Патарина

Во Јапонија градовите се поврзани со мрежа од автопати. Мрежата поврзува N градови со M автопати. Секој автопат поврзува директно пар од различни градови. Не постојат два автопати кои поврзуваат ист пар градови. Градовите се нумерирани со целите броеви од 0 до $N-1$, а автопатите се нумерирани со целите броеви од 0 до $M-1$. Сите автопати се двонасочни т.е. може да се патува и во двете насоки. Од секој град до секој друг град може да се патува по мрежата од автопати.

За возење по секој автопат се плаќа патарина. Патаринатa зависи од состојбата на **сообраќајот** на автопатот. Сообраќајот може да биде **умерен** или **густ**. Кога сообраќајот е умерен, патаринатa за автопатот е A јени (валута на Јапонија). Кога сообраќајот е густ, патаринатa е B јени. Секогаш важи дека $A < B$. Да забележиме дека вие ги знаете вредностите на A и B.

Вие имате машина која за дадени податоци за состојбата на сообраќајот (на секој автопат), ја пресметува најмалата вкупна цена која треба да се плати за патарина за патување меѓу градовите S и T ($S \neq T$), притоа имајќи ги предвид дадените податоци за сообраќајот.

Меѓутоа, машината е само прототип. Вредностите S и T се фиксни (т.е. вредностите се хардкодирани во машината), но вие не ги знаете. Вие сакате да ги определите S и T. За да можете да го направите тоа, вие имате план да го проверите одговорот на машината за неколку различни податоци за состојбата на сообраќајот кои вие ќе ги дефинирате. Вие планирате да ги искористите одговорите на машината (за најмалата вкупна цена која треба да се плати за патарина) за да ги откриете S и T. Но, вие не сакате да ја "прашувате" машината многу пати затоа што дефинирањето на состојбата на сообраќајот за секој автопат е скапа операција.

Имплементациски детали

Вие треба да ја имплементирате следната процедура:

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

- `N`: бројот на градови.
- `U` и `V`: низи со должина M, каде M е бројот на автопати со кои се поврзани

градовите. За секое i ($0 \leq i \leq M-1$), автопатот i ги поврзува директно градовите `U[i]` и `V[i]`.
- `A`: цената за патарина за автопат каде има умерен сообраќај.
- `B`: цената за патарина за автопат каде има густ сообраќај.
- Оваа процедура се повикува точно еднаш за секој тест случај.
- Да забележиме дека вредноста на M е должината на низите и може да се добие како што е дадено во Забелешки за имплементацијата (Ден 2).

Процедурата `find_pair` може да ја повика следната функција:

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

- должината на `w` мора да е M. Со низата `w` е дефинирана состојбата на сообраќајот на секој автопат.
- за секое i ($0 \leq i \leq M-1$), `w[i]` е состојбата на сообраќајот на автопатот i. Вредноста на `w[i]` мора да е или 0 или 1.
  - ако `w[i] = 0` автопатот i има умерен сообраќај.
  - ако `w[i] = 1` автопатот i има густ сообраќај.
- Оваа функција ја враќа најмалата вкупна цена која треба да се плати за патарина за патување меѓу градовите S и T, притоа имајќи ги предвид податоците за состојбата на сообраќајот дадени со `w`.
- Оваа функција може да биде повикана најмногу 100 пати (за секој тест случај).

`find_pair` треба да ја повика следната функција за да го дадете одговорот на прашањето дефинирано во оваа задача:

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

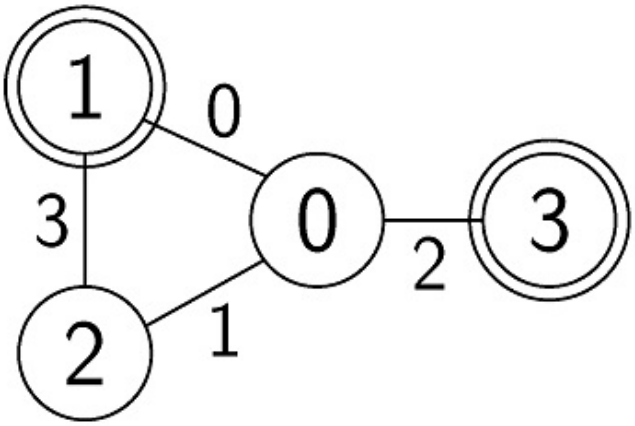
- `s` и `t` треба да се вредносите на парот S и T кои вие ги откривте (редоследот на S и T не е важен).
- Оваа процедура мора да се повика точно еднаш.

