CSU22012: Data Structures and Algorithms Final Project

Kenneth Harmon SN: 20333659

# Part 1:

For the shortest path section of the assignment, since there wasn’t any obvious heuristic to be used in this implementation, this eliminated A\* and Greedy-best first search algorithms. My main two options from there were Dijkstra and Floyd Warshall which were both valid options since after exploring the files there were no negative cycles. My initial thought was to use Dijkstra due to the size that would be associated with an adjacency matrix for a graph of this size. And also, since its performance of v2 would be favorable with a large graph even though it would have to be run each time a path was searched for. However, I needed to make sure the wait time was acceptable for something that would have to run on each search. So, after some testing I felt it was at an acceptable level and decided to proceed with Dijkstra. After implementing the rest of the assignment, I still feel happy with this decision since with the other initialization of the hash maps and TST’s at the beginning of the program, the initialization time is longer than that of the runtime for Dijkstra, and so this would have been further increased by running Floyd Warshall.

# Part 2:

For searching for a bus stop by name I used a TST as specified however I initially created an object to be returned by the TST instead of a plain string. After reviewing this I decided a plain string should be enough for this application since the search is just returning all of the information about that stop however if these stops were going to be used elsewhere it might be worth it to make then individual objects, similar to my implementation for part 3. To return this string with the stop information I used a HashMap which maps the string fed to the TST to the string with all of the information about that stop. So when a user searches for a stop, if that string can be found in the TST it is passed to the HashMap which returns all of the necessary information about that stop.

# Part 3:

For the search by arrival time I decided to use a HashMap in combination with a class to represent each stop along the trips. I am aware this increases space requirements however this makes it easier to sort the results by trip ID, so I thought it was a worthwhile tradeoff. On startup, the program reads the stop\_times.txt file, creates a new StopTime object for each entry, and stores it in an arraylist which is accessed by using the arrival time as a key for the hashmap storing these arraylists. One saving I noted that I could have made here was that I am loading the stop\_times.txt file twice, once for the graph, and once for this section, I did consider trying to populate the data for both of these sections using one read of this file, however I felt it would add unnecessary complexity to the program and make the classes less modular.