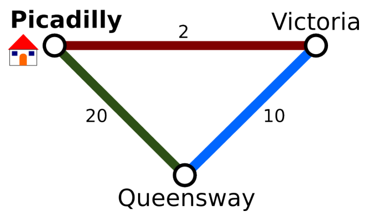
G
Expensive subway

`subway.{c,cpp,java}`

Peter lives in *Expensive City*, one of the most expensive cities in the world. Peter has not got enough money to buy a car, and the buses in *Expensive City* are pretty bad, so he uses the subway to go to work. Up to now, the subway was very cheap: you could travel anywhere with just one \$2 ticket. Last month, the managers decided that it was too cheap so they invented the EFS (Expensive Fare System). With this system, users can only buy monthly tickets between adjacent stations, which allows them to move between these stations any number of times. The price of the monthly ticket varies between stations, so the decision of which tickets to buy must be taken carefully.



With the previous subway plan, the cheapest way to travel from Picadilly to Victoria and Queensway was to buy the monthly ticket Picadilly-Victoria and Queensway-Victoria, for a total cost of \$12.

Peter is a salesperson, so he needs to be able to travel to any part of the city. He wants to spend as little money as possible, and here is where you come into the picture. He has hired you to write a program that, given the list of stations, the fares of the monthly tickets between pairs of stations and the station nearest Peter's home, returns the minimum amount of money Peter has to spend in order to travel to any other station. This program also has to return value if it is not possible to go from Peter's home station to all the rest, because in this case Peter will begin to consider using buses...

Input Description

The input consists of several test cases. A test case begins with a line containing two integers: $1 \leq s \leq 400$ (the number of stations) and $0 \leq c \leq 79800$ (the number of connections) separated by a single space. This is followed by s lines, each one containing the name of a subway station. These names will be strings of characters (uppercase or lowercase) without punctuation marks or whitespace characters, and with a maximum length of 10 characters. After the names of the stations there will be c lines showing the connections between stations. A connection allows people to travel from one station to the other in both directions. Each connection is represented as two strings indicating the names of the stations and a positive integer indicating the cost of the monthly ticket, all of which are separated by single spaces. All names of stations appearing in the connections will have previously appeared in the list of s stations. The connections will all be different, and there will not be any connection from a station to itself. The test case will end with a line containing the name of the station from which Peter needs to travel to all the other stations.

The input finishes with the phantom test case `0 0`, which must not be processed.

Output Description

For every test case, the output will be a line containing an integer, the minimum monthly price that Peter has pay to travel from the given station to all the others, or `Impossible` if it is not possible to travel to all the stations.

Sample Input

```
3 3
Picadilly
Victoria
Queensway
Picadilly Victoria 2
Queensway Victoria 10
Queensway Picadilly 20
Picadilly
4 2
Picadilly
Victoria
Queensway
Temple
Picadilly Victoria 2
Temple Queensway 100
Temple
0 0
```

Sample Output

```
12
Impossible
```