# رسوم الطرق السريعة

في اليابان, المدن متصلة ببعضها عبر شبكة من الطرق السريعة. هذه الشبكة تتألف من $N$ مدينة و $M$ طريق سريع. كل طريق سريع يصل بين زوج من المدن. لا يوجد طريقين سريعين يصلان بين نفس الزوج من المدن. المدن مرقمة من $0$ إلى $N-1$, والطرق السريعة مرقمة من $0$ إلى $M-1$. يمكنك القيادة على أي طريق سريع بالاتجاهين. يمكنك الانتقال من أي مدينة إلى أي مدينة عبر الطرق السريعة.

يتم فرض رسوم لأجل القيادة على الطريق السريع. رسم الطريق السريع يعتمد على حالة **حركة المرور** على الطريق السريع. حركة المرور هي إما **خفيفة** or **مزدحمة**. عندما تكون حركة المرور خفيفة, يكون الرسم هو $A$ ين (العملة اليابانية). عندما تكون حركة المرور مزدحمة, الرسم هو $B$ ين. من المضمون أن $A < B$. لاحظ انك تعلم قيم $A$ و $B$.

انت لديك آلة اذا اعطيتها حالات حركات المرور على جميع الطرق السريعة, ستقوم بحساب اصغر مجموع كلي للرسوم بين المدينتين $S$ و $T$ ($S \neq T$), ضمن حالات حركات المرور المحددة.

ولكن، الآلة ما تزال في النموذج الاولي. قيم $S$ و $T$ هي مثبتة (أي موجودة ضمن الآلة) وهي غير معروفة من قبلك. انت تريد تحديد قيم $S$ و $T$. لتقوم بذلك, انت تخطط لتحديد بعض حالات حركة المرور إلى الآلة, واستخدام الرسوم التي تخرجها الآلة لكي تستنتج قيم $S$ and $T$. بما أن تحديد حالات حركة المرور هو أمر مكلف, أنت لا تريد استخدام هذه الألة الكثير من المرات.

## تفاصيل البرمجة

يجب عليك القيام ببرمجة الاجراء التالي:

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

- N: عدد المدن.
- U و V: مصفوفتان بطول $M$, حيث $M$ هو عدد الطرق السريعة التي تصل المدن. من اجل كل $i$ ($0 \leq i \leq M-1$), الطريق السريع $i$ يصل بين U[i] و V[i].
- A: رسوم الطريق السريع عندما تكون الحركة المرورية منخفضة.
- B: رسوم الطريق السريع عندما تكون الحركة المرورية مزدحمة.
- هذا الاجراء يتم استدعاؤه مرة واحدة فقط لكل حالة اختبار.
- لاحظ ان $M$ هو طول المصفوفتين, ويمكن الحصول عليه كما هو مذكور في ملاحظة البرمجة.

الاجراء find_pair يمكنه استدعاء التابع:

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

- طول w يجب أن يكون $M$. المصفوفة w تشرح حالات الحركة المرورية.
- لأجل كل $i$ ($0 \leq i \leq M-1$), w[i] تحدد حالة الحركة المرورية على الطريق السريع $i$. قيمة w[i] يجب ان تكون $0$ أو $1$.
  - w[i] = 0 تعني أن الحركة المرورية على الطريق السريع $i$ هي خفيفة.
  - w[i] = 1 تعني أن الحركة المروية على الطريق السريع $i$ هي مزدحمة.
- هذا التابع يعيد اصغر مجموع كلي للرسوم بين المدينتين $S$ و $T$, ضمن حالات حركة المرور المحدد بواسطة w.
- هذا التابع يمكن استدعاؤه كحد اقصى $100$ مرة (من أجل كل حالة اختبار).

find_pair يجب ان يستدعي الاجراء التالي للابلاغ عن الجواب:

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

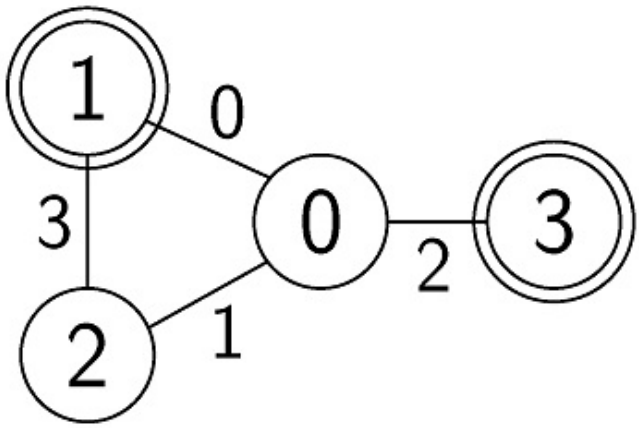
- s و t يجب ان يكونا $S$ و $T$ (الترتيب لا يهم).
- هذا الاجراء يجب ان يتم استدعاؤه مرة واحدة.

