DS210 Final Project Report

December 15, 2022

0.0.1 DS210 Final Project Report

In this project, I analyzed a bus dataset hoping to find general trends in how buses in the greater Boston area operate. However, upon a closer look at the dataset, I found that the data was quite disorganized and difficult to manage. As such, I used a small amount of Python to manipulate the data until I could properly analyze it using Rust.

Python Code In my Python code, I shrunk the dataset to focus on specific times of day, as well as directions the buses are traveling in, as bus stops differ depending on whether buses are inbound (headed into the city), or outbound (heading to surrounding areas). For this specific project, I limited the time of day to AM_PEAK hours, limited the routes to outbound, and only selected stops which were either (1) stops where multiple bus lines connected or (2) the startpoint or endpoint for a given line. I also averaged out ridership numbers in order to create a weighted graph in Rust, where the weight of a vertex is the difference in the average number of people the bus is estimated to be carrying between the first and the next stop. To estimate which stops connect to one another (since typically only one station can come before and after a given stop on a given route), I took the simpler approach of averaging out the sequence (in its respective route) of each stop, and then sorting them by average sequence to determine roughly which stops were in front of or behind others. This rudimentary approach allowed me to create a largely accurate graph of the key bus routes of Boston, focusing on the terminal stations as well as connecting stations.

Rust Code After importing my dataset into Rust, I then created a vector of instances of the Struct Stop, which I used to define each bus stop. Once I had a vector of stops, I was then able to create an adjacency matrix, with the value of each non-zero entry being the difference in ridership between the two stops. Finally, I executed a depth-first search algorithm on a select number of stops, which returns a list of indexed stops which compose of the shortest path based on the depth-first search algorithm. My depth first search algorithm uses a recursive helper function to determine where it has travelled and what direction it should go in next. Once it is complete, it returns a list of nodes (stops), which it has found to be the shortest route between two given stops. I could then backtrace these indicies to find the original station names in my main dataset. I have also attempted to implement a shortest path algorithm based on Dijkstra's algorithm, but as of the due date, it does not work properly with the rest of my code.

Findings What I found was that, as could be expected, many stops are not connected with one another as the bus system mostly runs linearly, and connecting to other lines without other forms of mass transit is difficult. I feel that this project could be expanded further to include more directions (inbound as opposed to outbound), times (perhaps PM_PEAK), as well as stops to create a better connected graph.

Ultimately, the most difficult part of this project is aggregating the bus data, and if that can be done more efficiently, more analysis could have been done within Rust.