Surabhi Ravishankar
Email: surabhiravishankar2016@u.northwestern.edu
EECS 395 – Data Structures and Data Management

PROJECT-4 REPORT

Execution:

$ make
$ ./proj4 locations.txt segments.txt trips.txt

Design and Implementation:

Implementation of Dijkstra's Algorithm:

The goal of this project was to implement the Dijkstra's Algorithm to find the shortest path. I implemented Dijkstra's algorithm by creating a class called dijkstra. This consists of functions to insert into the adjacency list, get at list of shortest distances and shortest times. I used the following pseudo code to implement the algorithm. First I initialize all the distances to each of the nodes in the graph to 9999 which means that the nodes are not reachable and mark all of them as not visited by maintaining a visited array. The algorithm works by adding the source node on to the heap. Then the node is marked as visited. The minimum node in the heap is deleted. For each of the nodes that is adjacent to the deleted node, the total distance from the source node is calculated. If it is the shortest path, then the adjacent node is inserted to the minimum heap. The same algorithm is implemented for the shortest distance and the shortest time. I have implemented the same thing that was taught in class.

A priority queue was to be implemented. This was done using a minimum heap. A separate class was created for this. Used an array implementation for minimum heap. Functions to alter the heap according to the minimum and maximum distances and time were written. There are detailed comments in the program files.

Implementation of the graph:

I have maintained an adjacency list to store all the nodes in the graph. The adjacency list is defined in AdjList.h. I have defined functions in the list class that would return the cost from node a to node b at every iteration.

The adjacency list stores the following information, the id of the adjacent vertex, the distance to that vertex and the time taken to travel to that vertex.

The next step was to construct a directed graph. I have used a location class in which I have written functions that are used to create the edges and get the destination names. I created another graph class that integrates the list class and the Locations class. In the graph class I have defined functions to actually add the vertices and edge information into the adjacency list and construct the graph. The edges contain information about the end node, the distance, time and speed.

The final program works by extracting the inputs form the user. Then, the vertices are added to the graph. Once the segments file is read, the edges are created. Then once the trips file is read, the a call to the dijkstras class is made in order to print the shortest path.

Please refer to the comments in the program for a more detailed explanation of the design decisions.