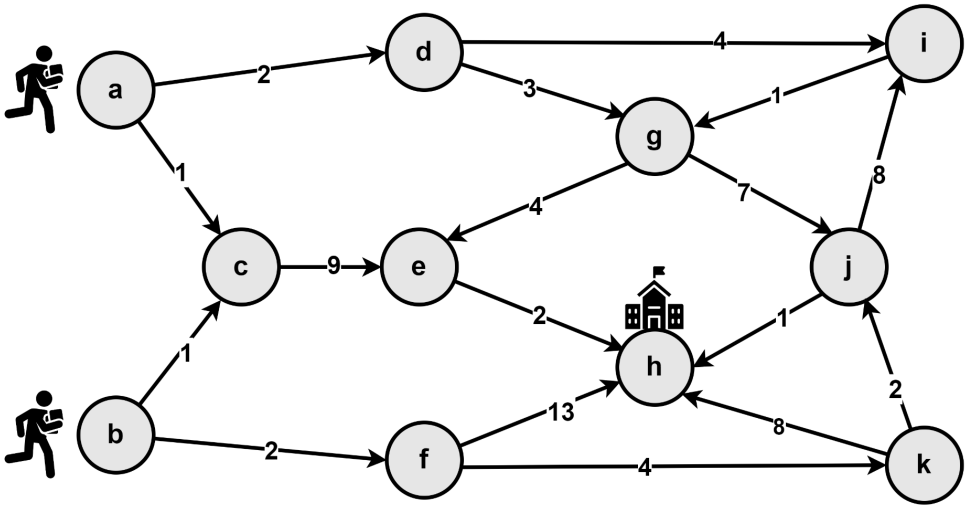
Fall 2023
CSE 221 Lab Final
Set A
Total Marks: 20
Time: 1 hour

There are two friends named Stewie and Brian; they go to the same school. One day, they decided to have a friendly competition to see who would get to the school first if they started their journey from their nearest stoppage. But Stewie is an intelligent kid, and he came to you with the city map because he knows you are doing an Algorithm course. He asked you to run an appropriate algorithm on the city map and tell him who will win the race based on the information available on the map about the stoppages and the time each stoppage will take.



You need to help Stewie to find who will arrive at the school first(the one with the lowest travel time). The person who will take less time to reach the school will be the winner.

Q1 (Marks 2): Read the graph inputs from a text (.txt) file following the given format.
Q2 (Marks 4): Show inputs (which you have taken from the text file) in the adjacency matrix or adjacency list to represent the graph.
Q3 (Marks 9): Apply a suitable algorithm by which you can find the minimum time Brian will take to reach the school, and Stewie will take to reach the school and then declare the winner between them. The algorithm should terminate once the minimum travel time to the school is found.

Input:

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

The next M lines each contain two characters(the nodes) and one integer(weight of the edge between the nodes).

The last line will contain three characters S1, S2 and D separated by space, denoting from where Stewie will start the journey, from where Brian will start the journey, and the stoppage of the school, respectively.

Output:

1st line: The smallest travel time Stewie will take to reach school.

2nd line: The smallest travel time Brian will take to reach school.

3rd line: The stoppage node from where the winner started the journey.

And if anyone can not reach the school, then output ‘IMPOSSIBLE’ for them, and if both fail to reach the school, then output ‘NO ONE CAN WIN’. If both of them reach the school at the same time, output ‘DRAW’. You are not allowed to change the inputs of the graph.

Sample Input	Sample Output
11 18 a*d*2 a*c*1 d*i*4 d*g*3 c*e*9 b*c*1 b*f*2 e*h*2 i*g*1 g*e*4 g*j*7 f*h*13 f*k*4 k*h*8 k*j*2 j*h*1 j*i*8 a b h	11 9 b

Q4 (Marks 5): There is construction work going on in the city, and due to that, the roads between ‘d’ and ‘g’ and between ‘k’ and ‘j’ and ‘b’ and ‘c’ are closed. Now, your task is to find the minimum time Stewie will take to reach the school without using those k->j and d->g and b->c edges, and the same for Brian. And based on that find the winner. You are not allowed to change the inputs of the graph.

Input:

For this task you will take input from the previous **task’s[Q3] input file** and in addition take **another input** from the below Sample input.

Here, the First line contains an integer R, denoting the number of roads that are blocked. Following R lines consist of the closed roads that means the edges that can not be used.

Output:

Newly calculated distances avoiding the blocked roads.

1st line: The smallest travel time Stewie will take to reach school.

2nd line: The smallest travel time Brian will take to reach school.

3rd line: The stoppage node from where the winner started the journey.

Sample Input	Sample Output
3 b->c d->g k->j	12 14 a

Lab5 1

```
f1= open("input1(a)_1.txt", 'r')
f2= open("output1(a)_1.txt", 'w')
N, M= map(int, f1.readline().split())
Prerequisite_List= [tuple(map(int, f1.readline().split())) for _ in range(M)]

def DFS(Course, Graph, Visited, Stack):
   Visited[Course]= True
   for Neighbor in Graph[Course]:
       if not Visited[Neighbor]:
           DFS(Neighbor, Graph, Visited, Stack)
   Stack.append(Course)

def DFS_Order(N, Prerequisite_List):
   Graph= {i: [] for i in range(1, N+1)}
   for u, v in Prerequisite_List:
       Graph[u].append(v)

   Visited= [0] * (N+1)
   Stack= []

   for i in range(1, N+1):
       if not Visited[i]:
           DFS(i, Graph, Visited, Stack)

   if len(Stack)== N:
       return Stack[::-1]
   else:
       None

Order= DFS_Order(N, Prerequisite_List)
if Order== "IMPOSSIBLE":
   print(Order, file= f2)
else:
   print(*Order, file= f2)

f1.close()
f2.close()
```