Northeastern University

HW4

EECE 5642 Data Visualization

Homework 4
Instructor: Y. Raymond Fu
Due Date: **11:59pm Mar. 21**
Submission: Canvas
Courtesy of Prof. Hanspeter Pfister, Harvard University.

1. Networks, Graphs, and the T (100/100)

For this part of the assignment you will be using data about Boston's subway network system, the T, to create a graph visualization with Processing. The T can be modeled as an undirected graph (*Stops* are nodes/vertices, and *Connections* are edges).

Did you know?... the Red Line is so named because its northernmost station used to be at Harvard University, whose school color, as you all know, is crimson.

To learn more about the history of the T, you can go here and here.

1.1 Getting the Tools in Hand

First, download this zip file containing the first two example sketches from Chapter 8 Fry's. Place the sketches in your Processing *sketchbook* directory.

Second, read carefully through the first part of Chapter 8 (pp. 220 - 242, Subsections: *Simple Graph Demo*, *A More Complicated Graph* and *Approaching Network Problems*) and work through the two example sketches. Both sketches are named according to their subsection title in the chapter.

1.2 The T

In this part of the assignment you will apply your Processing skills to visualize a real world network: the T. We'll focus not only on visualizing the network, but also on data acquisition from the web (preparation for the final projects).

1.2.1 Setting Up

Create a new folder named "HW4" in your *sketchbook* directory and download the following Framework sketch. Copy the unzipped sketch folder "Framework" to your newly created "HW4" directory.

Open the sketch and familiarize yourself with the code. Run the sketch. You can drag Nodes by pressing and holding down the left mouse button. If you press "p" the next call to *draw()* writes everything into a pdf file "output.pdf" and places it into your sketch folder.

Did you know?... The PDF file format is a vector format. This allows to resize the image file without loss of resolution.

Save a copy of the "Framework" sketch as "Ex_1_2_1" in the "HW4" directory.

You will now add a field *col* to the *Edge* class, which allows us to specify its color. Add a fourth argument *col* of type String to the constructor of the *Edge* class. Assign an edge color based on the first letter of the argument *col*. We only allow four different colors:

col	first letter	(r, g, b) triple
"red"	'r'	(230, 19, 16)
"green"	'g'	(1, 104, 66)
"blue"	'b'	(0, 48, 140)
"orange"	'o'	(255, 131, 5)

The *addEdge()* routine in the main tab should also take a fourth argument *col*. Change the code accordingly. You also have to make changes to the *draw()* method of the *Edge* class.

Add color to the edges of the graph. They are also a little bit too thin -- increase their stroke weight by one.

1.2.2 Acquiring the Data

Next, you'll acquire all the needed data by parsing the following webpage: data. We collected station and connection data from the MBTA website. (Note: Data for the "Minutes" column of the "Connections" table was not available for all lines -- some values may differ from the real amount of time it would take to travel between the two corresponding stations.)

Your task is now to write two Python scripts (or any other language that you used for the first part of this assignment): "stations.py" and "connections.py". The first extracts the list of stations from the first table of data. The second reads the connection data.

The output of "stations.py" should be a list of station names, with one station name per row. Write the list to a file "stations.csv". The file extension ".csv" stands for comma-separated values. It's similar to the ".tsv" file format we were using before. The only difference is that the tabs are replaced by commas. (We introduce it here because it's a widely used format to store data and may be useful for your final projects.)

The information about the connections should be stored into a file named "connections.csv". The row format there is defined as follows: "From", "To", "Color", "Minutes".

Store copies of your Python scripts in a folder called "TheTDataCollection" in your "HW4" directory. Include instructions in your write-up about how to run the scripts.

Now we are ready to define the Node locations on the screen. Download the Using_Your_Own_Data sketch and place the unzipped sketch folder into your "HW4" directory. We used this sketch to define locations on a map. We can reuse this code to define the locations of the stations now.

Save a copy of "Using_Your_Own_Data" as "Ex_1_2_2". Add the following MBTA map to the sketch. We'll use it to define the 2D locations of our T stops. Change the code so that it reads and shows station names. Use the *Table* class to read in the "stations.csv" file. Store the station names together with their x and y coordinates to a file "locations.csv". (Row format: "Station Name", "x", "y".)

Remark: If you change the argument "TAB" of the split command to ',' in the *Table* class, it is able to read comma-separated files. By adding another line of code, we make sure that the white space at the begining and end of the strings are removed:
Add: *for (int j = 0; j < pieces.length; j++) { pieces[j] = trim(pieces[j]); }*
For reading and displaying the station names in the above example it doesn't really matter, but it will be helpful/important in the next couple of tasks.

Whew! Take a deep breath as the painful part is done -- the data is acquired.

1.2.3 Visualization of the Network

We'll now visualize the network. Before you start, save "Ex_1_2_1" as "Ex_1_2_3". Add the Table class to the sketch (changes from the above remark).

Load the two files "connections.csv" and "locations.csv" into tables. Load this data by making changes to the *loadData()* and the *addNode()* routines. If you now run the sketch you will see the T network.

Next, add code so that the station name is displayed in the upper right corner of the screen, when the mouse rolls over a station.

1.2.4 Shortest Path

Save the previous sketch as "Ex_1_2_4".

What is a shortest path in a network? That's pretty simple. Let us assume the following scenario: You are late for a meeting at the Government Center. You are still at Harvard. At which stations do you have to change to get there as fast as possible?

