Izmir Dokuz Eylul University

Department of Computer Engineering

CME 2201 - Assignment 2

Designing a Journey Planner For Paris Metro Lines

By Using Graphs Data Structures

2021510010 - Çağrı AYDIN

Data Structure

The focus of this project is learning and getting experienced at Graph theory, graph algorithms and ADT. For this purpose, we used Directed Graph data type for our homework. We stored every train/bus station as a vertex and every path as an edge.

How code/methods work

->In DirectedGraph.java

In **addEdge** methods we get few inputs form the user such as destination point, start point, weight of the edge and routes short name. Inside of the method we first look at the given destination point and start point if our vertex array list doesn't have one of them or both first, we add the vertices inside of our vertex array list then we make connection between these vertices by using **addEdge** method.

In **fewer_stops** method we aim to use least amount of stops to reach our destination point for doing this we need start and end stop names. When we give the names of the points we search down all vertices and get their vertices. Then we push our start vertex into our childs1 stack. Then we pop our vertex in childs1 stack and assign it as a current vertex. We get all the neighbours of our current vertex and push them into childs2 stack while pushing them into the stack we assign their parent value as our current vertex. When we empty our chids1 stack and we empty childs2 stack into childs1 stack and repeat the process until our current vertex equals our destination vertex. Then we kill all the loops and call **write_parents** method to write down stop names.

In **write_parents** method. This method is a recursive method that will write current vertex's parent until it sees a null parent.

In **reset_verticies** method we visit all the vertices and make their visit, cost and parent values to begging values.

In **dijkstra_algorithm_search** method, we are able to find the path that takes the least time reaching from the starting point to the end point. Source and destination stops' names are given as parameters and vertices are gotten. When source and destination vertices are gotten, then we apply the dijkstra algorithm to find the fastest path. Dijkstra algorithm works by adding the fastest way from one vertex to another as parent so we implemented another method called **print** in order to print all the stops along the path.

In **print** method, we take the source and destination vertices as parameters. Starting from the destination vertex, we push the parents name, discovered by the **dijkstra_algorithm_search** method, to a Stack data typed object with the help of a “while” loop. This helps us ordering the path from source to destination since Stack data types work as “First In Last Out”. At last, we add all the elements inside the Stack to a string in order to add some “—>” between the stops, print the result and the total time spent on the journey.