Assignment 2 - Q1 Report

Code Explanation

We take input arguments as N- number of vertices, adjacency list with each element as a pair of distance of the edge and destination edge, and the source vertex.

Then, define a priority queue for edges in ascending order sense with the lowest distance at the top. We will add adjacent edges to the priority queue and the total distance from the source to the adjacent vertex.

Then, we make an array to update the distances of all vertices from the source. And initialise all distances as infinity.

We will repeatedly add and remove edges to the priority queue until all the vertices are traversed and the shortest distance is found(i.e. no more edges can be added to the priority queue and the shortest distance is found.

We start the graph traversal by initialising the current vertex as the source vertex.

We will take an element of a priority queue as our current vertex and then check all the adjacent vertices from the adjacency list. If
(distance of the current vertex from source) + (distance of the adjacent vertex from current vertex) < (distance of the vertex from the source stored in dist array),
We will update the shortest distance in that array. Then, we will add these adjacent vertices with their distances to the priority queue.

In the next iteration of the priority queue, the vertex with the least distance is chosen, and the current vertex is changed to it.

In this way, the whole graph is traversed, and the shortest distance between each vertex from the source is obtained.

Time complexity

in the nested loop, all the edges are traversed, and the vertex with a shorter distance is pushed into the priority queue. At max, all the vertices will be put in the priority queue, so this pushing operation takes O(Log(n)) time. While all edges are traversed, the total time complexity is O(m. Log(n)) where m = number of edges