You do not have to code a shortest path algorithm yourself --- we've done that for you. Download the ShortestPath file and add it to your sketch. It is (hopefully!) straight-forward to use:

1. Add two global arrays of type "boolean", one named *activeNodes* and the other named *activeEdges*. Intialize them by adding *initializeActiveDataStructures();* to the end of the setup function. If *activeNodes[nodeIndex]* evaluates to true, then the node with index *NodeIndex* is part of the shortest path. Similar for the edges: If *activeEdges[edgeIndex]* evaluates to true, then the edge with index *EdgeIndex* is part of the shortest path.
2. Add *initializeAdjacencyMatrix();* to the end of the setup function. This initializes a couple of other data structures. You do not have to worry about them. You only have to make sure that they are

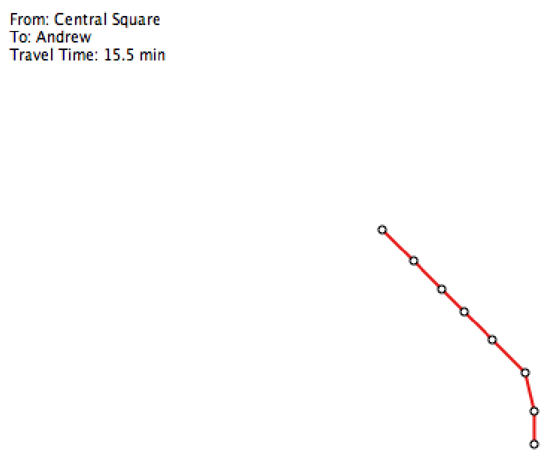
initialized!
3. Now let us assume that you would like to compute the shortest path between a node A and a node B. The following call *shortestPath(A.getIndex(), B.getIndex())* updates the arrays *activeNodes* and *activeEdges* and returns the corresponding travel time. That's all you have to do!

For those who are interested in the underlying algorithm, may find the article about [Dijkstra's Shorest Path algorithm]() a good starting point.

Ok. Now it's your turn. Add four global variables: *A* and *B* of type "Node", *numOfNodes* of type "int" and *numOfMinutes* of type "float". Add code to the *mousePressed()* method so that the following requirements are met:

1. If you click on a node with the right mouse button it should assign the corresponding node to the global variable *A* and increment the *numOfNodes* variable.
2. If you click now on another node with the right mouse button it should assign it to the global variable *B* and increment the *numOfNodes* variable. Further, it should compute the shorest path (don't forget *numOfMinutes*). Make also changes to the *draw()* routine in the main tab, so that only nodes and edges on the shortest path are drawn.
3. If you then right click somewhere on the screen you should clear *A* and *B*, you should set the variables *numOfNodes* and *numOfMinutes* to zero again and you should set all the nodes and edges to active again.

Run the sketch and try it... there is still something missing. You do not display the time it takes to travel from A to B! Add code to the *draw()* routine in the main tab so that it writes the shortest path information in the upper left corner:



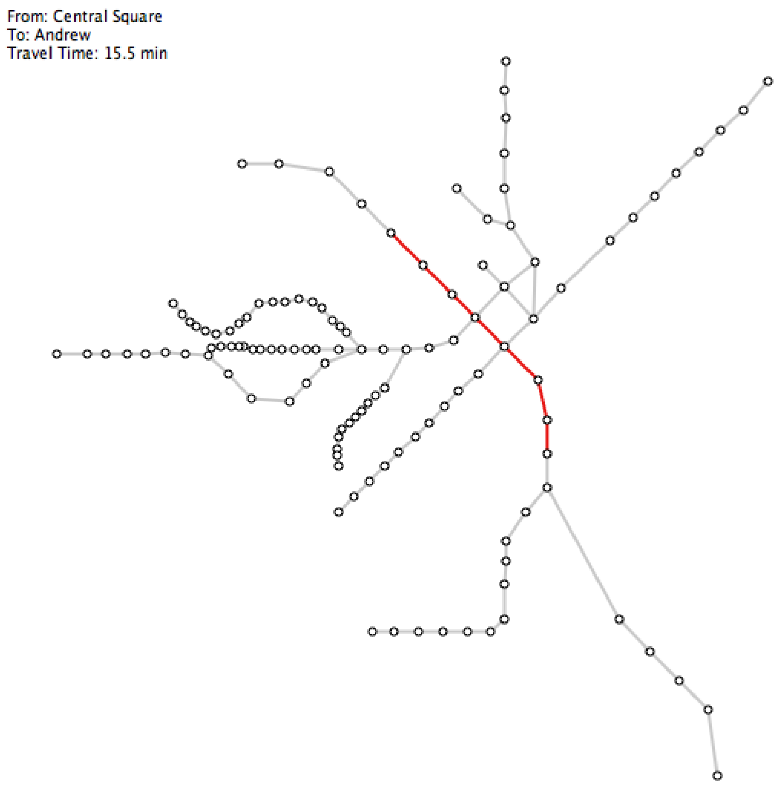
```
From: Central Square
To: Andrew
Travel Time: 15.5 min
```

That's it.

1.2.5 Color Effect

Save the previous sketch as "Ex_1_2_5".

Wouldn't it be nice to not keep the nonactive nodes and edges in the background? For example by coloring the edges in gray? We could animate this change in color.



To do so, download the Integrator class and add it do the sketch.

Use the *Integrator* class to animate the change in color. It's better to do this interpolation in the HSB color space. Leave the hue the same, but change the "target" saturation and "target" brightness for all the nonactive edges to 0 and to 200. Set this new "target" whenever a new shortest path is going to be computed. If the display changes again to display all the nodes, change the target edge color to its original color again: