Платные дороги

Города в Японии соединены дорожной сетью. Сеть состоит из N городов и M дорог. Каждая дорога соединяет пару различных городов. Никакие две дороги не соединяют одну и ту же пару городов. Города пронумерованы от 0 до $N-1$, а дороги пронумерованы от 0 до $M-1$. По каждой дороге можно проехать в любом из двух направлений. Из любого города в любой другой можно добраться по дорогам.

За проезд по каждой дороге необходимо заплатить пошлину. Пошлина на каждой дороге зависит от **загруженности** этой дороги. Загруженность дороги может быть **слабой** либо **сильной**. Пошлина для слабо загруженной дороги составляет A йен (японских денежных единиц), а для сильно загруженной дороги — B йен. Гарантируется, что $A < B$. Обратите внимание, что значения A и B вам известны.

У вас есть устройство, которое по заданной вами информации о загруженности всех дорог вычисляет наименьшую суммарную пошлину, которую необходимо заплатить для проезда между городами S и T ($S \neq T$) при такой загруженности дорог.

Ваше устройство еще находится в стадии разработки. Значения S и T зафиксированы внутри устройства и вам неизвестны. Вы хотите найти S и T. Для этого вы собираетесь воспользоваться устройством в нескольких сценариях загруженности дорог и, пользуясь полученными из устройства значениями пошлин для проезда, определить S и T. Поскольку управлять устройством сложно, вы хотите воспользоваться им не слишком много раз.

Детали реализации

Вам требуется реализовать следующую процедуру:

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

- `N`: количество городов.
- `U` и `V`: массивы длины M, где M — количество дорог в сети. Для каждого i $(0 \leq i \leq M-1)$ дорога i соединяет города `U[i]` и `V[i]`.
- `A`: пошлина за проезд по слабо загруженной дороге.
- `B`: пошлина за проезд по сильно загруженной дороге.
- Процедура будет вызвана ровно один раз для каждого теста.

- Обратите внимание, что M — это длина массивов, способ получения длины массива описан в памятке о деталях реализации.

Процедура `find_pair` может вызывать следующую функцию:

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

- Длина `w` должна быть равна M. Массив `w` описывает сценарий загруженности дорог.
- Для каждого i ($0 \leq i \leq M-1$), `w[i]` задает загруженность дороги i. Значение `w[i]` должно быть равно 0 или 1:
  - `w[i] = 0` означает, что дорога i слабо загружена.
  - `w[i] = 1` означает, что дорога i сильно загружена.
- Функция возвращает наименьшую суммарную пошлину, которую необходимо заплатить для проезда между городами S и T ($S \neq T$) при условиях загруженности, заданных `w`.
- Функция может быть вызвана не более 100 раз (для каждого теста).

Чтобы сообщить ответ, `find_pair` должна вызвать следующую процедуру:

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

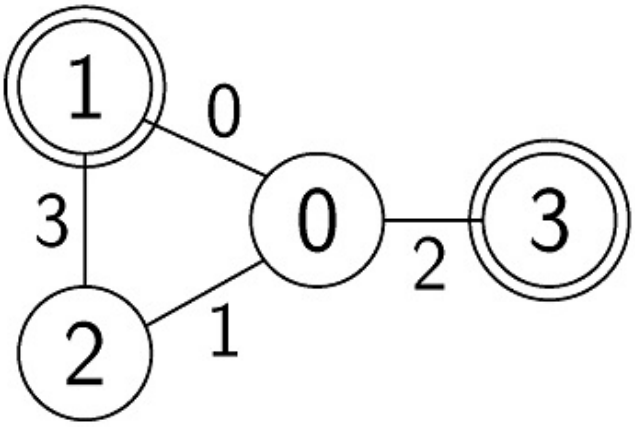
- `s` и `t` должны содержать значения S и T (порядок не важен).
- Процедура должна быть вызвана ровно один раз.

Если какие-то из описанных выше требований не выполнены, ваша программа получает вердикт **`Wrong Answer`**. Иначе программа получает вердикт **`Accepted`**, и полученные баллы вычисляются в зависимости от количества вызовов `ask` (см. раздел Подзадачи).

Пример

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

Проверяющий модуль вызывает `find_pair(4, [0, 0, 0, 1], [1, 2, 3, 2], 1, 3)`.



На рисунке выше ребро с номером i соответствует дороге i.

Некоторые возможные вызовы `ask` и соответствующие им возвращаемые значения показаны ниже:

Вызов	Возвращаемое значение
`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 вызове `ask([0, 0, 0, 0])` все дороги слабо загружены и пошлина для каждой из них равна 1. Самый дешевый путь из $S = 1$ в $T = 3$ — это $1 \to 0 \to 3$. Суммарная пошлина для пути равна 2. Таким образом, эта функция вернёт 2.

Для правильного ответа процедура `find_pair` должна вызвать `answer(1, 3)` или `answer(3, 1)`.

Файл `sample-01-in.txt` в прилагаемом архиве соответствует этому примеру. В архиве есть и другие примеры входных данных.

Ограничения

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

- Из каждого города можно добраться в любой другой по дорогам.
- $0 \leq S \leq N - 1$
- $0 \leq T \leq N - 1$
- $S \neq T$

В этой задаче проверяющий модуль НЕ ЯВЛЯЕТСЯ адаптивным. Это означает, что S и T зафиксированы в начале запуска модуля и не зависят от запросов, сделанных вашим решением.

Подзадачи

1. (5 баллов) S либо T равно 0, $N \leq 100$, $M = N - 1$
2. (7 баллов) S либо T равно 0, $M = N - 1$
3. (6 баллов) $M = N - 1$, $U[i] = i$, $V[i] = i + 1$ ($0 \leq i \leq M - 1$)
4. (33 балла) $M = N - 1$
5. (18 баллов) $A = 1$, $B = 2$
6. (31 балл) Нет дополнительных ограничений

Предположим, что ваша программа получила вердикт **Accepted** и сделала X вызовов `ask`. Тогда ваша оценка P за тест, в зависимости от подзадачи, вычисляется следующим образом:

- Подзадача 1. $P = 5$.
- Подзадача 2. Если $X \leq 60$, то $P = 7$. В противном случае $P = 0$.
- Подзадача 3. Если $X \leq 60$, то $P = 6$. В противном случае $P = 0$.
- Подзадача 4. Если $X \leq 60$, то $P = 33$. В противном случае $P = 0$.
- Подзадача 5. Если $X \leq 52$, то $P = 18$. В противном случае $P = 0$.
- Подзадача 6.
  - Если $X \leq 50$, то $P = 31$.
  - Если $51 \leq X \leq 52$, то $P = 21$.
  - Если $53 \leq X$, то $P = 0$.

Обратите внимание, что ваши баллы за каждую подзадачу — это минимальный результат среди всех тестов этой подзадачи.

Пример проверяющего модуля

Пример проверяющего модуля читает входные данные в следующем формате:

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

Если ваша программа оценена как **Accepted**, пример тестирующего модуля печатает `Accepted: q`, где `q` — число вызовов `ask`.

Если ваша программа оценена как **`Wrong Answer`**, он печатает `Wrong Answer: MSG`, где `MSG` — одно из следующих предложений:

- `answered not exactly once`: процедура `answer` была вызвана не в точности один раз.
- `w is invalid`: длина `w`, переданного в `ask`, не равна M, либо для некоторого i $(0 \leq i \leq M-1)$ `w[i]` не равно ни 0, ни 1.
- `more than 100 calls to ask`: функция `ask` вызвана более 100 раз.
- `{s, t} is wrong`: процедура `answer` вызвана с некорректной парой `s` и `t`.