Lab Assignment - 04

Submission Link: https://forms.gle/ewki67FvStCp77Tf7

Submission Guidelines:

1. You can code all of them either in Python or Java or any language you want. But you should choose 1 language for all tasks.
2. For **each task** write separate python files like task1.py, task2.py and so on. For **task 3**, you can add a pdf file to answer the question.
3. For each problem, you must take input from files called "inputX.txt", and output at "outputX.txt", where X is the task number. So, for problem 1, the input file name should be- "input1.txt" & the output file name should be - “output1.txt” and so on.
4. For each task, include the input files (if any) in the submission folder.
5. All files **MUST** be put in a folder. The folder **MUST** be named following the format: **sec_ID_labNo**. Zip this folder. This will result in a zip format of **sec_ID_labNo.zip**. (Example: 2_20101234_1.zip). Submit this zip file in the above mentioned submission link.
6. Don't copy from anyone. Both the source and the destination will be punished irrespectively.
7. You **MUST** follow all the guidelines, naming/file/zipping convention stated above. ***Failure to follow instructions will result in straight 50% mark deduction.***

Problems Description:

The following tasks need to be solved using Shortest path algorithms.In graph theory, the shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized. You have learned several shortest path algorithms such as Dijkstra’s algorithm, Bellman-Ford algorithm etc. The problems below are related or similar to some of the shortest path algorithms you learned in class.

Read the task descriptions carefully and implement the algorithm using either Java or Python or any language you are comfortable with. The output format **MUST** match exactly as shown in each task.

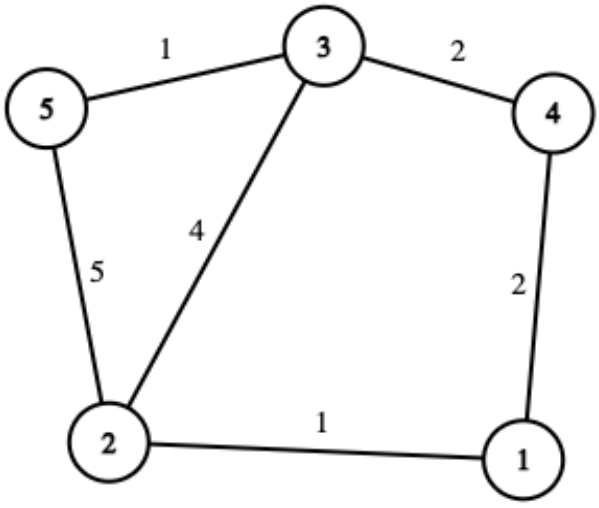
TASK 1 [5 Marks]

Oh No! Wall Maria is Breached! Titans (human eating creatures) are coming. Eren has to reach his home as soon as possible to save his mother.

For simplicity, let's think of Eren's hometown map containing **N** places and **M** roads. A road is a connection between two places.Here, each place is represented by a unique number between **1** and **N**.

Suppose, each road is represented by i and each i^{th} road connects 2 places $\mathbf{u_i}$ **and** $\mathbf{v_i}$ (u_i and v_i are the unique number of the places) and this road is overrun by $\mathbf{w_i}$ titans. You can assume Eren's current position as **1** and Eren's home as **N**. Now help Eren to find a path so that he has to face as minimum number of titans as possible.

For example, in the following graph (vertices represent places and edges represent roads), Eren has to face a minimum 5 titans by using the path 1 → 4 → 3 → 5.



Input Format:

First line contains a number **T** which represents the number of test cases.

For each test case:

First line will contain 2 numbers **N, M** representing the number of places and roads respectively.

Each of the following **M** lines will contain 3 numbers ($\mathbf{u_i}$ $\mathbf{v_i}$ $\mathbf{w_i}$) representing one of the roads that connects places $\mathbf{u_i}$, $\mathbf{v_i}$ and is guarded by $\mathbf{w_i}$ titans.

