Problem B - Piltover Towers

Description

You are working on a project for the Piltover Academy of Science, where you are trying to help Professor Heimerdinger with his new invention: the H-29 Towers.

He has been working on his new project for months, and he wants to make sure they are perfect before he uses them in the city of Piltover. Depending on the location, the towers can be used for different purposes, such as communication, surveillance, or even as a weapon. Professor Heimerdinger is very excited about this project and wants to make sure everything is perfect before he presents it to the council.

He has been working on a new algorithm to find the best way to connect the towers, but he is having some trouble with it. He has a graph with N towers (nodes) and E edges, where each edge has a cost associated with it. He needs to find a **tree** that connects a set of M **mandatory towers** in order to **minimise the highest cost of the edges** used in the tree. The mandatory towers need to be present in the tree while the other towers are optional. Consider the following example:

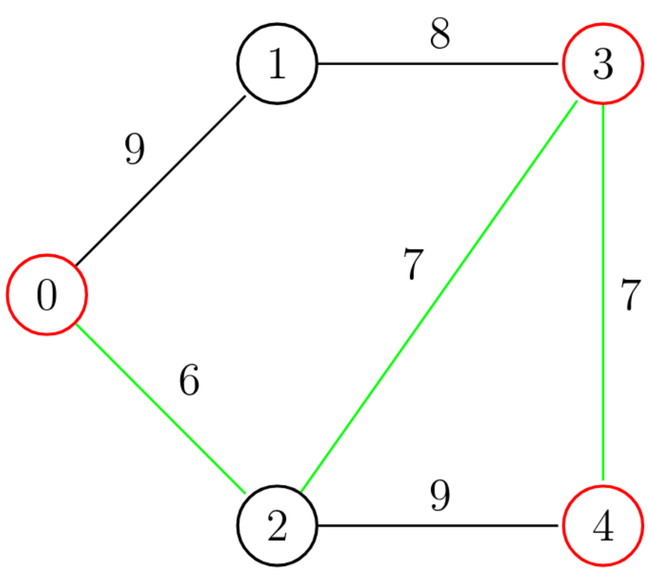

Figure 1 - Graph example.

There are 5 towers and 6 edges. The edges are represented by the lines connecting the towers, and the cost of each edge is represented by the number next to it. The mandatory towers are 0, 3, and 4 (in red). The optimal solution is 7, since the used edges (in green) are 0-2, 2-3, and 3-4 and the highest cost is 7 (edges 2-3 and 3-4). Tower 1 is not used since edges 0-2, 2-3 and 3-4 are enough to connect the mandatory towers.

Input

Each input block starts with a line containing the N towers and E edges separated by a whitespace. Then, E lines follow, each representing a connection between two towers and the cost of using that edge. Afterwards, there is a line with the number of mandatory towers M and M lines with the mandatory towers.

Output

For each test case, you should print the **highest edge value** required to ensure connection between the mandatory towers as a tree or **Impossible to connect!** if it is not possible to generate a tree between all the mandatory nodes.

Para pré processamento podemos tentar descobrir se existem subdivisões e se os nodes obrigatórios se dividem em ambos

---

Constraints

1 <= N <= 1000
1 <= E <= 500000
2 <= M <= N

Example

Input:

```
5 6
0 1 5
0 2 6
1 3 8
2 3 7
2 4 9
3 4 7
3
0
3
4
7 10
0 1 2
0 2 5
1 2 3
1 3 7
2 4 6
3 4 4
3 5 9
4 5 8
4 6 10
5 6 11
2
0
3
7 8
0 1 2
0 2 5
1 2 3
3 4 4
3 5 9
4 5 8
4 6 10
5 6 11
2
0
3
```

Output:

```
7
6
Impossible to connect!
```

Explanation

There are 3 test cases. In the first test case, the solution is presented in Figure 1.

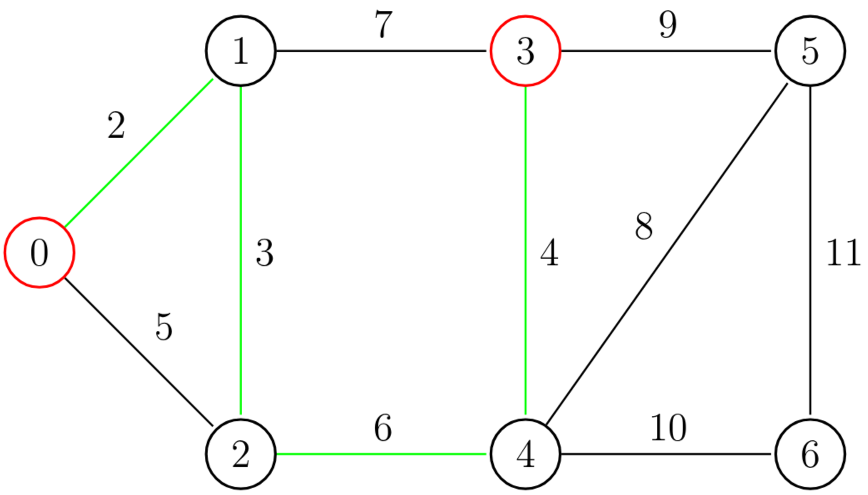

Figure 2 - Second test case.

In the second test case (Figure 2), there are 7 towers, 10 edges, and 2 mandatory towers (0 and 3, in red). The optimal solution is 6, since the used edges (in green) are 0-1, 1-2, 2-4, and 3-4. Towers 5 and 6 do not need to be used.

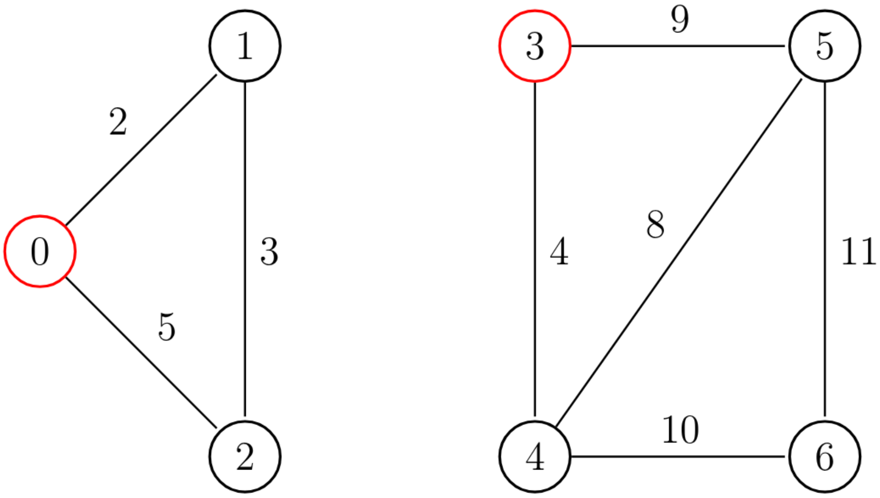

Figure 3 - Third test case.

In the third test case (Figure 3), there are 7 towers, 8 edges, and 2 mandatory towers (0 and 3, in red). Since tower 0 cannot be connected to tower 3, it is impossible to generate a tree between mandatory towers. The output is "Impossible to connect!".