CSE221
Lab Assignment 06
SUMMER 2023

Submission Guidelines :

1. You can code all of them either in Python, CPP, or Java. But you should choose a specific language for all tasks.
2. For each task write separate python files like task2.py, task3.py, and so on. For problems that have subproblems, name those like task1A.py, task1B.py, and so on.
3. Add a hand written explanation of 3-4 lines for each of your solutions and submit that as a single document.
4. For each problem, take input from files called **"inputX_Y.txt"** and output at **"outputX_Y.txt"**, where X is the task number and Y is the sample i/o number. For example, for problem 2 sample 1, the input file is this, "input2_1.txt". For problems that have subproblems, name the files like "input1a_1.txt", "output1a_1.txt" and so on. Same for output.
5. For each task include at least one input file (if any) in the submission folder.
6. Finally zip all the files and rename this zip file as per this format:**LabSectionNo_ID_CSE221LabAssignmentNo_Summer2023.zip** [Example:**LabSection01_21101XXX_CSE221LabAssignment06_Summer2023.zip**]
7. Don't copy from your friends.
8. You MUST follow all the guidelines, naming/file/zipping convention stated above.
   Failure to follow instructions will result in a straight 50% mark deduction.

A useful tool for making graphs:

https://csacademy.com/app/graph_editor/

Task 01:

You are given a weighted, directed graph with N nodes and M edges. Each edge is represented as a triple (u, v, w), where u and v are the nodes connected by the edge and w is the weight of the edge.

Your task is to find the shortest path from a source node S to all other nodes in the graph using Dijkstra's algorithm. You should output the shortest distance from the source node to each of the other nodes in the graph. If a node is unreachable from the source node, its distance should be represented as -1.

Input

The first line of the input contains two integers, N and M (1 <= N <= 1000, 1 <= M <= 100000) denoting the number of nodes and edges in the graph, respectively.

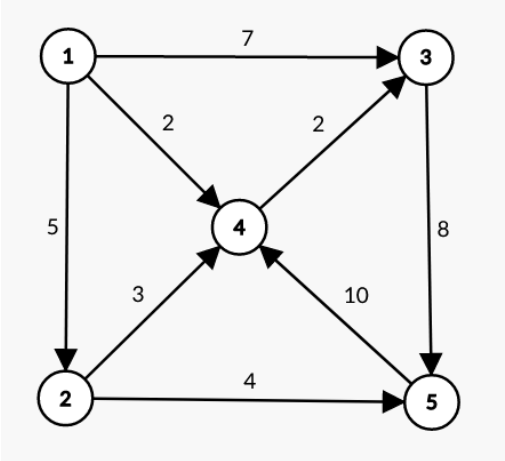
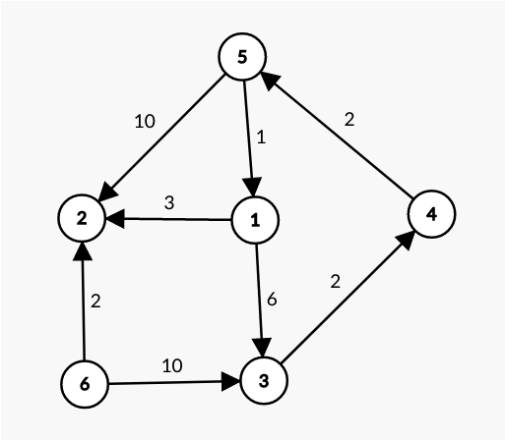
The next M lines each contain three integers, u, v (1 <= u, v <= N), and w (1 <= w <= 100) denoting an edge from node u to node v with weight w.

The last line of the input contains an integer S (1 <= S <= N) denoting the source node.

Output

Output N space-separated integers, where the i-th integer represents the shortest distance from the source node to node i. If a node is not reachable from the source node, output -1 instead.

Sample Input 1	Sample Output 1	Sample Graph 1

5 8 1 2 5 1 3 7 1 4 2 2 4 3 2 5 4 3 5 8 4 3 2 5 4 10 1	0 5 4 2 9	
Sample Input 2	Sample Output 2	Sample Graph 2
6 8 1 2 3 1 3 6 3 4 2 4 5 2 5 2 10 5 1 1 6 2 2 6 3 10 3	5 8 0 2 4 -1	

Explanation of Sample Input 1

Using Dijkstra's algorithm, the shortest path from node 1 to all other nodes in the given graph is:

- Distance from node 1 to node 1 is 0.
- Distance from node 1 to node 2 is 5.
- Distance from node 1 to node 3 is 4.
- Distance from node 1 to node 4 is 2.
- Distance from node 1 to node 5 is 9.

Task 02:

Alice and Bob are in a hurry to meet each other and have to traverse through a directed graph with weighted edges. Alice starts from node S and Bob starts from node T. They want to find a common node in the graph, where they can meet each other in the minimum amount of time. Alice or Bob can wait at any node if they want to.

Input

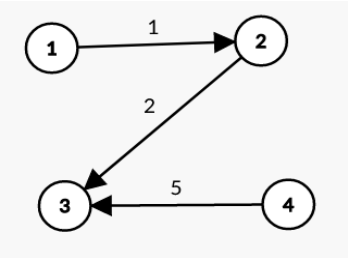
The first line contains two integers N (1 ≤ N ≤ 10000) and M (1 ≤ M ≤ 100000) separated by a space, denoting the number of nodes and edges in the graph, respectively.