Output Format:

For each test case, print one number which will be the minimum number of titans Eren has to face to reach his home. It is ensured that there exists roads connecting Eren's current position to his home.

Sample Input:

```
3
1 0
2 1
1 2 10
5 6
3 5 1
1 2 1
2 3 4
1 4 2
4 3 2
2 5 5
```

Sample Output:

```
0
10
5
```

Sample Description:

Here we have 3 test cases.

In the 1st case, there is 1 place and 0 roads. So, Eren is already at home, i.e. he has to face 0 titans.

In the 2nd case, there are 2 places and 1 road connecting them which is guarded by 1 10 titans. So, Eren has to face 10 titans to reach home.

In the 3rd case, the example graph shown before is given. For that, it was shown that Eren has to face at least 5 titans.

Hint 1:

You can use the following pseudocode of Dijkstra's algorithm.

```
function Dijkstra(Graph, source):
    dist[source] ← 0                          // Initialization
    create a priority queue Q for the vertices        // min-heap
    create a list visited for the vertices
    for each vertex v in Graph:
        if v ≠ source:
            dist[v] ← ∞                       // Unknown distance from source to v
            prev[v] ← NULL                    // Predecessor of v
        add v to Q with priority value dist[v]
        visited[v] ← False
    while Q is not empty:                     // The main loop
        u ← Q.extract_min()                   // Remove and return best vertex
        if visited[u]:
            continue
        visited[u] ← True
        for each neighbor v of u:
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u
                add v to Q with priority value dist[v]
    return dist, prev
```

Note:

- What are you supposed to return for problem 1?
- Do you need to compute $prev$?

Hint 2:

Min-heap data structure is used to implement the priority queue. There are libraries that implement this.

Hint 3:

Use adjacency list representation of Graph while reading from input.

Hint 4:

Try to incorporate $\mathbf{w_i}$'s into the adjacency list.

TASK 2 [5 Marks]

Modify your code for the first problem to find the path Eren has to follow to face the minimum number of titans.

For the previous sample input, the output is:
1
1 2
1 4 3 5

Sample Description:
Here we have 3 test cases.

In the 1st case, there is 1 place and 0 roads. So, Eren is already at home, i.e. the path is: 1.

In the 2nd case, there are 2 places and 1 road connecting them which is guarded by 1 10 titans. So, the path to face minimum titans is the only existing path: 1 → 2.

In the 3rd case, the example graph shown before is given. For that, it was shown that, to face the minimum number of titans, Eren has to follow the path: 1 → 4 → 3 → 5.

TASK 3 [1.5+ 1.5 + 2 = 5 Marks]

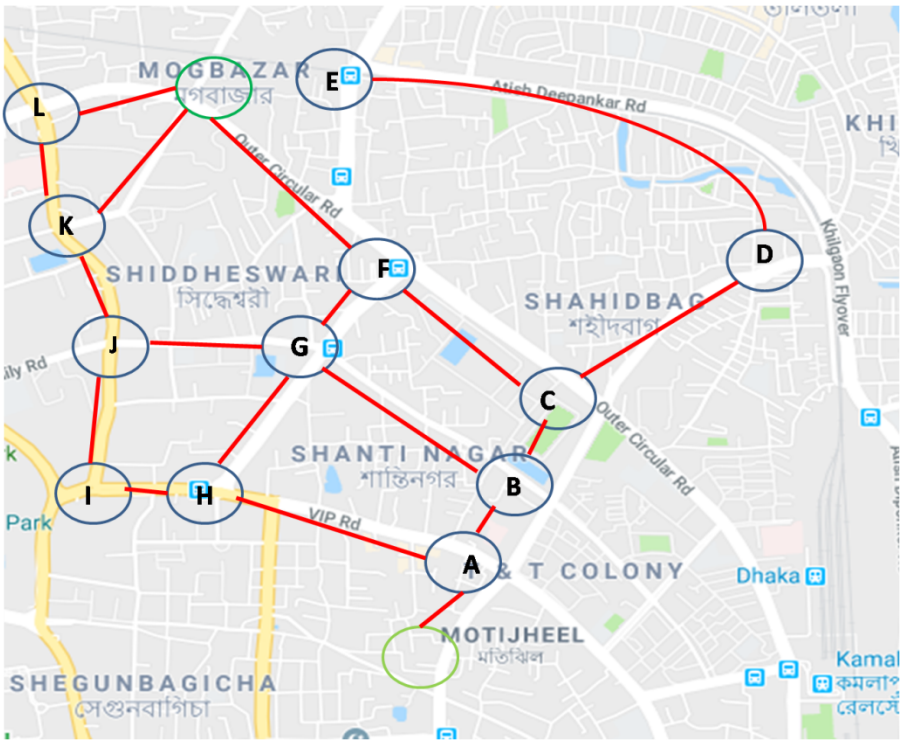
If there are N places and M roads, what are the time complexities of the solutions you provided in problem 1 & problem 2 respectively?
If the number of titans in each road is exactly 1, there is an $O(N + M)$ algorithm to solve this problem. What is it?
Note: If you can use any known algorithm, write its name and the inputs that will be given to it. Otherwise, give a pseudocode of the algorithm.

