B. 자매 도시

Time limit	2 s
Memory limit	512 MB

설명

인도네시아에는 N 개의 도시가 있고, 0부터 $N-1$까지 번호가 매겨져 있다. 또 M 개의 양방향 도로가 있는데, 0 부터 $M-1$까지 번호가 매겨져 있다. 각 도로는 두 개의 서로 다른 도시를 연결한다. i 번 도로는 $U[i]$ 번 도시와 $V[i]$ 번 도시를 연결하고, 자동차로 여행하려면 휘발유 $W[i]$ 만큼이 필요하다. 어떤 두 도시도 서로 오갈 수 있도록 도로망이 구축되어 있다.

앞으로 Q 일 동안, 매일매일 한 쌍의 도시가 자매 도시 관계를 맺으려고 한다. 구체적으로는, j번째 날, $X[j]$ 번 도시는 $Y[j]$ 번 도시와 자매 도시 관계를 맺으려고 한다. 이러려면, $X[j]$번 도시는 대표단을 자동차를 이용해서 $Y[j]$ 번 도시로 보낸다. 비슷하게, $Y[j]$ 번 도시도 대표단을 자동차를 이용해서 $X[j]$ 번 도시로 보낸다.

혼잡을 막기 위해서, 두 자동차는 어느 순간에도 만나면 안된다. 보다 구체적으로는, 두 자동차가 동시에 같은 도시에 있으면 안된다. 또, 같은 도로를 동시에 서로 반대 방향으로 여행해도 안된다. 추가로, 어떤 도로를 여행하는 자동차는 이 도로를 끝까지 가서 목적지에 도착해야 한다. (다른 말로 하면, 도로 중간에서 유턴할 수 없다.) 그렇지만, 자동차는 같은 도시 또는 같은 도로를 한 번 이상 방문할 수 있다. 또, 자동차는 언제든지 어떤 도시에서든지 대기할 수 있다.

연료 탱크가 큰 자동차는 비싸기 때문에, 두 도시는 사용할 두 자동차의 연료 탱크의 최대 용량을 최소화하는 경로를 택하고 싶다. 각각의 도시에는 무한히 많은 휘발유가 있는 주유소가 있기 때문에, 자동차가 필요한 연료 탱크의 최대 용량은 자동차가 이용할 모든 도로에서 **최대**로 필요한 휘발유의 양이다.

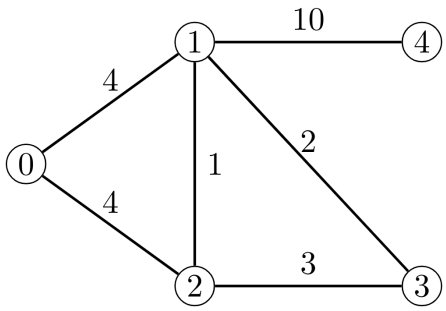
할 일

다음 `init`와 `getMinimumFuelCapacity` 함수를 구현해야 한다.

- `init(N, M, U, V, W)` - 이 함수는 `getMinimumFuelCapacity`함수를 호출하기 전에 정확히 한 번 호출된다.
  - N: 도시의 수를 나타내는 정수.
  - M: 도로의 수를 나타내는 정수.
  - U: 길이 M인 정수의 배열로 도로의 한 쪽 끝을 나타낸다.
  - V: 길이 M인 정수의 배열로 도로의 다른 쪽 끝을 나타낸다.
  - W: 길이 M인 정수의 배열로 각 도로를 여행하는데 필요한 휘발유의 양을 나타낸다.
- `getMinimumFuelCapacity(X, Y)` - 이 함수는 그레이더가 정확히 Q 번 호출한다.
  - X: 첫번째 도시를 나타내는 정수.
  - Y: 두번째 도시를 나타내는 정수.
  - 이 함수는 X번 도시의 대표단이 Y번 도시로, Y번 도시의 대표단이 X번 도시로 위에서 설명한 규칙을 따라 여행할 때 두 자동차가 필요한 연료 탱크의 최대 용량의 최소값을 리턴한다. 만약 규칙을 따라 여행하는 것이 불가능하다면, -1을 리턴한다.

예제

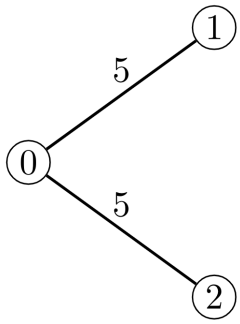
첫번째 예제에서, $N=5$, $M=6$, $U=[0,0,1,1,1,2]$, $V=[1,2,2,3,4,3]$, $W=[4,4,1,2,10,3]$, $Q=3$, $X=[1,2,0]$, $Y=[2,4,1]$이다. 이 예제는 다음 그림과 같다.



그레이더는 처음에 `init(5, 6, [0, 0, 1, 1, 1, 2], [1, 2, 2, 3, 4, 3], [4, 4, 1, 2, 10, 3])`를 호출한다. 그 후, 그레이더는 다음 일들을 한다.

- `getMinimumFuelCapacity(1, 2)`. 먼저, 1번 도시에서 출발한 자동차는 3번 도시로 이동한다. 다음, 2번 도시에서 출발한 자동차는 1번 도시로 이동하고, 3번 도시에 있던 자동차는 2번 도시로 이동한다. 따라서 두 자동차가 필요한 연료 용량의 최대값은 3이다. (3번 도시에서 2번 도시로 이동하는데 필요) 이보다 적은 연료 용량으로 갈 수 있는 경로가 없기 때문에, 이 함수의 리턴값은 3이어야 한다.
- `getMinimumFuelCapacity(2, 4)`. 4번 도시에서 오거나 4번 도시로 가는 자동차는 연료가 10만큼 필요하기 때문에, 이 함수의 리턴값은 10이어야 한다.
- `getMinimumFuelCapacity(0, 1)`. 이 함수는 4를 리턴해야 한다.

두번째 예제에서는, $N = 3, M = 2, U = [0, 0], V = [1, 2], W = [5, 5], Q = 1, X = [1], Y = [2]$이다. 이 예제는 다음 그림과 같다.



그레이더는 처음에 `init(3, 2, [0, 0], [1, 2], [5, 5])`를 호출한다. 그 후, 그레이더는 다음 일을 한다.

- `getMinimumFuelCapacity(1, 2)`. 자동차가 1번 도시에서 2번 도시로 이동하는 과정에서 다른 자동차를 만나지 않는 것은 불가능하기 때문에, 이 함수는 -1을 리턴해야 한다.

제약조건

- $2 \le N \le 100\,000$.
- $N - 1 \le M \le 200\,000$.
- $0 \le U[i] < V[i] < N$.
- 두 도시를 잇는 도로는 최대 1개이다.
- 어떤 두 도시도 하나 또는 둘 이상의 도로를 이용하여 오갈 수 있다.
- $1 \le W[i] \le 10^9$.
- $1 \le Q \le 200\,000$.
- $0 \le X[j] < Y[j] < N$.

Subtask 1 (6 points)

- 각 도시는 최대 2개의 도로의 끝점이다.

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)

- 추가적인 제약 조건이 없다.

샘플 그레이더

샘플 그레이더는 입력을 다음 양식으로 읽는다.

```
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]
```

샘플 그레이더는 `getMinimumFuelCapacity` 함수를 호출할 때마다, 이 함수의 리턴값을 출력한다.