ITCS 6114/8114: Algorithms and Data Structures
Programming Project 1: Shortest Paths in a Network

Name: Rucha Pramod Visal
Student ID: 801257824
Email: rvisal@uncc.edu

Introduction:

A network that follows the OSPF (Open Shortest Path First) protocol routes packets using Dijkstra's shortest path algorithm. The criteria used to compute the weight corresponding to a link can include the time taken for data transmission, reliability of the link, transmission cost, and available bandwidth. Typically each router has a complete representation of the network graph and associated information available to it.

The shortest time path minimizes the sum of the transmission times of the links along the path. The network topology can change dynamically based on the state of the links and the routers. For example, a link may go down when the corresponding cable is cut, and a vertex may go down when the corresponding router crashes. In addition to these transient changes, changes to a network occur when a link is added or removed.

Program Description:

Programming Language : Python
Compiler Version: Python 3.9.9 64-bit
OS: Windows X System

Run the Program:

- Varies according to the IDE/Environment.
- If you compile and run the program using command prompt, go the the file destination and execute "py -3.9 FileName.py"

Data Structure:

1. Priority Queue data structure is used to implement the algorithm and have design
   Dijkstra's Shortest Path Algorithm.
2. Dictionary data structure is used to store information about the vertex.
3. Dictionary data structure is used to add all the edges of the graph.

4. Queue data structure stores the visited nodes [Sorted sort, no duplicate values]. It sorts the vertex in alphabetical order useful while printing the graph.

Implementation:

Implementation code is in a single file ShortestPathsInANetwork.py which has complete 5 tasks required for the project. The algorithm implemented is an algorithm used to find the shortest path between nodes in a graph called Dijkstra's Algorithm. Adding to the above task, the reachable vertex task is accomplished using Mapping and Sorting to find the vertices connected to each vertex.
The task of reachable is achieved in O(n logn) time, where n = Number of vertices.
Classes used to achieve the implementation are:

1. Graph: Build the network Graph using input file and perform operations based on query file on the network.
2. Vertex: Vertex class is used to represent vertex of the graph.
3. EdgeWeight: EdgeWeight class is used to represent edges of the graph.
4. Path: Path class stores name and distance of path.
5. PriorityQueue: PriorityQueue class is used to insert element, delete element, find element and replace element.

Task:

1. Building the Graph: Using a network.txt input file we will build our graph. We have used float to handle the floating point distance. Each input link is represented by adding two edges, one in each direction
2. Changes to the Graph: All the operations for the changes of the Graph are present in the Graph class.
3. Finding the Shortest Path: Using Dijkstra's algorithm we will compute the shortest path distance, and we have implemented using Priority Queue Data structure.
4. Print Graph: We have used Sorting and Mapping to obtain the vertex in alphabetical order, and no duplicates.
5. Reachable Vertices: For every vertex, we have found the reachable vertices in alphabetical order and we have not used the shortest path algorithm to compute that.

Complexity:

The algorithm getReachableEdges() will be called for every n vertices. So it will take O(n) running time. The recursive call to reachable() will call recursively for every adjacent vertex. So it will take O(logn) running time. Therefore, the overall running time of the algorithm will be O(n logn).
Time Complexity of Dijkstra's Algorithm is O(V^2) but with a min-priority queue it drops down to O(V+E logV).

What works and fails?

1. Input file is expected to be in precise format as required by the correct execution of the program.
2. Query file should have "quit" as the last query, else the program will run indefinitely.
3. Not tested on UNIX OS. Works well on windows X
4. The building graph, operations on graph, printing the entire graph, printing shortest path and finding the reachable vertices from every vertex all the 5 tasks expected for the project works well.