TASK 4 [5 marks]
Problem Description: A portion of the map of Dhaka is given in the picture.

There are 2 mother nodes Motijheel, which is the source, and Moghbazar the destination. The other nodes from A to L represent intersections. There are multiple routes to reach from source to destination. The table below shows the weights of each route which represent the level of traffic. The higher the value, higher the traffic. Using your knowledge on graph implement Dijkstra's algorithm. Print the output.

BFS also gives the shortest path between source and destination. Why not use BFS in this situation?

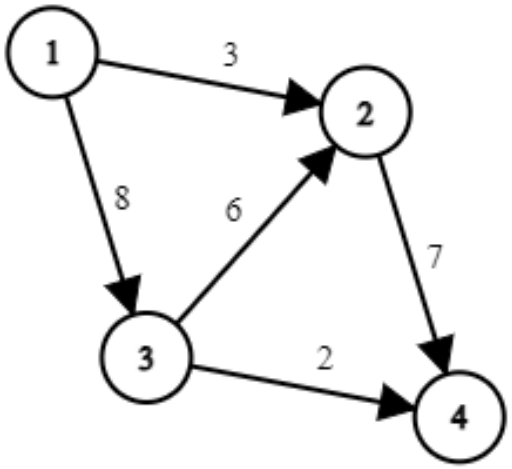
Vertex 1	Vertex 2	Traffic Level
Motijheel	A	3
A	B	4
A	H	6
B	G	2
H	I	5
B	C	7
C	F	7
C	D	3
D	E	1
F	G	2
F	MOGHBAZAR	4
G	H	3
G	J	1
I	J	7
J	K	6
K	L	4
K	MOGHBAZAR	7
L	MOGHBAZAR	2



TASK 5 [5 Marks]

In a computer network, there are **N** devices and **M** links. A link is a one way connection from one device to another. Each link has a data transfer rate **d** it can support.

The data transfer rate of a **connection** from one device to another is the minimum data transfer rate of the links in that connection. [You can think of connection as a directed path. So, a connection from device u to device v is a sequence of links that joins a sequence of distinct devices where the links are directed from u to v.]



For example, in the network above, there can be three possible connections from 1 to 4:

(i) 1 → 2 → 4 : alt = 3
(ii) 1 → 3 → 4: alt = 2>3
(iii) 1 → 3 → 2 → 4: alt = 6 >3

In connection (i), there are two links: 1 → 2 with data rate 3, 2 → 4 with data rate 7. So, data transfer rate of this connection is minimum of 3 and 7, that is 3.
Similarly, data transfer rate of connection (ii) and (iii) are 2 and 6 respectively.

