



Autobahnggebühren

Die Städte Japans sind durch ein Netzwerk aus Autobahnen miteinander verbunden. Das Netzwerk besteht aus N Städten und M Autobahnen. Jede Autobahn verbindet zwei verschiedene Städte miteinander. Keine zwei Autobahnen verbinden das gleiche Städtepaar. Städte sind von 0 bis $N - 1$ durchnummeriert und Autobahnen sind von 0 bis $M - 1$ durchnummeriert. Du kannst jede Autobahn in beide Richtungen benutzen. Du kannst mittels Autobahnen von jeder Stadt aus alle anderen Städte erreichen.

Für das Benutzen jeder Autobahn muss eine Gebühr bezahlt werden. Die Gebühr hängt von der aktuellen **Staulage** der Autobahn ab. Der Stau ist entweder **schwach** oder **stark**. Wenn der Verkehr schwach ist, beträgt die Gebühr A Yen (Japanische Währung). Wenn der Verkehr stark ist, beträgt die Gebühr B Yen. Es wird garantiert, dass $A < B$. Beachte, dass du die Werte von A und B weißt.

Du hast eine Maschine, welche dir für eine gegebene Staulage die kleinste Gesamtgebühr zwischen einem Städtepaar S und T ausrechnet ($S \neq T$).

Leider ist deine Maschine nur ein Prototyp. Die Werte von S und T sind fixiert (fest in der Maschine einprogrammiert) und sind dir nicht bekannt. Du möchtest nun S und T herausfinden. Um dies zu erreichen, planst du, der Maschine mehrere Staulagen anzugeben und mithilfe der berechneten Gesamtgebühren S und T zu bestimmen. Weil es sehr teuer ist, die Staulagen anzugeben, möchtest du die Maschine so selten wie möglich zu Rate ziehen.

Implementierungsdetails

Du sollst folgende Methoden implementieren:

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

- N : die Anzahl Städte.
- U und V : Arrays der Länge M , wobei M der Anzahl Autobahnen entspricht. Für jedes i ($0 \leq i \leq M - 1$) verbindet die Autobahn i die Städte $U[i]$ und $V[i]$.
- A : die Gebühr für eine Autobahn bei schwachem Stau.
- B : die Gebühr für eine Autobahn bei starkem Stau.
- Diese Methode wird genau einmal pro Testfall aufgerufen.
- Beachte, dass der Wert von M der Länge der Arrays entspricht. Die Länge kann wie in den Implementierungshinweisen beschrieben, abgefragt werden.

Die Methode `find_pair` kann die folgende Funktion aufrufen:

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

- Die Länge von w muss M sein. Das Array w beschreibt die Staulage.
- Für jedes i ($0 \leq i \leq M - 1$) gibt $w[i]$ die Staulage auf der i -ten Autobahn an. Der Wert von $w[i]$ muss entweder 0 oder 1 sein.
 - $w[i] = 0$ bedeutet, dass der Stau auf der i -ten Autobahn schwach ist.
 - $w[i] = 1$ bedeutet, dass der Stau auf der i -ten Autobahn stark ist.
- Diese Funktion gibt dir für die Staulage w die kleinste Gesamtgebühr zwischen den Städten S und T zurück.
- Diese Funktion kann höchstens 100 Mal aufgerufen werden (für jeden Testfall).

`find_pair` soll die folgende Methode aufrufen, um die Antwort anzugeben:

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

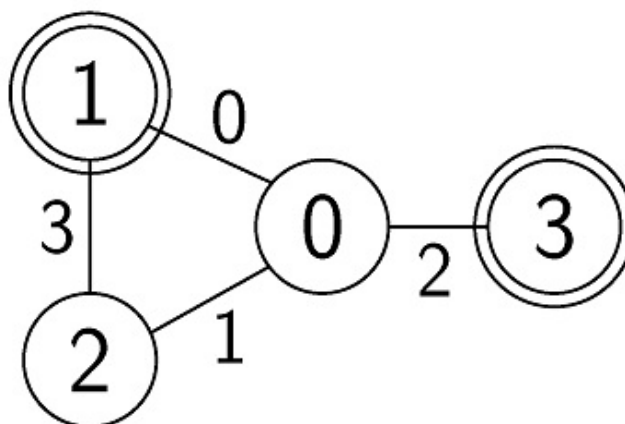
- s und t muss dem Paar S und T (die Reihenfolge spielt keine Rolle) entsprechen.
- Diese Methode muss genau einmal aufgerufen werden.

Falls eine der obigen Bedingungen nicht erfüllt ist, wird dein Programm mit **Wrong Answer** bewertet. Andernfalls wird dein Programm mit **Accepted** bewertet und deine Punktzahl wird abhängig von der Anzahl Aufrufen von `ask` bestimmt (siehe Teilaufgaben).

