232 Project

Setup:

Gephi Version 0.9.1 has been used to import the "babel.gexf" which was gathered from http://moviegalaxies.com/movies/download/92/Babel then the graph was built visually in Gephi.

Next edges and edge properties (source, target, type, edge id, weight) has been exported to a csv file called "babel.csv".

Classes of the project:

Only the classes which has been provided by textbooks has used. *Bag* DijkstraEdgeWeighted *Edge* EdgeWeightedGraph *In* IndexMinPQ *LinearProbingHashST* Queue *Stack* StdIn *StdOut* StdRandom * SymbolGraphTwo

Editing:

First Edge then EdgeWeightedGraph class was edited as needed. Changed the constructor of Edge and methods in EdgeWeightedGraph as follows.

Old one:

- public Edge(int v, int w, double weight)

This constructor used to hold only starting node, ending node and weight of an edge.

- addEdge(int v, int w, double weight)

This method used to take only starting node, ending node and the weight od the edge.

New one:

- public Edge(int v, int w, double weight, int id, int timesUsed, String from, String to)

Now it holds starting node, ending node, weight, id of the edge and an integer value timesUsed which counts the shortest paths that is used on specific edge. String from and to is used for source and target nodes as string values because i merged 2 csv files node and edge csv instead of using txt that has been provided.

- addEdge(int v, int w, double weight, int id, int timesUsed, String from, String to)

Now addEdge method which is in EdgeWeightedClass takes 3 more inputs as follows: timesUsed to see how many shortest path that specific edge has, from and to for string input which is gathered from csv file.

- Added edgeID() method into the Edge class to get information on edge to see how many times an edge has been used.

- Added getFrom() and getTo() methods to see starting and ending nodes of an edge.

- Also added minor changes to make it visually clear.

Running:

Project runs with SymbolGraphTwo class. SymbolGraphTwo class has been modified to work with Dijkstra's Algorithm. SymbolGraphTwo contstructor takes two string inputs first for the path of the input file and the second one for the splitting a line according to properties of line.

Then we do have two hashtables called "st" and "st2", both of them were created from LinearProbingHashST class. "BabelTest.csv" file only contains edges so we see a specific nodes several times but by checking it with "contains" method we only put nodes only one time in to the "st" hashtable. so the size of st becomes 71 and we create an EdgeWeightedGraph with the size of "st".

Then we read the csv again but this time there is no need to check if node has been used once, twice or more because we are adding the edges in to the graph we created earlier. So the EdgeWeightedGraph is ready with all it's all nodes and edges.

Now we have a graph but we want to use Dijkstra's Algorithm in this graph. "st2" hashtable is defined and it is empty. We create a SymbolGraph and turn it into EdgeWeightedGraph. And we call Dijkstra into the EdgeWeightedGraph by loops to check for every node. First for loop is to apply Dijkstra' Algorithm in specific range of nodes. Second loop takes the every node in graph and checks if there is a path between the nodes.

And if there is a path has been found then it uses these edges and increases the time used value by one on the specific edge that has been taken with "e.edgeID()".

Finally with a for loop in st2 hashtable we are able to see which edge has been used how many times.

HashMaps:

- st is for nodes.
- st3 is for the number of shortest paths that has been used between every pair of graph.