Now, the maximum data transfer rate from a device **u** to a device **v** is the maximum of the data transfer rates of all possible **connections** from **u** to **v**. For example, the maximum data transfer rate from 1 to 4 that this network supports is 6 (through connection (iii)).

Now, given a computer network, you have to calculate the maximum data transfer rate this network supports from a designated *source device* to all other devices.

Input Format :

First line contains a number **T** which represents the number of test cases.

For each test case:

First line contains 2 numbers **N, M** representing the number of devices and links respectively.

Each of the following **M** lines will contain 3 numbers ($\mathbf{u_i}$ $\mathbf{v_i}$ $\mathbf{d_i}$) representing a link from device $\mathbf{u_i}$ to device $\mathbf{v_i}$ with data transfer rate $\mathbf{d_i}$.
The last line contains a number **s**, which is the *source device.*

Output Format :

For each test case, print a list of numbers, where i^{th} number represents the maximum data transfer rate from **s** to **i**.
Note: print 0 when **s == i** and print -1 if there is no connection from **s** to **i**.

Sample Input:

```
4
1 0
1
2 1
1 2 1
1
2 1
1 2 1
2
4 5
3 2 6
1 2 3
2 4 7
3 4 2
1 3 8
1
```

Sample Output:
0
0 1
-1 0
0 6 8 6

Sample Description:

Here we have 4 test cases.

In the 1st case, there is 1 device and 0 links, so there is no edge. Here the source device is 1. As the maximum data transfer rate between device 1 to 1 itself is 0, the output is 0.

In the 2nd case, there are 2 devices and 1 link. Here the source device is 1. From device 1→1 the data transfer rate is 0. The maximum data transfer rate between 1→ 2 devices is 1 (As there is only 1 data rate). So the output is 0 1.

In the 3rd case, there are 2 devices and 1 link. Here the source device is 2. From the input we can see that there is a connection between 1→ 2 with a data transfer rate of 1. but there is no connection between 2→1. So, from 2→1 the output will be -1 as noted above in the output format. And for 2→2 the output will be 0. So, the output is -1 0.

In the 4th test case, the example graph is given. Here the source device is 1. From 1→1, output is 0. From device 1 to device 2 there can be 2 possible paths, 1→2 and 1→3→2. The maximum data transfer rate for 1→2 will be 6 if we consider this path 1→3→2. Similarly, From 1→3, there is only one path and the data transfer rate is 8, so output is 8. For 1→4 the example is given in the beginning. So the output is 0 6 8 6.

Hint:

1. In the shortest path problem, you calculate path length by **summing** all edge weights in the path. Whereas in this problem you need to calculate data transfer rate which is obtained by taking the **minimum** of all edge weights in the path.
2. In the shortest path problem, you are trying to **minimize** path length, whereas in this problem you are trying to **maximize** data transfer rate.
   [In Dijsktra's algorithm, initialize $dist$ array differently and change the codes related to priority queue for maximization]

You can use the following pseudocode which is based on these hints.

function Network($Graph, source$):

$dist[source] \leftarrow \infty$ *// Initialization*

create a priority queue $Q$for the vertices *// Max-priority Queue*

for each vertex v in $Graph$:

if $v \neq source$:

$dist[v] \leftarrow -\infty$ *// Unknown data transfer rate from $source$ to v*

$prev[v] \leftarrow NULL$ *// Predecessor of v*

add v to Q with priority value $dist[v]$

while Q is not empty: *// The main loop*

$u \leftarrow Q$.extract_max() *// Remove and return best vertex*

for each neighbor v of u:

$alt \leftarrow min(dist[u], length(u, v))$

if $alt > dist[v]$:

$dist[v] \leftarrow alt$

$prev[v] \leftarrow u$

add v to Q with priority value $dist[v]$

return $dist, prev$

Note: The libraries for priority queue may only implement min-heap data structure. But for this problem, we need a max-heap.