اذا لم يتحقق احد الشروط اعلاه سيتم الحكم على البرنامج كـ **Wrong Answer**. وإلا سيتم الحكم على البرنامج كـ **Accepted** ونقاطك سيتم حسابها بحسب عدد الاستدعائات لـ ask (انظر المسائل الجزئية).

## مثال

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

المقيم يستدعي find_pair(4, [0, 0, 0, 1], [1, 2, 3, 2], 1, 3).




في الشكل اعلاه، الوصلة رقم $i$ تمثل الطريق السريع رقم $i$.

بعض الاستدعائات الممكنة لـ ask والقيم المعادة الموافقة كالتالي:

| Call | Return |
|---|---|
| `ask([0, 0, 0, 0])` | 2 |
| `ask([0, 1, 1, 0])` | 4 |
| `ask([1, 0, 1, 0])` | 5 |
| `ask([1, 1, 1, 1])` | 6 |

بالنسبة لاستدعاء التابع `ask([0, 0, 0, 0])`, حركة المرور لجميع الطرق السريعة هي خفيفة والرسوم الموافقة لهم هي $1$. أرخص طريق من $S = 1$ إلى $T = 3$ هو $3 \to 0 \to 1$. الرسوم الاجمالية لهذا الطريق هي $2$. لذلك، هذ التابع يعيد $2$.

لكي يكون الجواب صحيحاً، الاجراء `find_pair` يجب ان يستدعي `answer(1, 3)` أو `answer(3, 1)`.

The file `sample-01-in.txt` in the zipped attachment package corresponds to this example. Other sample inputs are also available in the package.

## القيود

- $2 \leq N \leq 90\,000$
- $1 \leq M \leq 130\,000$
- $1 \leq A < B \leq 1\,000\,000\,000$
- لأجل كل $0 \leq i \leq M - 1$
  - $0 \leq U[i] \leq N - 1$
  - $0 \leq V[i] \leq N - 1$
  - $U[i] \neq V[i]$
- $(U[i], V[i]) \neq (U[j], V[j])$ و $(U[i], V[i]) \neq (V[j], U[j])$ $(0 \leq i < j \leq M - 1)$
- يمكنك العبور من أي مدينة إلى أي مدينة باستخدام الطرق السريعة.
- $0 \leq S \leq N - 1$
- $0 \leq T \leq N - 1$
- $S \neq T$

في هذه المسألة, المقيم هو غير متكيف. هذا يعني أن $S$ و $T$ هما ثابتين عند بداية تنفيذ المقيم ولا يعتمدان على الاسئلة التي سُئلت من قبل حلك.

## المسائل الجزئية

1. (5 نقطة) اما $S$ او $T$ هو $0$ و $N \leq 100$, $M = N - 1$
2. (7 نقطة) اما $S$ او $T$ هو $0$ و $M = N - 1$
3. (6 نقطة) $M = N - 1$ و $U[i] = i$ و $V[i] = i + 1$ $(0 \leq i \leq M - 1)$
4. (33 نقطة) $M = N - 1$
5. (18 نقطة) $A = 1$ و $B = 2$
6. (31 نقطة) لايوجد قيود اضافية

لنفترض ان برنامجك تم تحكيمه كـ **Accepted**, ويقوم بـ$X$ استدعاء لـ ask. عندها نقاطك $P$ من اجل كل حالة اختبار (بحسب رقم المسألة الجزئية) يتم حسابه كالتالي:

- المسألة الجزئية 1. $P = 5$.
- المسألة الجزئية 2. اذا كان $X \leq 60$ يكون $P = 7$. وإلا $P = 0$.
- المسألة الجزئية 3. اذا كان $X \leq 60$ يكون $P = 6$. وإلا $P = 0$.
- المسألة الجزئية 4. اذا كان $X \leq 60$ يكون $P = 33$. وإلا $P = 0$.
- المسألة الجزئية 5. اذا كان $X \leq 52$ يكون $P = 18$. وإلا $P = 0$.
- المسألة الجزئية 6.
  - اذا كان $X \leq 50$ يكون $P = 31$.
  - اذا كان $52 \leq X \leq 51$ يكون $P = 21$.
  - If $53 \leq X$, $P = 0$.

لاحظ أن نقاطك لأجل كل مسالة جزئية هي اقل نقاط من اجل كل حالة اختبار في هذه المسألة الجزئية

## مقيم الاختبار

The sample grader reads the input in the following format:

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

If your program is judged as **Accepted**, the sample grader prints `Accepted: q`, with `q` the number of calls to `ask`.

If your program is judged as **`Wrong Answer`**, it prints `Wrong Answer: MSG`, where `MSG` is one of:

- `answered not exactly once`: The procedure `answer` was not called exactly once.
- `w is invalid`: The length of `w` given to `ask` is not $M$ or `w[i]` is neither $0$ nor $1$ for some $i$ ($0 \leq i \leq M - 1$).
- `more than 100 calls to ask`: The function `ask` is called more than 100 times.
- `{s, t} is wrong`: The procedure `answer` is called with an incorrect pair `s` and `t`.