عوارض بزرگراه

در ژاپن شهرها توسط شبکه‌ای از بزرگراه‌ها به هم متصل شده‌اند. این شبکه شامل N شهر و M بزرگراه است. هر بزرگراه دو شهر متمایز را به هم وصل می‌کند و بین هر دو شهر حداکثر یک بزرگراه وجود دارد. شهرها از 0 تا $N-1$ و بزرگراه‌ها از 0 تا $M-1$ شماره‌گذاری شده‌اند. شما می‌توانید در هر بزرگراه در هر دو جهت حرکت کنید. همچنین از طریق از بزرگراه‌ها می‌توان از هر شهر به هر شهر دیگری رسید.

عبور از هر بزرگراه مشمول پرداخت عوارض است. عوارض یک بزرگراه بستگی به شرایط ترافیکی آن بزرگراه دارد. ترافیک یا **سبک** است یا **سنگین**. وقتی ترافیک سبک است، عوارض بزرگراه A پِن (واحد پول ژاپن) است و وقتی ترافیک سنگین است، عوارض B ین است. ما مقادیر A و B را می‌دانیم .همچنین تضمین شده است که $A < B$.

شما یک دستگاه دارید که با داشتن شرایط ترافیکی بزرگراه‌ها، کم‌ترین مقدار عوارضی که برای سفر از شهر S به شهر T ($S \neq T$) تحت شرایط ترافیکی داده‌شده باید پرداخت شود را محاسبه می‌کند.

با این حال، دستگاه فقط یک نمونه‌ی اولیه است. مقادیر S و T ثابت هستند (یعنی داخل دستگاه، هاردکد شده‌اند) و شما مقدار آن‌ها را نمی‌دانید. شما قرار است مقادیر S و T را به دست آورید. برای این کار، شما قصد دارید چند شرایط ترافیکی را برای دستگاه مشخص کنید و از مقدار عوارضی که دستگاه برمی‌گرداند، مقادیر S و T را نتیجه بگیرید. با توجه به این که مشخص‌کردن شرایط ترافیکی هزینه‌بر است، شما نمی‌خواهید از دستگاه به دفعات زیاد استفاده کنید.

جزئیات پیاده‌سازی

شما باید رویه‌ی زیر را پیاده‌سازی کنید:

```
find_pair(int N, int[] U, int[] V, int A, int B)
```

- `N`: تعداد شهرها.
- `U` و `V`: آرایه‌هایی به طول M، که M تعداد بزرگراه‌ها است. به ازای هر i ($0 \leq i \leq M-1$)، بزرگراه i دو شهر `U[i]` و `V[i]` را به هم متصل می‌کند.
- `A`: عوارض هر بزرگراه وقتی ترافیک سبک است.
- `B`: عوارض هر بزرگراه وقتی ترافیک سنگین است.
- این تابع دقیقاً یک بار به ازای هر مورد نمونه فراخوانی می‌شود.
- دقت کنید که مقدار M اندازه‌ی آرایه‌ها است و می‌توان آن را طبق آن‌چه در نکات پیاده‌سازی توضیح داده شده به دست آورد.

رویه‌ی `find_pair` می‌تواند تابع زیر را فراخوانی کند:

```
int64 ask(int[] w)
```

- طول w باید M باشد. آرایه‌ی w شرایط ترافیکی را مشخص می‌کند.
- به ازای هر i ($0 \leq i \leq M-1$)، w[i] شرایط ترافیکی در بزرگراه i را مشخص می‌کند.
  - w[i] = 0 به این معنی است که ترافیک در بزرگراه i سبک است.
  - w[i] = 1 به این معنی است که ترافیک در بزرگراه i سنگین است.
- این تابع، کم‌ترین مجموع عوارض ممکن که برای سفر بین شهرهای S و T تحت شرایط ترافیکی w باید پرداخت شود را برمی‌گرداند.
- این تابع را می‌توان حداکثر ۱۰۰ مرتبه برای هر مورد آزمون فراخوانی کرد.

برای گزارش پاسخ، find_pair باید رویه‌ی زیر را فراخوانی کند:

```
answer(int s, int t)
```

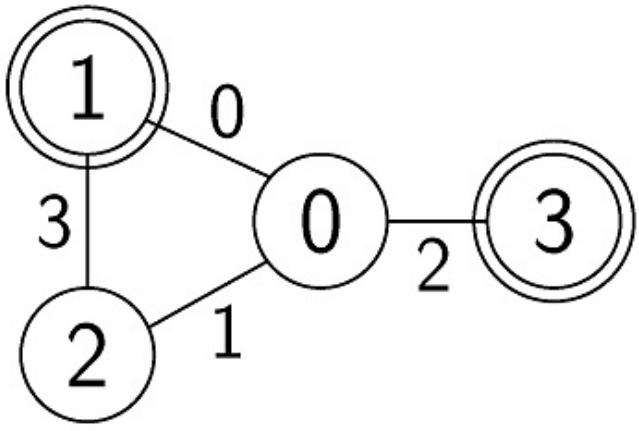
- s و t باید برابر مقادیر زوج S و T باشند (ترتیب اهمیتی ندارد).
- این رویه باید دقیقاً یک بار فراخوانی شود.

اگر برخی از شرایط فوق برآورده نشود، برنامه‌ی شما به صورت **Wrong Answer** داوری خواهد شد. در غیر این صورت برنامه به صورت **Accepted** داوری شده و امتیاز آن بر اساس تعداد فراخوانی‌های ask محاسبه می‌شود (بخش زیرمسئله‌ها را ببینید).

