Магистрални такси

Градовете в Япония са свързани в мрежа от магистрали. Мрежата се състои от N града и от M магистрали. Всяка магистрала свързва два различни града и два града са свързани с най-много една магистрала. Градовете са номерирани от 0 до $N-1$, а магистралите – от 0 до $M-1$. По всяка магистрала може да се пътува в двете ѝ посоки. От всеки град може да се пътува по магистрали до всеки дрг град.

За преминаване по всяка от магистралите трябва да бъде платена такса. Таксата зависи от интензивността на трафика. Трафикът може да бъде **слаб** или **силен**. Когато трафикът е слаб, таксата е A йени, а когато е силен, таксата е B йени. Гарантирано е, че $A < B$. Вие знаете стойностите на A и B.

Разполагате с машина, която при зададен вид на трафика по магистралите пресмята най-малката обща сума на таксите, която трябва да се плати, за да се пътува от град S град T ($S \neq T$).

Обаче машината, която използваме е само прототип и стойностите на S и T са фиксирани в нея, които вие не знаете. Вие трябва да намерите S и T. За да ги намерите, вие възнамерявате да зададете по няколко видове трафик на машината и като използвате пресметнатите от машината такси, да намерите S и T. Понеже задаването на много видове трафик е трудно, вие трябва да използвате машината възможно по-малко пъти.

Детайли за реализация

Трябва да напишете следната процедура:

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

- `N`: брой на градовете.
- `U` и `V`: масиви с дължина M, където M е броят на магистралите. За всяко i ($0 \leq i \leq M-1$), магистралата i свързва градовете `U[i]` и `V[i]`.
- `A`: таксата по магистрала, когато трафикът е слаб.
- `B`: таксата по магистрала, когато трафикът е силен.
- Тази процедура трябва да извикате точно веднъж за всеки тест.
- Обърнете внимание, че M е дължината на масивите и може да се намери, както е описано в таблицата, дадена в "Бележки".

Процедурата `find_pair` може да извиква следната функция:

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

- Дължината на `w` трябва да е M. Масивът `w` описва вида на трафика.
- За всяко i ($0 \leq i \leq M-1$), `w[i]` описва вида на трафика по магистрала i. Стойнсотта на `w[i]` трябва да бъде или 0, или 1.
  - `w[i] = 0` означава, че трафикът по магистрала i е слаб.
  - `w[i] = 1` означава, че трафикът по магистрала i е силен.
- Тази функция връща най-малката обща сума на таксите за пътуването от S до T при вид на трафока, зададен чрез `w`.
- Тази функция може да бъде извикана най-много 100 пъти при всеки тест.

`find_pair` трябва да извика следната процедура, за да докладва резултата:

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

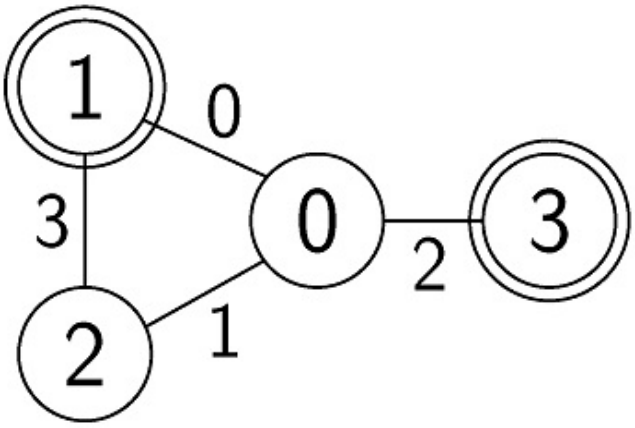
- `s` и `t` трябва да задават двойката S и T (редът няма значение).
- Тази процедура трябва да бъде извикана точно веднъж.

Ако някои от горните условия не са изпълнени, вашата програма ще бъде оценена с **`Wrong Answer`**. В противен случай, вашата програма ще бъде оценена с **`Accepted`** и вашите точки ще бъдат пресметнати чрез броя на извикванията на `ask` (вижте описанието на подзадачите).

Пример

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

Грейдерът извиква `find_pair(4, [0, 0, 0, 1], [1, 2, 3, 2], 1, 3)`.



На фигурата ребро с номер i съответства на магистрала i.

Някои възможни извиквания на `ask` и съответни върнати стойности са показани тук:

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

При извикването `ask([0, 0, 0, 0])`, трафикът по всяка магистрала е слаб и таксата за магистралата е 1. Най-евтиният маршрут от $S = 1$ до $T = 3$ е $1 \to 0 \to 3$. Общата сума на таксите по този маршрут е 2. Така тази функция връща 2.

За правилния отговор процедурата `find_pair` трябва да извика `answer(1, 3)` или `answer(3, 1)`.

Файлът `sample-01-in.txt`, който е даден като зипнат атачмънт, съответства на този пример. Предоставят ви се и други примери.

Ограничения

- $2 \le N \le 90\,000$
- $1 \le M \le 130\,000$
- $1 \le A < B \le 1\,000\,000\,000$
- За всяко $0 \le i \le M - 1$
  - $0 \le U[i] \le N - 1$
  - $0 \le V[i] \le N - 1$
  - $U[i] \neq V[i]$
- $(U[i], V[i]) \neq (U[j], V[j])$ and $(U[i], V[i]) \neq (V[j], U[j])$ $(0 \le i < j \le M - 1)$
- Може да се пътува от всеки град до всеки друг град по магистралите.
- $0 \le S \le N - 1$
- $0 \le T \le N - 1$
- $S \neq T$

За тази задача грейдерът НЕ Е адаптивен, т.е. S и T са фиксирани в началото на работата на грейдера и не зависят от извикванията, направени от вашето решение.

Подзадачи

1. (5 точки) едно от S или T е 0, $N \le 100$, $M = N - 1$
2. (7 точки) едно от S или T е 0, $M = N - 1$

3. (6 точки) $M = N - 1$, $U[i] = i$, $V[i] = i + 1$ $(0 \leq i \leq M - 1)$
4. (33 точки) $M = N - 1$
5. (18 точки) $A = 1$, $B = 2$
6. (31точки) Няма допълнителни ограничения.

Когато вашата програма е оценена с **Accepted** и е направила X извиквания на `ask`, тогава вие ще получите P точки за тестовия пример, които в зависимост от номера на подзадачата се пресмятат, както следва:

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

Обърнете внимание, че вашите точки за всяка задача се пресмятат като минимум на точките, получени за тестовете в подзадачата.

Примерен грейдер

Примерният грейдер чете вход в следния формат:

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

Ако вашата програма е оценена с **Accepted**, примерният грейдер отпечатва `Accepted: q`, където `q` е броя на извикванията на `ask`.

Ако вашата програма е оценена с **Wrong Answer**, примерният грейдер отпечатва `Wrong Answer: MSG`, където `MSG` е едно от следните съобщения:

- `answered not exactly once`: процедурата `answer` не е извикана точно веднъж.
- `w is invalid`: дължината на `w`, подадена на `ask` не е M, или пък `w[i]` е различно от някоя от двете стойности 0 или 1 за някое i $(0 \leq i \leq M - 1)$.
- `more than 100 calls to ask`: функцията `ask` е извикана повече от 100 пъти.
- `{s, t} is wrong`: процедурата `answer` е извикана с неправилни стойности на двойката `s` и `t`.