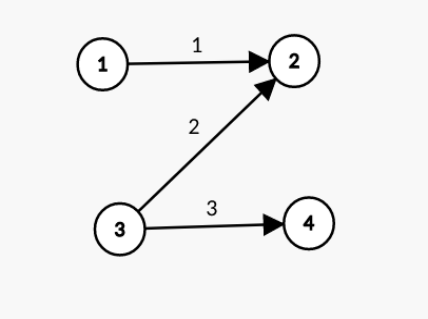
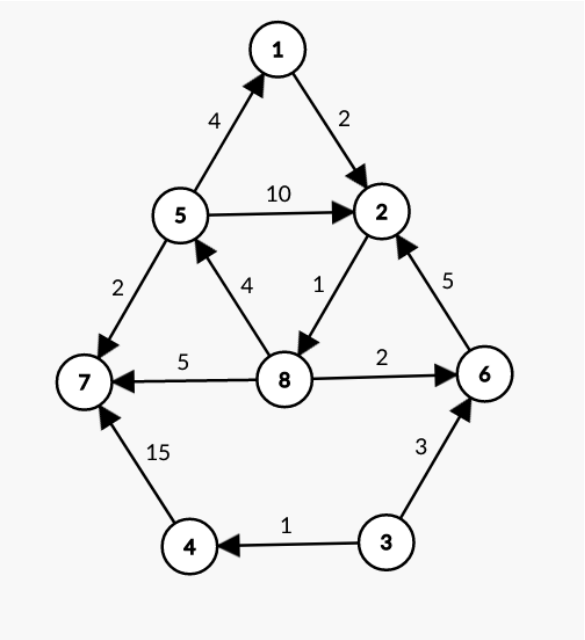
The next M lines each contain three integers, u, v (1 <= u, v <= N), and w (1 <= w <= 100) denoting an edge from node u to node v with weight w.

The last line contains two integers S,T (1 <= S,T <= N) which denotes the starting node of Alice and the starting node of Bob respectively.

Output

Output two integers separated by a space. The first integer is the minimum time it takes for Alice and Bob to meet. The second integer is the vertex where they will meet. If there is no such node, print "Impossible".

Sample Input 1	Sample Output 1	Sample Graph 1
4 3 1 2 1 2 3 2 4 3 5 1 4	Time 5 Node 3	
Sample Input 2	Sample Output 2	Sample Graph 2

4 3 1 2 1 3 2 2 3 4 3 1 4	Impossible	
Sample Input 3	Sample Output 3	Sample Graph 3
8 12 3 6 3 3 4 1 6 2 5 4 7 15 2 8 1 8 6 2 8 7 5 8 5 4 5 2 10 5 1 4 1 2 2 5 7 2 3 5	Time 8 Node 2	

Task 03:

You are traveling through dangerous wilderness with bridges and rivers. Each of them has a danger level that represents the risk of injury if you attempt to cross it.

Your goal is to find the safest path from your current location to your destination node **N**. You will start your journey from node **1**.

You define the danger level of a path as the maximum danger level of all the edges along the path. To maximize your chances of survival, you decide to take the safest path possible. The safest path is the one with the minimum danger level among all the paths from node 1 to node N.

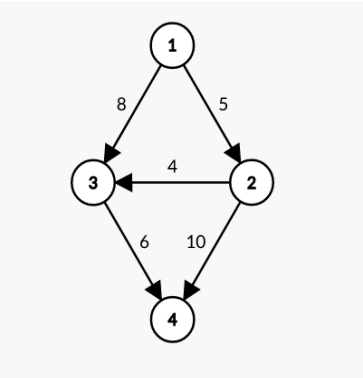
Input

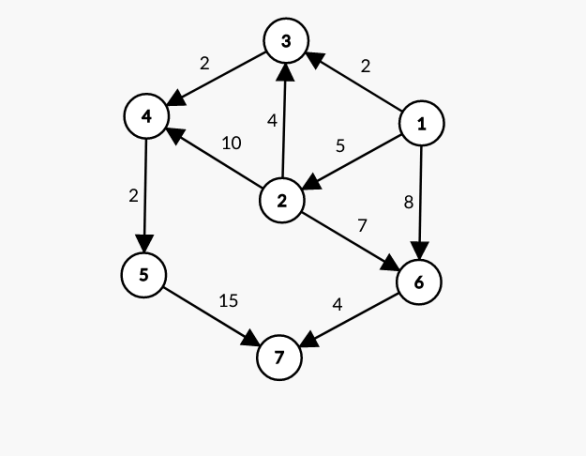
The first line of the input contains two integers, N and M (1 <= N <= 1000, 1 <= M <= 100000) denoting the number of nodes and edges in the graph respectively.

The next M lines each contain three integers, u, v (1 <= u, v <= N), and w (1 <= w <= 100) denoting an edge from node u to node v with weight w.

Output

Find a path with the minimum danger level from source node 1 to destination node N. Print "Impossible" if there is no path from node 1 to N.

Sample Input 1	Sample Output 1	Sample Graph 1
4 5 1 2 5 2 3 4 3 4 6 1 3 8 2 4 10	6	
Sample Input 2	Sample Output 2	Sample Graph 2

7 10 1 2 5 2 3 4 3 4 2 1 3 2 2 4 10 4 5 2 5 7 15 2 6 7 1 6 8 6 7 4	7	

Explanation of Sample Input 1

There are three paths from node 1 to node 4.

a) 1 → 3 → 4
   In path (a), the danger level is maximum(8,6) = 8

b) 1 → 2 → 4
   In path (b), the danger level is maximum(5,10) = 10

c) 1 → 2 → 3 → 4
   In path (c), the danger level is maximum(5,4,6) = 6

Among all the three paths, the minimum danger level is 6 by following the path 1 → 2 → 3 → 4.