EECE 5642 Data Visualization
HW 4

Name: Sai Prasasth Koundinya Gandrakota
NUID: 002772719

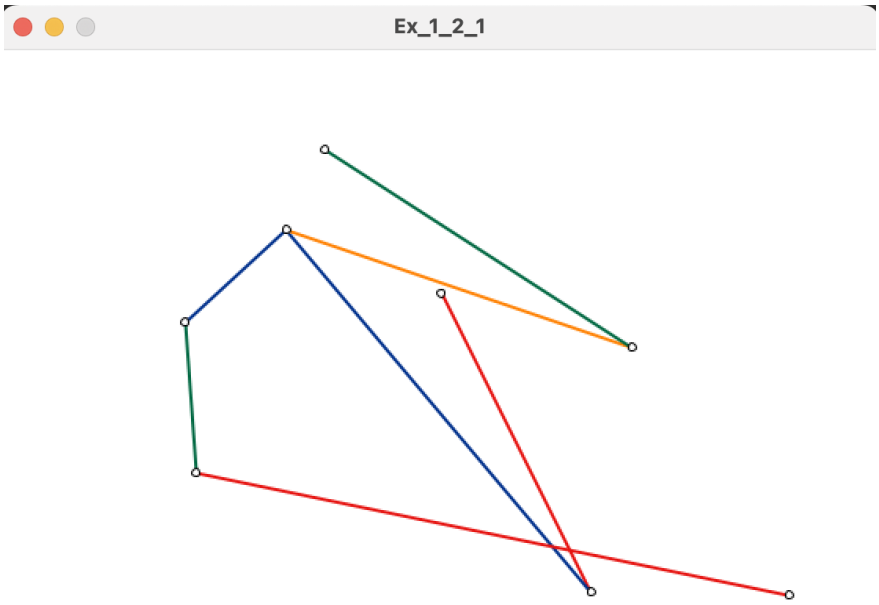
Visualizing Boston's Subway Network System

Setting Up

Ex_1_2_1

Firstly, we use the Framework sketch provided in the assignment to set up the *Edge* and *Node* classes along with the main code that draws the edges and nodes. Once analyzed, we add a field *col* of String type to the Edge constructor. The edge constructor takes in the string *col* and based on the first letter of the string, assigns RGB values to a color variable edgeColor and stores it for each edge. Currently four colors are accepted: red, blue, orange and green.

Now in *Ex_1_2_1* sketch we can add the *col* argument to the *addEdge()* method so as to assign color to the edges created by the Framework sketch. The outcome is as shown below:



