Dept. of Computer Science and Engineering
CS69011: Computing Lab 1

Assignment 2 — Maximum Marks: 30

Consider a system designed to monitor the spread of a viral infection in a population. Let $X = \{x_1, \dots, x_n\}$ be a set of people in the area. Some of these people are tested positive with the viral infection, at different points in time. Also, some of these people come in contact with other people. This relationship is described using a "contact graph", which is a weighted graph, $G(X, E, W)$, with people as the vertices. An edge $e_{ij} = (x_i, x_j)$ of the contact graph G denotes that there is contact between persons x_i and x_j. We assume that the contact graph has no timestamps, implying that there is contact between the people at all times or never. The weights w_{ij} denote the inverse of strength of contact, or the distance between (x_i, x_j). A larger distance between two persons indicate a lower probability of one person infecting another. Further, we can define the distance between two persons who are not directly connected in the contact graph as the total length of the shortest path between them. This simulates infection through a carrier person.

We are interested in finding people who caused a particular infection event, also called the *source*, and the persons who were infected, also called the *targets*.

Part 1: *Top-k Target Identification* (10 marks)

The first task is to write a program given a contact graph and a source person, find the top k most probable targets. Since the infection probability is a decreasing function of distance, this problem is the same as finding the top k people with the lowest distance in the contact graph. You can assume that the weights of the contact graph are positive and use a modified version of Dijkstra's algorithm (https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm) to find the top-k closest targets.

Input format:
The input will be given in a file of the format:

$n, m, source, k$
x_{i_1}, x_{j_1}, w_1
...
x_{i_m}, x_{j_m}, w_m

Here *n* is the number of people, *m* is the number of edges in the contact graph, *source* is the index of the source node, (x_{i_k}, x_{j_k}) are the vertices of the k^{th} edge and w_k is the corresponding weight.

Output format:
Print the indices of the top *k* target nodes in stdout.

Part 2: *Top-k Source identification* (10 marks)

In the second part, along with the contact graph, you are also given a list of nodes with their infection times (called the ***infection list***) and a specific ***target*** node whose infection time is also known. You must find the top-k closest nodes which are potential sources for the current target node. A source must have an infection time which is earlier than the infection time of the target.

Important note: All nodes on the path to a source node need not be on the infection list. You can use a modified Dijkstra's algorithm for this problem.

Input format:
The input will be given in a file of the format:

$n, m, target, k$

x_{i_1}, x_{j_1}, w_1

...

x_{i_m}, x_{j_m}, w_m

o

i_1, t_1

...

i_0, t_o

Here *n* is the number of people, *m* is the number of edges in the contact graph, *target* is the index of the target node, (x_{i_k}, x_{j_k}) are the vertices of the k^{th} edge and w_k is the corresponding weight. o is the number of infected nodes, and i_j, t_j are the indices and infection times of j^{th} infection.

Output format:
Print the indices of the top *k* source nodes to stdout.

Part 3: ***Possible infection recommendation*** (10 marks)

In this part, you are given the contact graph and the infection list. Your task is to find all nodes which are possibly infected in the contact graph. These are nodes which are on the shortest paths from a source to a target node, and are themselves not infected.

You need to find all possible source nodes to all infected nodes (recall that source node must be infected before the target node). All, nodes on the shortest paths from sources to targets are *possibly infected nodes*. List all possibly infected nodes for a given input contact graph and infection list.

Hint: Since you must find all pair shortest paths, you can use a variant of the Floyd-Warshall algorithm (https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm).

Input format:
The input will be given in a file of the format:

$n, m,$

x_{i_1}, x_{j_1}, w_1

...

x_{i_m}, x_{j_m}, w_m

o

i_1, t_1

...

i_0, t_o

Here *n* is the number of people, *m* is the number of edges in the contact graph, (x_{i_k}, x_{j_k}) are the vertices of the k^{th} edge and w_k is the corresponding weight. o is the number of infected nodes, and i_j, t_j are the indices and infection times of j^{th} infection.

Output format:
Print the indices of the possibly infected nodes to stdout.