Name: Akash Singh (Student ID: 801022198)

Project 2 — Shortest Paths in a Network

Program design:

The file Graph.java consist of below Interface:

interface priorityQueue

- This provides method like remove, enqueuer, isEmpty to be implemented by its implementing class

The file Graph.java consist of below class:

class Edge

- This class is used to represent Edges in the Graph
- Constructor to initialize the edges with attributes destination vertex, cost, and edge status

class Vertex

- This class is used to represent Vertex in the Graph*/
- Constructor to initialize the edges with attributes vertex name, List of Edges, cost, and vertex status.

class Graph

- This class contains the main method. The program execution begins at main method. The network file and queries file is passed to it through it as a command line arguments.
- First, it reads the network file (first argument) and constructs a graph using it.
- Second, it reads the query file and executes each query as per the query file. For each query a corresponding case from switch case is executed and the output of these queries are written into Output file using Print Writer class in Java.
- Each case calls the required method to perform the required task based on query. All the methods like changes to graph such as

- *addedge, deleteedge, edgedoen, edgeup, vertexup, vertexdown* are implemented in this class.
- This class also implements like dijkstras algorithm to find shortest distance between source and destination.
- This class also implements method to print a graph and method to find all reachable vertex from each vertex.

class LexicographicComparator

- This class is used to implement an Interface Comparator to sort Edge destinations in Alphabetical order

class LexicographicComparatorVertex

- This class is used to implement an Interface Comparator to sort Vertex in Alphabetical order

class MinBinaryHeap implements priorityQueue

- This class implements the Interface priorityQueue.
- It provides implementation of enqueue,remove and isEmpty method of priority queue Interface
- Min binary heap is used to implement Priority Queue.

class GraphException extends RuntimeException

- This is an Custom Exception class which is used for Graph exceptions

- ✓ **Please note that Explanation of each method is mentioned as a comment in java file**

Data structure design

Map<String, Vertex> vertexMap = new HashMap<String, Vertex>();

- This Map is defined in Graph class, and is used to store vertex name and its corresponding object.

ArrayList<Edge> minbinheap = new ArrayList<Edge>();

- This is defined in MinBinaryHeap class. It is list of edges. It acts a priority queue and stores edges based on cost.
- This an edge with minimum cost is always its first elememt. Heap algorithm like bulid heap, minheapify, and extract min is implemented to achieve functionalities of priority queue like remove and enqueue

Queue<Vertex> q = new LinkedList<Vertex>();

- It is queue which stores the edges when it is first seen in BFS traversal of a Graph

ArrayList<Vertex> listofadjVertex = new ArrayList<Vertex>();

- This is list which stores the vertex, it is used to print vertex in sorted order.

TreeMap<String, Vertex> vertexMapSorted = new TreeMap<String, Vertex>();

- This is map, which is used to sort vertex on the basis of key which in this case is vertex name.

Summary:

- The initial information about the network graph will come from a file specified as a command-line argument.
- The directed graph with two edges is created, one in each direction, for each input link.
- The queries to be executed on the graph network graph also comes from a file specified as a second command-line argument
- For each query a corresponding case from switch case is executed and the output of these queries are written into Output file (provide as command line arguments)

Analysis of Complexity of Algorithm:

For finding the shortest path I have used Dijkstra's Algorithm.

- Every time the main loop executes, one vertex is extracted from the queue. Therefore, for V vertices in the graph, the queue may contain O(V) vertices. Each pop operation takes O(lg V) time since I have the Min Binary heap implementation of priority queues. Therefore the total run time is O(V lg V + E lg V), which is O(E lg V)

Time complexity to Implement methods of priority queue using bin Min Heap

- Build Heap takes 0(N) times if we have N no. of elements and calling minHeapify takes LogN time. To remove we simply need to extract first element which takes 0(1) time, however, we need to make sure that it remains min heap, for which we need to call Minheap again.
- This again takes O(N) time. Therefore the total run time of Dijkstra's Algorithm for a graph with E edges and V vertex is O(V lg V + E lg V), which is O(E lg V)

For finding the reachable vertex from a given vertex I have used BFS traversal.

- I have used BFS Algorithm to find the all vertex which are reachable from the given Start Node
- Time Complexity of each BFS call is O(V+E), where V is # of vertex and E is # of Edges.
- The time complexity can be expressed as O(V+E), since every vertex and every edge will be explored in the worst case.
- Since this method is called for all vertex, the total time complexity becomes V(V+E)
- Please note that this V(V+E) time complexity does not consider the time taken for sorting the vertex

Programming language and compiler version

- I have used Java 1.8 as a programing language to implement Encoding and decoding modes
- Compiler Version 1.8

Running the program:

- Place the two java file Graph.java in current directory
- Compile the java file using cmd as below
- > javac Graph.java
- Run the generated .class file step as below
- > java Graph network.txt < queries.txt > output.txt
- We get output.txt on running the Graph.java with command line arguments as above.