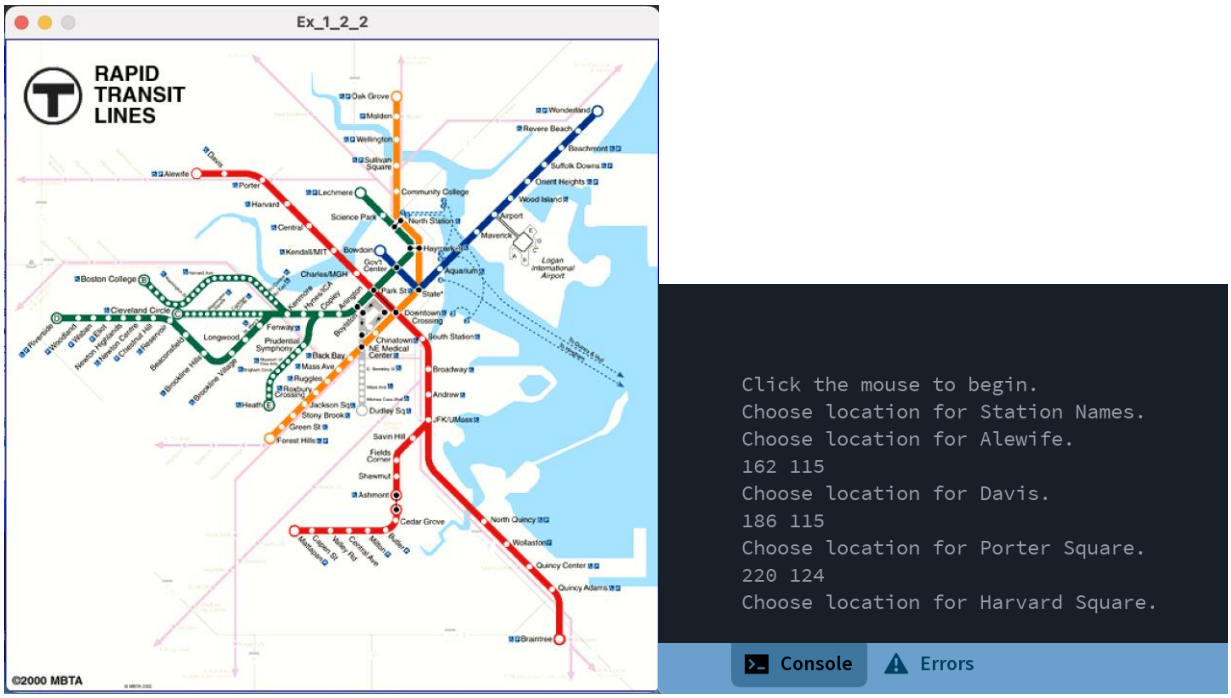
Acquiring the Data

Ex_1_2_2

Once we setup the base sketch, we then create two python scripts, *connections.py* and *station.py* which extract the connection and station tables respectively from the given html file *data.html* into CSV files following the same naming pattern. The python scripts can be run using the local installation of Python 3.0 on a machine. Once run, the connections.csv and stations.csv are created in the TheTDataCollection folder which houses the python scripts.

After creating the CSV files, we copy the *stations.csv* file to the *Ex_1_2_2* folder, derived from the *Using_Your_Own_Data* sketch. We alter the code so that the *stations.csv* file is read in, the output file is called *locations.csv* and the provided image mbta-map.gif is used to define locations on the map. In the *Table* class, we update the write method to now separate the extracted values by commas instead of tabs. We also add the given line to trim the whitespaces before and after station names. In the main *Ex_1_2_2* sketch, we run a for loop through the list station names and for each station the console requests and input for the user which is given by clicking on the approximate location of the station on the map image. Once coordinates for all stations are added, the *Station Name, X-coordinate and Y-coordinate* are stored in the *locations.csv* file.

As we can see below, once the program is run a map is shown with all the stations, and the console requests input from the suer which is acheived a mouse press within the image.



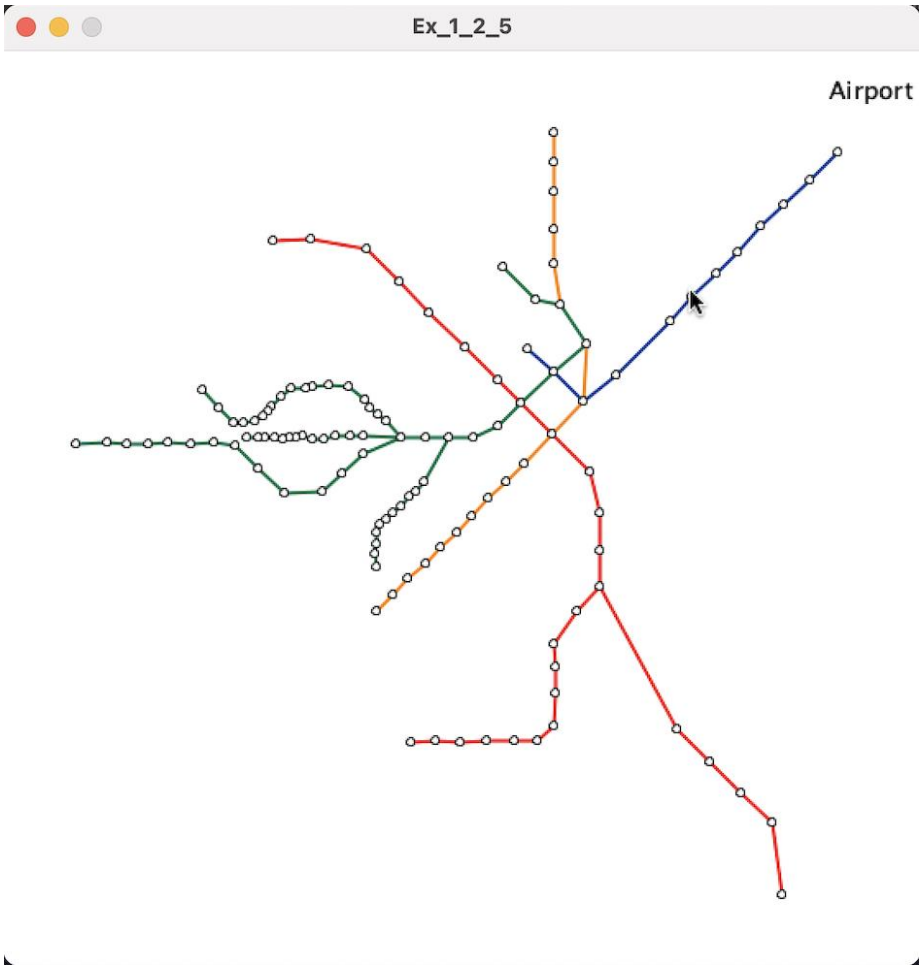
Visualization of the Network

Ex_1_2_3

After collecting the data, we copy both the locations.csv and connections.csv files to the Ex_1_2_3 folder which is a copy Ex_1_2_1 sketch but with an added Table class.

