب نمونه

ارزیاب نمونه ورودی را در قالب زیر می‌خواند:

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

ارزیاب نمونه خروجی `check_validity` را در قالب زیر برمی‌گرداند.

- خط $1 + i$ $(0 \le i \le Q - 1)$: A_i