مثال

فرض کنید $N = 4$، $M = 4$، $U = [0, 0, 0, 1]$، $V = [1, 2, 3, 2]$، $A = 1$، $B = 3$، $S = 1$ و $T = 3$.

ارزیاب find_pair(4, [0, 0, 0, 1], [1, 2, 3, 2], 1, 3) را فراخوانی می‌کند.



در شکل فوق، یال شماره‌ی i متناظر با بزرگراه شماره‌ی i است. چند فراخوانی ممکن به تابع ask و مقادیر برگشتی متناظر آن در زیر آمده‌اند:

فراخوانی	مقدار برگشتی
`ask([0, 0, 0, 0])`	2
`ask([0, 1, 1, 0])`	4
`ask([1, 0, 1, 0])`	5
`ask([1, 1, 1, 1])`	6

برای فراخوانی تابع `ask([0, 0, 0, 0])`، ترافیک در تمام بزرگراه‌ها سبک و عوارض هر بزرگراه برابر 1 است. ارزان‌ترین مسیر از $S = 1$ به $T = 3$ برابر است با $3 \to 0 \to 1$. مجموع عوارض برای این مسیر برابر 2 است. بنابراین تابع مقدار 2 را برمی‌گرداند.

برای گزارش صحیح پاسخ، رویه‌ی `find_pair` باید `answer(1, 3)` یا `answer(3, 1)` را فراخوانی کند.

فایل `sample-01-in.txt` در بسته‌ی فشرده‌ی پیوست مربوط به این مثال است. نمونه‌های ورودی دیگری نیز در این بسته قرار دارد.

محدودیت‌ها

- $2 \le N \le 90\,000$
- $1 \le M \le 130\,000$
- $1 \le A < B \le 1\,000\,000\,000$
- به ازای هر $0 \le i \le M - 1$
  - $0 \le U[i] \le N - 1$
  - $0 \le V[i] \le N - 1$
  - $U[i] \ne V[i]$
- $(U[i], V[i]) \ne (U[j], V[j])$ و $(U[i], V[i]) \ne (V[j], U[j])$ $(0 \le i < j \le M - 1)$
- شما می‌توانید از طریق از بزرگراه‌ها از هر شهر به هر شهر دیگر سفر کنید.
- $0 \le S \le N - 1$
- $0 \le T \le N - 1$
- $S \ne T$

در این مسئله، ارزیاب تطابقی نیست. این بدان معنی است که S و T در ابتدای اجرای ارزیاب ثابت هستند و بر اساس فراخوانی‌های تابع `ask` توسط راه‌حل شما تغییر نمی‌کنند.

زیرمسئله‌ها

1. (۵ نمره) یکی از S و T برابر 0 است، $N \le 100$، $M = N - 1$
2. (۷ نمره) یکی از S و T برابر 0 است، $M = N - 1$
3. (۶ نمره) $M = N - 1$، $U[i] = i$، $V[i] = i + 1$ $(0 \le i \le M - 1)$
4. (۳۳ نمره) $M = N - 1$
5. (۱۸ نمره) $A = 1$، $B = 2$
6. (۳۱ نمره) بدون محدودیت اضافی

فرض کنید برنامه‌ی شما به صورت **Accepted** داوری شده و X فراخوانی به تابع ask داشته است. در این صورت امتیاز شما P برای نمونه‌ی ورودی، بسته به شماره‌ی زیرمسئله به صورت زیر محاسبه می‌شود:

- زیرمسئله‌ی ۱. $P = 5$.
- زیرمسئله‌ی ۲. اگر $X \leq 60$، $P = 7$. در غیر این صورت $P = 0$.
- زیرمسئله‌ی ۳. اگر $X \leq 60$، $P = 6$. در غیر این صورت $P = 0$.
- زیرمسئله‌ی ۴. اگر $X \leq 60$، $P = 33$. در غیر این صورت $P = 0$.
- زیرمسئله‌ی ۵. اگر $X \leq 52$، $P = 18$. در غیر این صورت $P = 0$.
- زیرمسئله‌ی ۶.
  - اگر $X \leq 50$، $P = 31$.
  - اگر $51 \leq X \leq 52$، $P = 21$.
  - اگر $53 \leq X$، $P = 0$.

توجه داشته باشید که امتیاز شما برای هر زیرمسئله برابر کم‌ترین امتیازی است که برای موارد آزمون در آن زیرمسئله دریافت کرده‌اید.

ارزیاب نمونه

ارزیاب نمونه ورودی را در قالب زیر می‌خواند:

- خط 1: $T\ S\ B\ A\ M\ N$
- خط $2 + i$ $(0 \leq i \leq M - 1)$: $V[i]\ U[i]$

اگر برنامه‌ی شما به صورت **Accepted** داوری شود، ارزیاب نمونه پیام `Accepted: q` را چاپ می‌کند که q تعداد فراخوانی‌ها به تابع ask است.

اگر برنامه‌ی شما به صورت **Wrong Answer** داوری شود، ارزیاب پیام `Wrong Answer: MSG` را چاپ می‌کند که MSG یکی از موارد زیر است:

- `answered not exactly once`: رویه‌ی answer دقیقاً یک بار فراخوانی نشده است.
- `w is invalid`: طول w که به تابع ask داده شده برابر M نیست، یا به ازای یک i $(0 \leq i \leq M - 1)$ w[i] نه 0 است و نه 1.
- `more than 100 calls to ask`: تابع ask بیش از ۱۰۰ بار فراخوانی شده است.
- `{s, t} is wrong`: رویه‌ی answer با مقادیر نادرست s و t فراخوانی شده است.