During the setup() method, we load data from the CSV files into respective tables. From the connections table we extract, from Label, to Label, minutes and line color in each row and then add edges using those values.

We create a mouseHover() method to call in the draw() method whenever the mouse rolls over a node by checking the current mouse pointer coordinates, and if it is within 5 units of node's corresponding coordinates, then the name of the node/station is displayed in the top right corner of the output sketch, as shown below:



Shortest Path

Ex_1_2_4

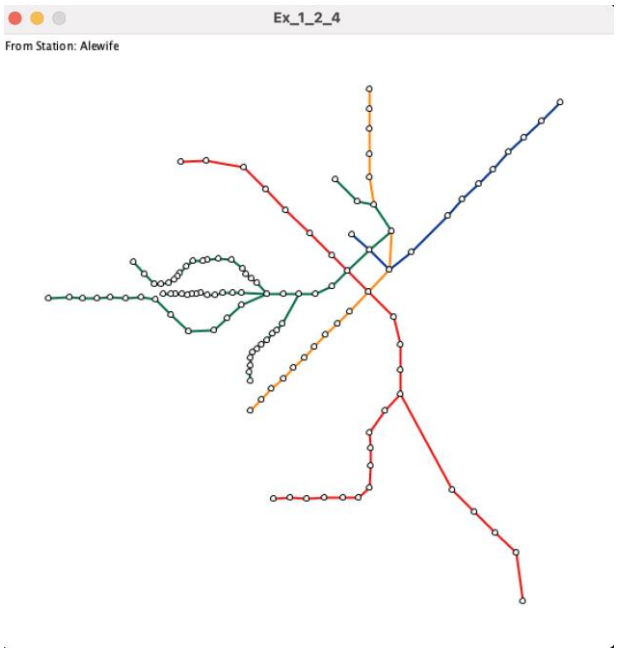
After visualizing the MBTA map as a series of edges and nodes, we then add the shortestPath class provided in the assignment to a copy of the above sketch, naming it as Ex_1_2_4. First we add the initializeActiveDataStructures(); method to the setup() block so that two new boolean arrays activeEdges and active Nodes are set to have all values as TRUE.

Then we add the initializeAdjacencyMatrix(); method which intializes a couple of other data structures.

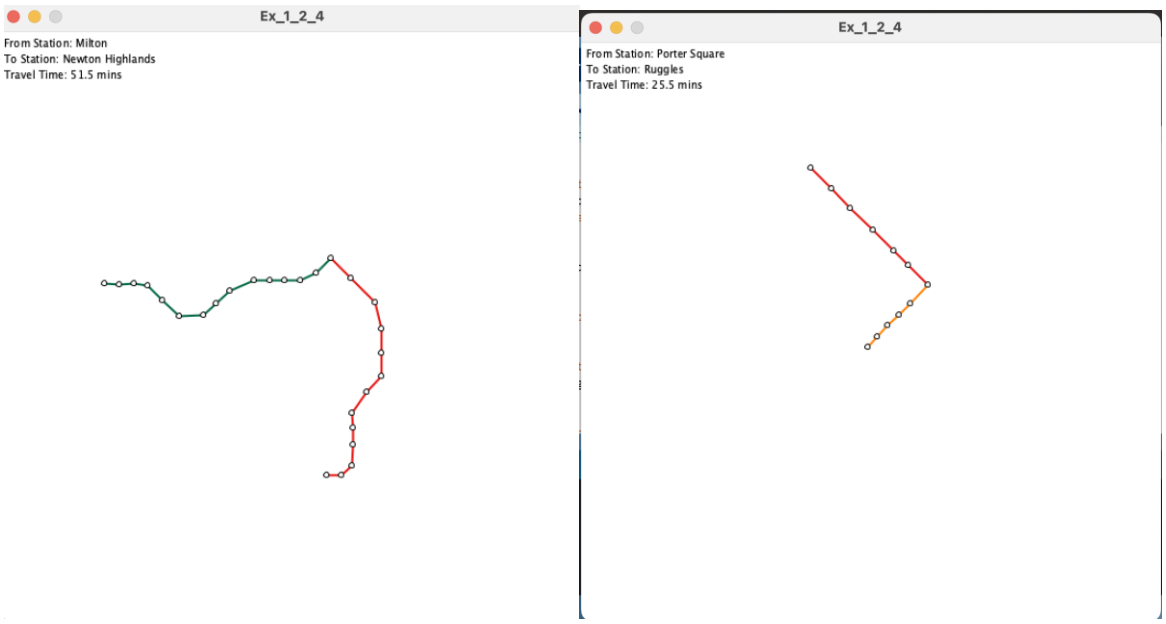
In the main code, we update the mousePressed() method such that when a node is right clicked on, it is loaded in a node A and the numOfNodes variable is incremented from 0 to 1. A subsequent click on a second node loads that node into node B and increments to numOfNodes

variable to 2. After the second click the shortest path between the two nodes is calculated by calling shortestPath(A.getIndex(), B.getIndex()) and stored in numOfMinutes variable, which gives the total time it takes to traverse the shortest path between two nodes. A third right-click re-initializes the nodes and the variables to 0.

