B. Swapping Cities

Time limit	2 s
Memory limit	512 MB

Описание

В Индонезии N городов, пронумерованных от 0 до $N-1$. Также там есть M двусторонних дорог, пронумерованных от 0 до $M-1$. Каждая дорога соединяет два различных города. Дорога с номером i соединяет города $U[i]$ и $V[i]$, проезд по ней на автомобиле требует $W[i]$ бензина. Города соединены дорогами таким образом, что от любого города можно добраться по дорогам до любого другого.

В каждый из следующих Q дней некоторая пара городов хотела бы установить дипломатические отношения. А именно, в j-й день города $X[j]$хотел бы установить дипломатические отношения с городом $Y[j]$. Чтобы этого добиться, города $X[j]$ должен послать представителя в город $Y[j]$на автомобиле. Аналогично, город $Y[j]$ должен послать в города $X[j]$своего представителя на автомобиле.

Чтобы избежать пробок, эти два автомобиля не должны встретиться по дороге. А именно, эти автомобили не должны оказаться одновременно в одном и том же городе, а также не должны в один и тот же момент двигаться навстречу друг другу по одной и той же дороге. Заехав на дорогу, автомобиль должен проехать её до конца, он не может развернуться и вернуться в город, из которого выехал. При этом автомобили могут посещать один и тот же город или одну и ту же дорогу несколько раз, а также могут ждать в городе произвольное время.

Поскольку автомобили с большим бензобаком стоят дорого, оба города хотели бы, чтобы автомобили ехали по таким маршрутам, чтобы максимальный необходимый объем бензобака был как можно меньше. В каждом городе есть заправочные станции, так что требуемый объем бенозбака равен **максимальному** требуемому количеству бензина среди всех дорог, по которым проедет автомобиль.

Задание

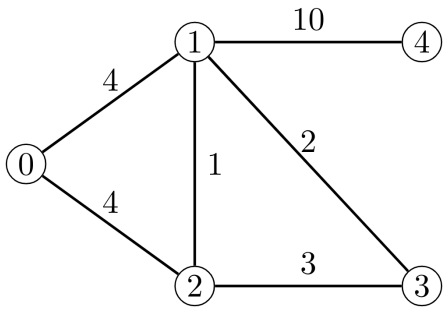
Вам необходимо реализовать две функции: `init` и `getMinimumFuelCapacity`.

- `init(N, M, U, V, W)` - Эта функция будет вызывана проверяющим модулем ровно один раз, перед вызовами `getMinimumFuelCapacity`.
  - N: Целое число - количество городов.
  - M: Целое число - количество дорог.
  - U: Массив из M целых чисел, для каждой дороги соответствующий элемент массива задает один из городов, соединенных этой дорогой.
  - V: Массив из M целых чисел, для каждой дороги соответствующий элемент массива задает второй из городов, соединенных этой дорогой.
  - W: Массив из M целых чисел, для каждой дороги задано количество бензина, требуемое для проезда по дороге.
- `getMinimumFuelCapacity(X, Y)` - Эта функция будет вызвана проверяющим модулем ровно Q раз.
  - X: Целое число, задающее первый город.
  - Y: Целое число, задающее второй город.
  - Функция должна вернуть целое число. Рассмотрим два автомобиля, которые будут доставлять представителей из города X в город Y и наоборот с ограничениями, описанными в условии задач, требуется минимизировать максимальный из объемов бензобака этих автомобилей, и

вернуть полученное значение. Если направить автомобили с представителями в соответствии с изложенными правилами невозможно, функция должна вернуть -1.

Пример

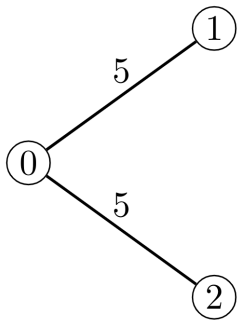
В первом примере, $N = 5$, $M = 6$, $U = [0, 0, 1, 1, 1, 2]$, $V = [1, 2, 2, 3, 4, 3]$, $W = [4, 4, 1, 2, 10, 3]$, $Q = 3$, $X = [1, 2, 0]$, $Y = [2, 4, 1]$. Система дорог изображена на следующем рисунке:



Исходно проверяющий модуль вызовет функцию `init(5, 6, [0, 0, 1, 1, 1, 2], [1, 2, 2, 3, 4, 3], [4, 4, 1, 2, 10, 3])`. После этого проверяющий модуль сделает следующие вызовы:

- `getMinimumFuelCapacity(1, 2)`. Сначала автомобиль из первого города направится с город 3. Затем автомобиль из второго города направится в первый город. Наконец затем автомобиль из третьего города направится во второй города. Максимальный объем бензобака из двух автомобилей равен 3 (такой объем необходим, чтобы доехать из третьего города до второго). Нет способа направить представитеелй, используя два автомобиля с меньшим объемом бензобака, поэтому функция должна вернуть 3.
- `getMinimumFuelCapacity(2, 4)`. Любой автомобиль, выезжающий из четвертого города, должен иметь объем бензобака хотя бы 10, поэтому функция должна вернуть 10.
- `getMinimumFuelCapacity(0, 1)`. Функция должна вернуть 4.

Во втором примере $N = 3$, $M = 2$, $U = [0, 0]$, $V = [1, 2]$, $W = [5, 5]$, $Q = 1$, $X = [1]$, $Y = [2]$. Система дорог изображена на следующем рисунке:



Проверяющий модуль исходно сделает вызов `init(3, 2, [0, 0], [1, 2], [5, 5])`. После этого проверяющий модуль сделает вызов:

- `getMinimumFuelCapacity(1, 2)`. Автомобили не могут, согласно описанным правилам, совместно добраться из города 1 до города 2 и наоборот, поэтому функция должна вернуть -1.

Ограничения

- $2 \le N \le 100\,000$.
- $N - 1 \le M \le 200\,000$.
- $0 \le U[i] < V[i] < N$.
- Между каждой парой городов не более одной дороги.
- Можно добраться от любого города до любого другого по дорогам.
- $1 \le W[i] \le 10^9$.
- $1 \le Q \le 200\,000$.
- $0 \le X[j] < Y[j] < N$.

Подзадача 1 (6 баллов)

- Из каждого города выходит не более двух дорог

Подзадача 2 (7 баллов)

- $M = N - 1$.
- $U[i] = 0$.

Подзадача 3 (17 баллов)

- $Q \le 5$.
- $N \le 1\,000$.
- $M \le 2\,000$.

Подзадача 4 (20 баллов)

- $Q \le 5$.

Подзадача 5 (23 баллов)

- $M = N - 1$.

Подзадача 6 (27 баллов)

- Нет дополнительных ограничений.

Пример проверяющего модуля

Пример проверяющего модуля читает данные со стандартного потока ввода в следующем формате:

```
N M
U[0] V[0] W[0]
U[1] V[1] W[1]
.
.
.
U[M-1] V[M-1] W[M-1]
Q
X[0] Y[0]
X[1] Y[1]
.
.
.
X[Q-1] Y[Q-1]
```

Для каждого вызова `getMinimumFuelCapacity` пример проверяющего модуля выводит на стандартный поток вывода значение, которое вернула функция.

Приложения