**SI 507 Final Checkpoint**

**Name Yujie Li**

**Github repo:**

<https://github.com/VectorLambda/Final_Project.git>

**Data Source:**

The initial plan was to use the API provided by metro bus system in Ann Arbor. Unfortunately, we require the permission of The Ride in order to access the API for their application. Given the timeframe and the uncertainty of using a company’s API, I was forced to drop the idea of accessing the API for coding. Instead, my plan was to use the bus schedules, which contain the route and public information, as the source of my data. The trick of this approach is to use the PDFQuery to extract the data from PDF documents, and reorganize them into JSON format.

For the simplicity’s sake, during the development of the code, we currently have 3 bus route schedules in database. Each Data file is in the form of JSON. For each bus route, there are two sets of schedules: one for each direction of bus route. For each set, there are three types of schedule: weekdays, Saturdays, and Sundays.

To cache the data file, first, we extract the file as a list of string. Then, we will commit a series of list elements operation to restructure the data in the form we desired. After that, we will then load the content of the txt file into json format as cached database.

The cached data files for each bus route is listed as following snapshots:

Text

Description automatically generated

**Data Structure:**

The data will be structured into as a graph. Each bus stop station is a node, while the lines connecting the nodes will be the estimated time from one station to another, with its weight representing the estimated time to travel between each node.

Using the bus schedules that we gathered, we overlap these bus route as one graph. Since the bus schedule has the bus leaving each station periodically, we will implemented a check to determine if the total cost of traveling so far had been a multiple of a certain value(assume 30 costs for 30 minutes). Every time it reaches a node, it will check the closest node that has the smallest value after modded by 30. Using this searching algorithm, we will pick the branches that take the least amount of cost to traverse from the starting node to the destination node.

Below is a rough idea of the data structure we are planning. Black,Red,and Yellow nodes are the stations for different bus routes, we will then connect nodes across different bus route as the transition path from one bus route to another(representing the walking time need to another bus station, the maximum weight of the path would not surpass 5, since we are looking for bus routes that is within reasonable walking distance.). The completed graph structure will be a complex and sophisticated bus transportation network that connects most of the stations in some ways.

Diagram

Description automatically generated

**Interaction and Presentation Plans:**

For interaction, the user needs to input the name of their starting location and destination. Upon receiving the inputs, the system will then process and generate a path that takes the least amount of time to travel when reaching the destination. The user can also provide the bus node which they are trying to avoid, reflecting the area where bus would require detour. As the result, the output will provide an alternative route that avoid those area completely.

For the final presentation, we expect to use plotly to construct out graph for the bus transportation network. In addition, we will also provide visualized graph data structure to reflect the structure as the map of the bus route as a reference for user. Whenever a bus route is generated, the path taken will be highlighted to give the viewers a visualized path in the final result. The purpose of the visualization is to imitiate the user experience and expected output when using GPS or other path-finding applications, such as Google Maps, Magic Bus or One Bus Away.