In the draw method, we update the code such that when the first node is selected, it's label/ station name is displayed in the top right corner as the 'From Station', shown below:



When the second node is selected, the activeEdges array is updated so that only the edges and nodes that are a part of the shortest path between the two nodes are set to TRUE and the rest are set to FALSE. Once this is done, the second node label is displayed as the 'To Station' below the 'From Station' and the total number of minutes is displayed as 'Travel Time' below that. Only the active edges and nodes are drawn, and the inactive edges and nodes are ont displayed. The outcome is shown below:

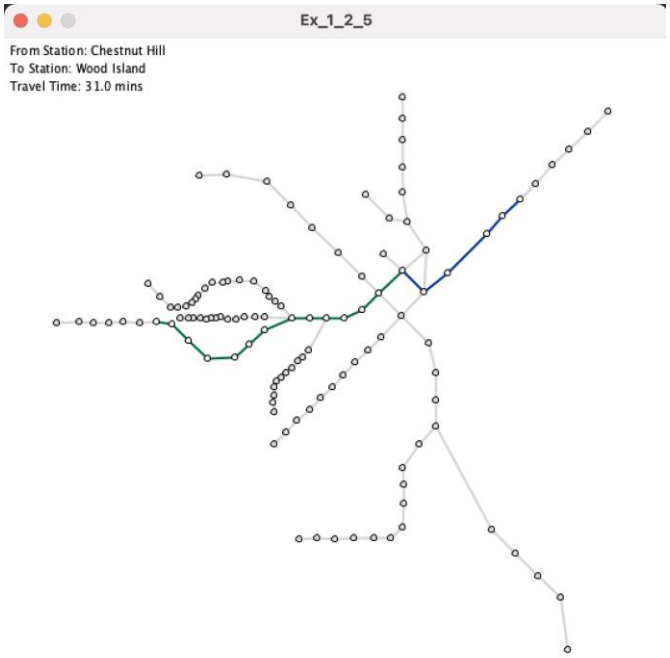


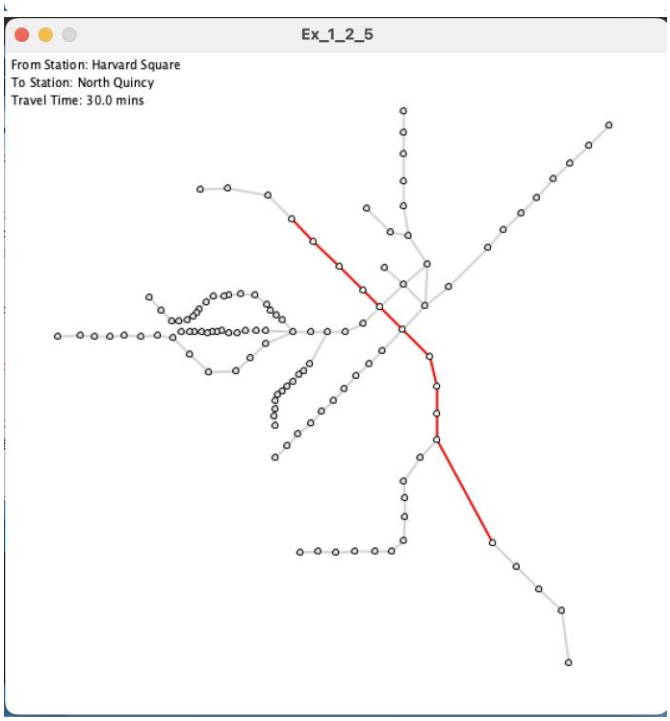
Upon right clicking anywhere on the output again, the entire map is reset to the original state, displaying all edges and nodes and the text in the top right corner is removed.

Color Effect

Ex_1_2_5

Finally, copying the above sketch into a new folder Ex_1_2_5, we add the Integrator class to the sketch. Here we update the code so that when two nodes are selected, the active nodes and edges are colored in, whereas the inactive nodes and edges are greyed out by setting the brightness and saturation to 200 and 0 respectively, while the hue remains the same. The Integrator class is used to smoothen the transition between the color changes. The outcome is as shown below:





REFERENCES

1. All code for all steps of the sketches (Ex_1_2_1, Ex_1_2_2, Ex_1_2_3, Ex_1_2_4, and Ex_1_2_5) along witha demo video and the TheTDataCollection folder containing the python scripts are uploaded along with this report in Canvas.