Assignment 2: Shortest Paths in Graphs
Documentation

Running:
The class with the main method to run is the GraphTester class. The program is to be ran in the format of:
java GraphTester file [dataStructure]
file is the directory of the file to read the graph from. DataStructure is the data structure to use for Dijkstra's algorithm. This can either be pq (for priority queue) or l (for list). If a data structure is chosen, the program does not output the results of Dijkstra's algorithm. Instead, it will display the time taken for the algorithm to run. If it is left blank, the default data structure of a priority queue will be used, and it will not time the results.

How it works:
It starts by creating a graph from the file that was read in (specified by user). The graph is made up of two main data structures. There is the adjacency list that is in the form of a TreeMap with vertices for keys and TreeMaps for values. These TreeMap values represent all of the edges coming from a vertex. The TreeMap value has a String for a key and an Edge object for a value. The string is composed from the name of the vertex that the edge goes to, followed by a space then an identifier for this edge (in case of multiple edges to the same vertex). An example of this could be “B 2” which means the edge goes to vertex B and there are at least two other edges that go to B from the current vertex. The Edge object has a weight and knows the source and destination of the edge that it represents. It also has a label so it can be identified internally. TreeMaps are used for the outer and inner sections as they keep the elements sorted and also provide decent access time. The second main data structure used by the graph class is the vertex list. The vertex list is a HashMap with String keys and Vertex values. This data structure exists so if the user knows the name of a vertex but doesn't have the object, they can look it up here. HashMap is used because of the great random access speed, and because of the fact that I know that reverse lookup isn't necessary because every vertex knows its own label.

For the get shortest paths method in AdjacencyListDirectedGraph, 3 main data structures are used. There is a HashMap of Vertex-Float pairs that stores the distance of every Vertex from the starting Vertex. There is a HashMap of Vertex-Vertex pairs that stores the predecessor of every Vertex in the shortest paths. The third is the priority queue that stores the vertices then uses a comparator to do approximate sorting from their respective Float values from the distance map.

After the method has run through Dijkstra's algorithm, it puts the results of the first two data structures into a format that is suitable for returning. It returns a TreeMap of Vertex-String pairs where the String is the distance from the starting vertex, a space, and then the predecessor to that vertex (e.g. “3 E”). This allows all necessary information to all be bundled up to return.

Link to git repository: https://github.com/MisterMarvellous/CP3Assignment2.git