# B. Swapping Cities

| Time limit | 2 s |
|---|---|
| Memory limit | 512 MB |

## Description

There are $N$ cities in Indonesia, numbered from $0$ to $N-1$. There are also $M$ two-way roads, numbered from $0$ to $M-1$. Each road connects two different cities. The $i$-th road connects the $U[i]$-th city and the $V[i]$-th city and consumes $W[i]$ units of gas when traversed by car. The cities are connected such that it is possible to travel between any pair of cities through the roads.

For each of the next $Q$ days, a pair of cities would like to establish a political relationship. In particular, on the $j$-th day, the $X[j]$-th city would like to establish a political relationship with the $Y[j]$-th city. In order to do this, the $X[j]$-th city should send a representative to go to the $Y[j]$-th city by car. Similarly, the $Y[j]$-th city should also send a representative to go to the $X[j]$-th city by car.

To avoid congestion, both cars should not meet at any point in time. In particular, both cars should not be in the same city at the same time. Also, both cars should not traverse the same road in the opposite direction at the same time. Additionally, cars that traverse the road must complete the road and go to the destination city (in other words, cars are not allowed to make a U-turn in the middle of a road). However, cars are allowed to visit the same city and road more than once. In addition, cars may also wait at any city at any point in time.

Since cars with high fuel capacity are expensive, both cities would like to choose routes for both cars such that the maximum fuel capacity of the two cars is minimized. There are gas stations in each city with an infinite supply of gas, thus the fuel capacity required by a car is the **maximum** gas consumption among all roads traversed by the car.

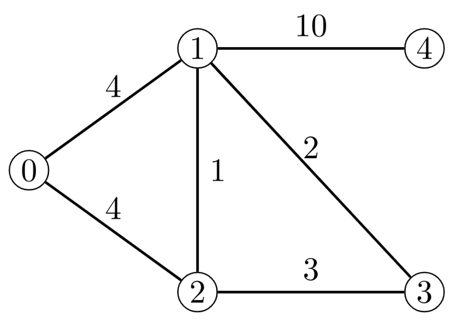
## Task

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

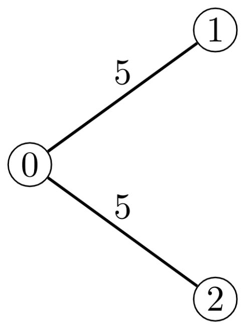
In the first example, $N=5$, $M=6$, $U=[0,0,1,1,1,2]$, $V=[1,2,2,3,4,3]$, $W=[4,4,1,2,10,3]$, $Q=3$, $X=[1,2,0]$, $Y=[2,4,1]$. The example is illustrated by the following image:



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

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

### 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)

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

### Subtask 4 (20 points)

- $Q \leq 5$.

### 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.