CSE 208 | Offline 2 | Single Source Shortest Path

Instructions:

(1) * Please DO NOT COPY solutions from anywhere (your friends, seniors, internet etc.). Any form of plagiarism (irrespective of source or destination), will result in getting -100% marks in the online/offline.

(2) Deadline: 11 December, 11:59 PM

(3) Rename all the problem solutions according to your student ID. If your ID is 2105XXX, then create a folder named 2105XXX. Afterward, rename problem 1 as 2105XXX_problem1.cpp, and similarly, rename the others. Next, move all the solutions inside the folder. Create a zip file of that folder. Lastly, submit the zip file.

(4) You get 10 marks for each right answer. A viva will also be conducted. If the teacher finds that you don't know how to implement it, you'll get a score of 0.

Problem 1

Your country has N ($N \leq 100$) cities and M ($M \leq 1000$) roads between them. You want to go from city S to city D. Your car's gas tank has a capacity of C ($C \leq 100$) liters and you need 1 liter of gas to travel per unit distance. Currently you have no gas. And you can only refill at the cities, there are no gas stations on the roads. But the price of gas varies in each city. Find the minimum amount of money you will need for your trip.

Hint: You can try to build a graph with the gas prices on the edges. Detailed hints on the last page

Input

The first line of input contains three integers N, M and C – denoting respectively the number of cities, the number of roads and your gas tank's capacity.

The next line contains N space separated integers – the price of gas (per liter) in each city.

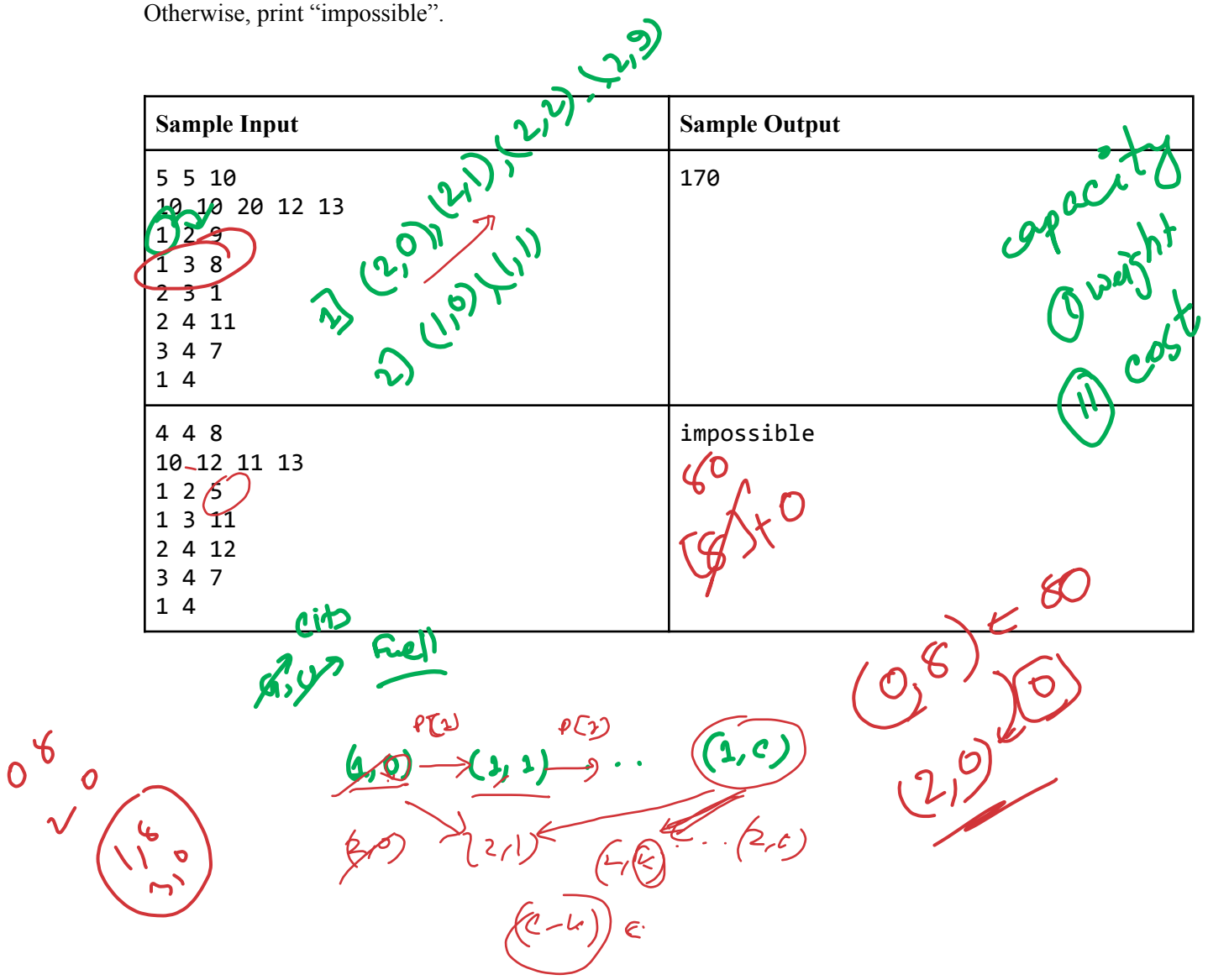
The next M lines contain three integers u, v, d ($1 \leq u, v \leq N$, $1 \leq d \leq 100$) – denoting that there is a road between nodes u and v with length d.

