CME2201 Data Structures Assignment

JOURNEY PLANNER FOR PARIS METRO

2022510127, Kerem Kalıntaş

2021510089, Berkay Karakuş

January 4, 2024

1. Introduction

In this assignment, we are expected to implement a simple journey planner for the Paris metro using weighted directed graphs. But of course, the journey planning implementation is just a blanket for the main goal of this assignment. The main goal is to implement these graph algorithms in a real like scenario.

We are going to explain the logic behind the graph algorithms and path finding in later chapters.

2. Explanation

2.1 Directed Graph

Directed graph class is for holding weighted directed graphs in an adjacency list. This list holds abstracted vertices and edges that connect to itself. Class has following functions:

void addEdge(Vertex source, Vertex destination, int weight);

boolean hasEdge(String source, String destination);

int getShortestPath(String begin, String end, LinkedList<Vertex> path, PathFinderLambdaInterface lambdaInterface);

public int getCheapestPath(String begin, String end, LinkedList<Vertex> path, PathFinderLambdaInterface lambdaInterface);

Difference between the getShortestPath and *getCheapestPath is getShortestPath* tries to find the path with minimum hop count while getCheapestPath searches the actual shortest path in terms of cost. getCheapestPath uses priority queue to implement its algorithm.

PathFinderLambdaInterface is an interface for restricting paths in some conditions. This interface is used in pathfinding algorithms *getShortestPath* and *getCheapestPath*. Has the following function:

boolean isValid(Vertex currentVertex);

This *isValid* function is called every time the path find reaches end of the path. And does not allow when this function returns false. This functionality is later used in limiting transfers and disabling two consecutive walks.

2.2 Metro Station

Metro station class is for abstracting the stations in this metro network. Every station in this metro network has at least two stops and some of them have neighbor stations that are allowed to walk in between. This class has the following member variables:

HashMap<String, MetroStop> stops;

HashSet<String> neighbourStations;

Stops are stored in the HashMap according to their short route name and direction id. These stops are then searched with another stations using these values.

2.3 Journey Planner

This the main class for this application. It creates the journey planner with given metro and walk edge information. Class has two attributes that change the behavior of the suggestion. These attributes are *pathOptimization* and *transferLimit*. First one being the optimization criteria when searching the path and later is the transfers a path can take at most. Class has the following function for suggesting routes and finding paths:

void suggestPath(String originStation, String destinationStation);

Average query time of this functions is measured as about 1.57 milliseconds with given test cases.

3. Conclusion

In conclusion we implemented a metro station journey planner using weighted directed graphs. This journey planner can be used to suggest paths in given destinations. It will change it's behavior based on its internal member variables and suggest a different path.