CS 225 Final Project Development

Saket Vissapragada, Siddhartha Adatrao, Sruthi Kilari, Shreya Sharma

Fall 2020

1 Week 1

In the first week, we met and decided on the project goals and team contract. Then, we divided up the work evenly between the group, with two people focusing on data pre-processing and the Landmark Path Algorithm and the other two focusing on the BFS traversal and Dijkstra's Algorithm. To begin, we must preprocess the data and create a Makefile and class for the data in order to complete the rest of the project. We will add tests as needed. As we continue with the project, we will have a clearer understanding of the specific tasks that need to be done, and we will assign them accordingly.

2 Week 2

During week two, we focused on implementing the BFS traversal while also preprocessing the data. While preprocessing the data, we ran into issues with the CSV file containing inconsistencies. We met and worked in two groups of two. Next, we will work on implementing both Dijkstra's Algorithm and the Landmark Path Algorithm.

3 Week 3

During week three, we once again worked in groups of two and successfully implemented our graph traversal and algorithms. We started off the week debugging issues that were associated with creating a Graph object (edge weights, connected nodes). Once we resolved these issues, we were then able to easily develop and test our graph traversals/algorithms. One issue we encountered was with our Makefile producing a linker error when we tried to compile the code intended to construct the graph of our data. We resolved this issue by manually adding the correct dependencies for each file. We also ran into some issues while implementing Dijkstra's Algorithm. We initially tried to use a priority queue to update the weights, but it wasn't working as we intended because if the current node approached a neighbor that was already on the queue with a lesser tentative distance, the priority queue would not edit in order. To solve this issue, we used Prof. Evans' suggestion to add a new node reflecting the lesser distance and every time a new node was popped, a check to see if the tentative distance of that node corresponded to the up to date distance map. If these were different, this node was ignored and the next node was checked.