The next line contains two integers S and D – the starting and the destination city.

Output

If it is possible to complete this trip, print a single integer – the minimum amount of money required. Otherwise, print "impossible".

Sample Input	Sample Output
5 5 10 10 10 20 12 13 1 2 8 1 3 8 2 3 1 2 4 11 3 4 7 1 4	170
4 4 8 10 12 11 13 1 2 5 1 3 11 2 4 12 3 4 7 1 4	impossible



Sample Input	Sample Output
5 5 10 1 10 2 12 13 1 2 9 1 3 8 2 3 1 2 4 11 3 4 7 1 4	20
5 5 10 1 10 2 12 13 1 2 9 1 3 8 2 3 1 2 4 1 3 4 7 1 4	10
4 5 9 5 3 6 2 1 2 1 2 3 3 1 3 5 3 4 1 1 4 6 1 4	17

Problem 2

You are given a directed graph with N ($N \leq 100$) nodes and M ($M \leq 1000$) edges with possibly negative weights, but no negative cycle. You are given the power to add a directed edge from node A to node B. You can assign any weight between L ($-100 \leq L \leq 100$) and H ($L \leq H \leq 100$) for this edge. Assign an appropriate weight for this edge so that the distance from node S to node D is minimized and no negative cycle is introduced in the graph.

Hint: Use bellman-ford to check from L to H.

Bonus 5 marks if you can do it by running the bellman-ford algorithm only once.

Input

The first line of input contains two integers N and M – denoting respectively the number of nodes and the number of edges in the graph.

The next M lines contain three integers $u,\ v,\ w$ ($1 \leq u,\ v \leq N,\ -100 \leq w \leq 100$) – denoting that there is an edge from node u to node v with weight d.

The next line contains two integers A and B – you can add an edge from node A to node B.

The next line contains two integers L and H – the range (inclusive) of weights for the edge.

The next line contains two integers S and D – the source and the destination node.

Output

Print two numbers on a single line – the weight you will assign to the edge and the minimum distance from S to D after adding it.

If it is impossible to change the distance from S to D **following the rules**, print "impossible" on a single line.

Sample Input	**Sample Output**
4 4 1 2 10 2 3 10 1 4 20 4 2 30 4 3 -50 50 1 3	-50 -30
5 6 1 2 1 2 3 2 3 5 1 3 4 2 4 5 5 1 4 3 4 2 -10 5 1 5	-4 2

Sample Input	Sample Output
4 4 1 2 10 2 3 10 1 4 20 4 2 30 4 3 1 20 1 3	impossible
4 4 1 2 10 2 3 12 3 4 -5 4 1 15 2 4 -40 -30 1 4	impossible
4 4 1 2 10 2 3 12 3 4 -5 4 1 -5 2 4 5 10 1 4	5 15
4 4 1 2 10 2 3 12 3 4 -5 4 1 -5 2 4 -10 10 1 4	-5 5

Hints for Problem 1

1. We want to minimize the cost of the trip. So if we can build a graph, where the weights of the edges correspond to your cost (of buying gas), then applying any SSSP algorithm on it should solve the problem. Since the weights (costs) can only be non-negative numbers, we can run Dijkstra's algorithm. Now, how to build the graph?
2. How can we describe the state of your car at any time? By the city where it's at and by how much gas it has left.
3. So build a graph where the nodes correspond to the states of your car. Say when the car is at city U and has gas G left, the state is $(U,\ G)$, the node will also be $(U,\ G)$.
4. Now from any state (or node) we have two choices:
    a. Buy 1 liter of gas: If the price of gas is P at city U, going from state $(U,\ G)$ to state $(U,\ G+1)$ will cost P. So the edge from $(U,\ G)$ to $(U,\ G+1)$ will have weight P.
    b. Or go to an adjacent city: If city V is D distance away from city U, our state changes from $(U,\ G)$ to $(V,\ G-D)$ with cost 0. Since we're not buying gas on the road.
5. Build this graph and apply Dijkstra.