Ако некој од условите дадени погоре не е исполнет, вашата програма ќе биде оценета со **`Wrong Answer`**. Ако условите се исполнети вашата програмата ќе биде оценета со **`Accepted`** и вашиот резултат ќе се пресмета зависно од бројот на повици на функцијата `ask` (видете го делот Подзадачи).

Пример

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

Оценувачот ја повикува процедурата `find_pair(4, [0, 0, 0, 1], [1, 2, 3, 2], 1, 3)`.



Во сликата дадена погоре, реброто со број i се однесува на автопатот i.

Еден пример на повици на функцијата `ask` и вредностите кои се вратени со секој повик соодветно, е даден во следната листа:

Повик	Вредност која ја враќа функцијата
`ask([0, 0, 0, 0])`	2
`ask([0, 1, 1, 0])`	4
`ask([1, 0, 1, 0])`	5
`ask([1, 1, 1, 1])`	6

Кај повикот `ask([0, 0, 0, 0])` сообраќајот на сите автопати е умерен и патаринатa за секој автопат е 1. Најефтината рута од $S = 1$ до $T = 3$ е $1 \to 0 \to 3$. Вкупната цена на патаринатa за оваа рута е 2. Оттука, оваа функција враќа 2.

Процедурата `find_pair` треба да ја повика функцијата `answer(1, 3)` или `answer(3, 1)` за да го даде точниот одговор за вредностите на парот S и T.

Датотеката `sample-01-in.txt` во zip архивата одговара на овој пример. Во оваа архива исто така се достапни и други примери за влез.

Ограничувања

- $2 \leq N \leq 90\,000$
- $1 \leq M \leq 130\,000$
- $1 \leq A < B \leq 1\,000\,000\,000$
- За секој $0 \leq i \leq M - 1$
  - $0 \leq U[i] \leq N - 1$
  - $0 \leq V[i] \leq N - 1$
  - $U[i] \neq V[i]$
- $(U[i], V[i]) \neq (U[j], V[j])$ и $(U[i], V[i]) \neq (V[j], U[j])$ $(0 \leq i < j \leq M - 1)$

- Од секој град до секој друг град може да се патува по мрежата од автопати.
- $0 \leq S \leq N - 1$
- $0 \leq T \leq N - 1$
- $S \neq T$

Во оваа задача оценувачот НЕ е адаптивен. Ова значи дека S и T се фиксирани на почетокот на извршувањето на оценувачот и тие не зависат од повиците на функцијата `ask` во вашето решение.

Подзадачи

1. (5 поени) или S или T е 0, $N \leq 100$, $M = N - 1$
2. (7 поени) или S или T е 0, $M = N - 1$
3. (6 поени) $M = N - 1$, $U[i] = i$, $V[i] = i + 1$ $(0 \leq i \leq M - 1)$
4. (33 поени) $M = N - 1$
5. (18 поени) $A = 1$, $B = 2$
6. (31 поени) Нема дополнителни ограничувања

Да претпоставиме дека вашата програма е оценета со **`Accepted`** и дека вие сте направиле X повици на `ask`. Тогаш вашиот резултат P за даден тест случај ќе се пресмета, зависно од тоа од која подзадача е, на следниот начин:

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

Да забележиме дека вашиот резултат за секоја подзадача ќе биде најмалиот резултат меѓу сите тест случаи во дадената подзадача.

Пример оценувач

Пример оценувачот го чита влезот во следниот формат:

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

Ако вашата програма е оценета со **`Accepted`**, пример оценувачот печати `Accepted: q`, каде `q` е бројот на повици на `ask`.

Ако вашата програма е оценета со **`Wrong Answer`**, пример оценувачот печати `Wrong Answer: MSG`, каде `MSG` може да биде:

- `answered not exactly once`: Процедурата `answer` не е повикана точно еднаш.
- `w is invalid`: Должината на `w`, кој се предава на `ask`, не е M или `w[i]` не е ни 0 ни 1 за некое i ($0 \leq i \leq M-1$).
- `more than 100 calls to ask`: Функцијата `ask` е повикана повеќе од 100 пати.
- `{s, t} is wrong`: Процедурата `answer` е повикана со грешна вредност за парот `s` и `t`.