Beispiel

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

Der Grader ruft `find_pair(4, [0, 0, 0, 1], [1, 2, 3, 2], 1, 3)` auf.



In der obigen Figur entspricht die Kante mit der Zahl i der Autobahn i . Einige mögliche Aufrufe von `ask` und die entsprechenden Rückgabewerte sind unten

aufgelistet:

Aufruf	Rückgabewert
<code>ask([0, 0, 0, 0])</code>	2
<code>ask([0, 1, 1, 0])</code>	4
<code>ask([1, 0, 1, 0])</code>	5
<code>ask([1, 1, 1, 1])</code>	6

Für den Funktionsaufruf `ask([0, 0, 0, 0])` ist die Staulage jeder Autobahn leicht und die Gebühr beträgt 1. Die billigste Route von $S = 1$ nach $T = 3$ ist $1 \rightarrow 0 \rightarrow 3$. Die Gesamtgebühr für diese Route beträgt 2. Demzufolge gibt die Funktion 2 zurück.

Für eine korrekte Antwort soll die Methode `find_pair` entweder `answer(1, 3)` oder `answer(3, 1)` aufrufen.

Die Datei `sample-01-in.txt` im ZIP-Archiv entspricht diesem Beispiel. Andere Beispieleingaben sind ebenfalls im Archiv enthalten.

Limits

- $2 \leq N \leq 90\,000$
- $1 \leq M \leq 130\,000$
- $1 \leq A < B \leq 1\,000\,000\,000$
- Für jedes $0 \leq i \leq M - 1$
 - $0 \leq U[i] \leq N - 1$
 - $0 \leq V[i] \leq N - 1$
 - $U[i] \neq V[i]$
- $(U[i], V[i]) \neq (U[j], V[j])$ und $(U[i], V[i]) \neq (V[j], U[j])$ ($0 \leq i < j \leq M - 1$)
- Du kannst von jeder Stadt aus jede andere Stadt mithilfe der Autobahnen erreichen.
- $0 \leq S \leq N - 1$
- $0 \leq T \leq N - 1$
- $S \neq T$

In diesem Problem ist der Grader NICHT adaptiv. Dies bedeutet, dass S und T von Anfang an fixiert sind und nicht von den gestellten Anfragen deiner Lösung abhängen.

Teilaufgaben

1. (5 Punkte) entweder S oder T ist 0, $N \leq 100$, $M = N - 1$
2. (7 Punkte) entweder S oder T ist 0, $M = N - 1$
3. (6 Punkte) $M = N - 1$, $U[i] = i$, $V[i] = i + 1$ ($0 \leq i \leq M - 1$)

4. (33 Punkte) $M = N - 1$
5. (18 Punkte) $A = 1, B = 2$
6. (31 Punkte) Keine weiteren Einschränkungen.

Wird dein Programm mit **Accepted** bewertet und `ask` X Mal aufgerufen, dann erhältst du, abhängig von der Teilaufgabe, P Punkte für den Testfall wie folgt:

- Teilaufgabe 1. $P = 5$.
- Teilaufgabe 2. Falls $X \leq 60$, $P = 7$. Andernfalls $P = 0$.
- Teilaufgabe 3. Falls $X \leq 60$, $P = 6$. Andernfalls $P = 0$.
- Teilaufgabe 4. Falls $X \leq 60$, $P = 33$. Andernfalls $P = 0$.
- Teilaufgabe 5. Falls $X \leq 52$, $P = 18$. Andernfalls $P = 0$.
- Teilaufgabe 6.
 - Falls $X \leq 50$, $P = 31$.
 - Falls $51 \leq X \leq 52$, $P = 21$.
 - Falls $53 \leq X$, $P = 0$.

Beachte, dass deine Punktzahl dem Minimum der Punkte aller Testfälle pro Teilaufgabe entspricht.

Beispielgrader

Der Beispielgrader liest deine Eingabe in folgendem Format ein:

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

Falls dein Programm mit **Accepted** bewertet wird, gibt der Beispielgrader `Accepted: q` aus. Die Zahl q entspricht der Anzahl Aufrufe von `ask`.

Falls dein Programm mit **Wrong Answer** bewertet wird, gibt der Beispielgrader `Wrong Answer: MSG` aus. `MSG` entspricht einer der folgenden Nachrichten:

- `answered not exactly once`: Die Methode `answer` wurde nicht genau einmal aufgerufen.
- `w is invalid`: Die Länge von `w`, welche `ask` übergeben wird, entspricht nicht M oder `w[i]` ist weder 0 noch 1 für ein i ($0 \leq i \leq M - 1$).
- `more than 100 calls to ask`: Die Funktion `ask` wird mehr als 100 Mal aufgerufen.
- `{s, t} is wrong`: Die Methode `answer` wird mit einem ungültigen Paar `s` und `t` aufgerufen.