B. Swapping Cities

Time limit	2 s
Memory limit	512 MB

Description

Индонезияда 0 - $N-1$ деп аталган N шаарлары бар. Ошондой эле 0 - $M-1$ чейин номурланган M эки тараптуу жолдор бар. Ар бир жол эки башка шаарды бириктирет. i - жол $U[i]$ - шаарды жана $V[i]$ - шаарды байланыштырат жана $W[i]$ - унаа менен өткөндө газдын бирдигин керектейт. Шаарлар бир-бирине туташкандыктан, жолдор аркылуу каалаган жуптардын ортосунда жүрүүгө болот.

Кийинки Q ар бир күнү үчүн, экиден шаар саясий мамиле түзүүнү каалашат. Атап айтканда, j - күнү $X[j]$ - шаар $Y[j]$ - менен саясий мамиле түзүүнү каалайт. Бул үчүн $X[j]$ - шаар $Y[j]$ - шаарга унаа менен баруу үчүн өкүл жөнөтүшү керек. Ошо сыяктуу эле, $Y[j]$ - шаар дагы $X[j]$ - шаарга автоунаа менен баруу үчүн өз өкүлүн жибериши керек.

Тыгындарды болтурбоо үчүн, эки унаа тең кайсы убакта болбосун жолугушууга болбойт. Тактап айтканда, эки автоунаа бир эле убакта бир шаарда болбошу керек. Ошондой эле, эки автоунаа бир эле учурда карама-каршы багытта бир эле жолду кесип өтпөшү керек. Мындан тышкары, жолду кесип өткөн унаалар жолду бүтүрүп, көздөгөн шаарга кетиши керек (башкача айтканда, автоунаалар жолдун ортосуна бурулуш жасоого тыюу салынат). Бирок автоунааларга бир эле шаарга жана жолго бир нече жолу барууга уруксат берилет. Мындан тышкары, унаалар каалаган шаарда, каалаган убакта күтө алышат.

Күйүүчү майдын жогорку кубаттуулугу жогору болгон унаалар кымбат болгондуктан, эки шаар тең эки унаа үчүн каттамдарды тандап алууну каалашат, ошондуктан эки унаанын эң көп күйүүчү майын минималдаштырышат. Ар бир шаарда газдын чексиз берилиши менен май куюучу жайлар бар, ошондуктан унаа талап кылган отундун кубаттуулугу унаа менен өткөн бардык жолдордо газды керектөө **максимуму** болуп саналат.

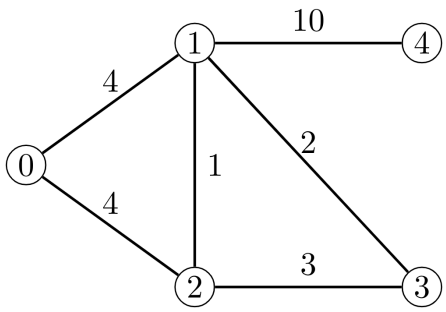
Task

You have to implement `init` and `getMinimumFuelCapacity` functions.

- `init(N, M, U, V, W)` - This function will be called by the grader exactly once before any `getMinimumFuelCapacity` calls.
  - N: An integer representing the number of cities.
  - M: An integer representing the number of roads.
  - U: An array of M integers representing the first endpoint of the roads.
  - V: An array of M integers representing the second endpoint of the roads.
  - W: An array of M integers representing the gas consumption of the roads.
- `getMinimumFuelCapacity(X, Y)` - This function will be called by the grader exactly Q times.
  - X: An integer representing the first city.
  - Y: An integer representing the second city.
  - This function must return an integer representing the minimum unit of fuel capacity of the maximum fuel capacity of the two cars such that a representative from the X-th city can go to the Y-th city and a representative from the Y-th city can go to the X-th city following the rules explained in the problem statement, or -1 if it is impossible to do so.

Example

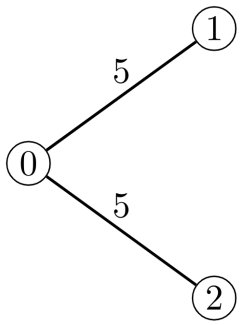
In the first example, $N = 5$, $M = 6$, $U = [0, 0, 1, 1, 1, 2]$, $V = [1, 2, 2, 3, 4, 3]$, $W = [4, 4, 1, 2, 10, 3]$, $Q = 3$, $X = [1, 2, 0]$, $Y = [2, 4, 1]$. The example is illustrated by the following image:



The grader will initially call `init(5, 6, [0, 0, 1, 1, 1, 2], [1, 2, 2, 3, 4, 3], [4, 4, 1, 2, 10, 3])`. After that, the grader will call the following:

- `getMinimumFuelCapacity(1, 2)`. First, the car from the first city can go to the third city. Next, the car from the second city can go to the first city, and the car from the third city can go to the second city. Therefore, the maximum fuel capacity of the two cars is 3 units of fuel (required to go from the third city to the second city. There is no route that requires less fuel capacity, thus the function should return 3.
- `getMinimumFuelCapacity(2, 4)`. Any car that goes to or from the fourth city should require 10 units of fuel capacity, thus the function should return 10.
- `getMinimumFuelCapacity(0, 1)`. The function should return 4.

In the second example, $N = 3$, $M = 2$, $U = [0, 0]$, $V = [1, 2]$, $W = [5, 5]$, $Q = 1$, $X = [1]$, $Y = [2]$. The example is illustrated by the following image:



The grader will initially call `init(3, 2, [0, 0], [1, 2], [5, 5])`. After that, the grader will call the following:

- `getMinimumFuelCapacity(1, 2)`. It is impossible for the car in the first city to go to the second city without meeting the other car at some time, thus the function should return -1.

Constraints

- $2 \le N \le 100\,000$.
- $N - 1 \le M \le 200\,000$.
- $0 \le U[i] < V[i] < N$.
- There is at most one road between each pair of cities.
- It is possible to travel between any pair of cities through the roads.
- $1 \le W[i] \le 10^9$.
- $1 \le Q \le 200\,000$.

- $0 \le X[j] < Y[j] < N$.

Subtask 1 (6 points)

- Each city is an endpoint of at most two roads.

Subtask 2 (7 points)

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

Subtask 3 (17 points)

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

Subtask 4 (20 points)

- $Q \le 5$.

Subtask 5 (23 points)

- $M = N - 1$.

Subtask 6 (27 points)

- No additional constraints.

Sample Grader

The sample grader reads the input in the following format:

```
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]
```

For each `getMinimumFuelCapacity` call, the sample grader prints the value returned by the function.