EECE 5642 Assignment 4
Yao Zhou 001586311

1.2.1 Setting Up

The first figure shows the color setting, I set red, green, blue and orange with static final color as the HW4 pdf mentioned. As for the second figure, I write the *loadData()* method and add 'String colo' attributes into *addEdge()* method. And the fig1.3 was the picture I used draw() method to create output.pdf.

EX_1_21 | Edge | Node

```
1  static final color red=#E61310; //"r"=(230,19,16)
2  static final color green=#016842;//"g"=(1,104,66)
3  static final color blue=#00308C; //"b"=(0,48,140)
4  static final color orange=#FF8305;//"o"=(255,131,5)
5
6  class Edge {
7    Node from;
8    Node to;
9    float minutes;
10   String colo;
11
12   Edge(Node from, Node to, float minutes, String colo) {
13     this.from = from;
14     this.to = to;
15     this.minutes = minutes;
16     this.colo = colo;
17   }
18
19   Node getFromNode() {
```

Fig.1.1

EX_1_21 | Edge | Node

```
27 void loadData() {
28   addEdge("A", "B", 7.0, "r");
29   addEdge("B", "C", 7.0, "r");
30   addEdge("C", "D", 3.5, "g");
31   addEdge("D", "E", 3.0, "g");
32   addEdge("C", "F", 2.5, "b");
33   addEdge("F", "G", 1.2, "b");
34   addEdge("G", "H", 0.1, "o");
35 }
36
37 void addEdge(String fromLabel, String toLabel, float minutes, String colo) {
38   // find nodes
39   Node from = findNode(fromLabel);
40   Node to = findNode(toLabel);
41
42   // old edge?
43   for (int i = 0; i < edgeCount; i++) {
44     if (edges[i].from == from && edges[i].to == to) {
45       return;
```

Fig.1.2

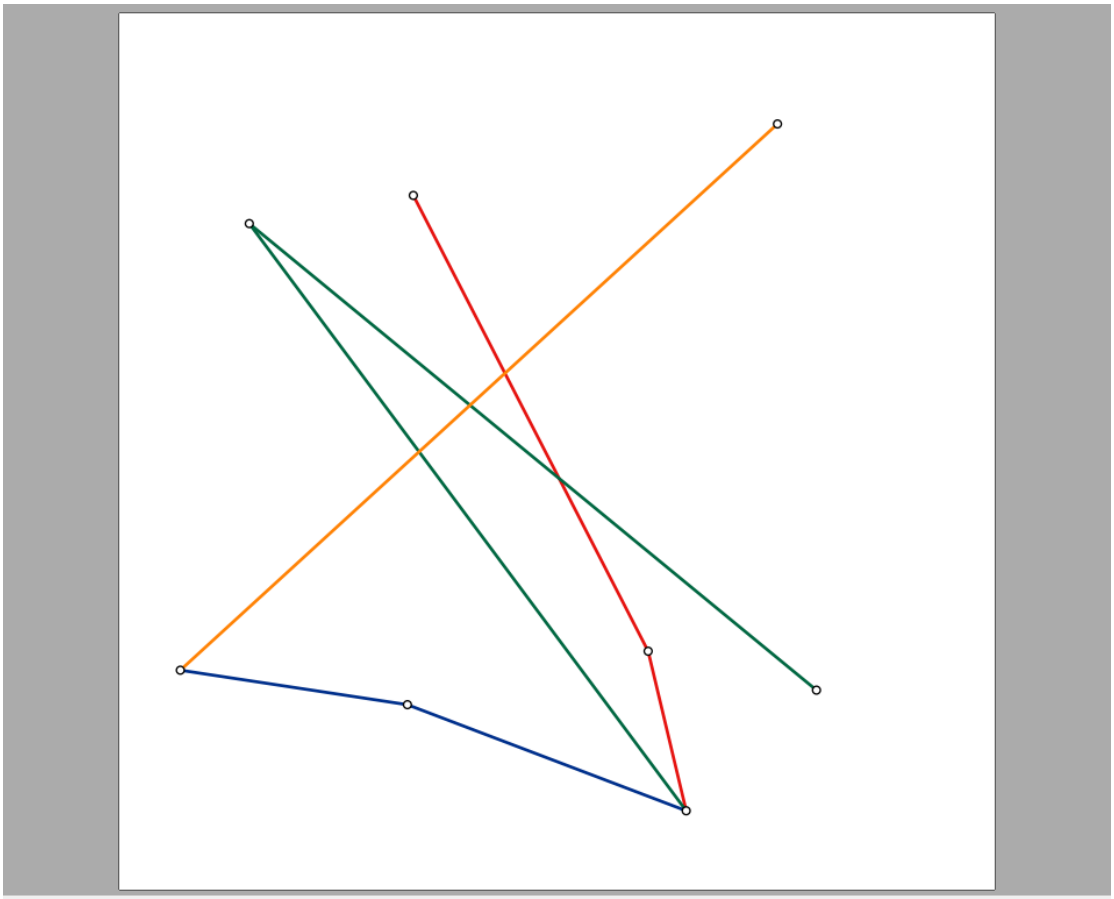

Fig.1.3

1.2.2 Acquiring the Data

This question first asks me to collect data from data.html and then create connections.csv and stations.csv to store the data. Based on this situation, I used python to create 2 python files stations.py and connections.py which import pandas to read the “data.html” and stored in stations.csv and connections.csv.


stations.py
connections.py
import pandas as pd
def main():
#file_path = "C:\Users\patto\Desktop\课程\dv5642\HW\HW4\HW4-1\HW4-code\data.html"
file_name = "data.html"
table = pd.read_html(file_name)
print("table count:", len(table))
data_frame1 = table[0]
data_frame2 = table[1]
#print(data_frame1)
data_frame1.to_csv("stations.csv", index = False)
if __name__ == "__main__":
main()
PyCharm 2021.3.3 available
Update...

Fig.2.1


connections.csv - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
#,From,To,Line,Minutes
1,Alewife,Davis,red,3.0
2,Davis,Porter Square,red,3.0
3,Porter Square,Harvard Square,red,3.0
4,Harvard Square,Central Square,red,3.5
5,Central Square,Kendall/MIT,red,3.5
6,Kendall/MIT,Charles/MGH,red,2.0
7,Charles/MGH,Park Street,red,2.0
8,Park Street,Downtown Crossing,red,1.5
9,Downtown Crossing,South Station,red,1.5
10,South Station,Broadway,red,2.5
11,Broadway,Andrew,red,2.5
12,Andrew,JFK/UMass,red,2.0
13,JFK/UMass,Savin Hill,red,2.5
14,Savin Hill,Fields Corner,red,2.5
15,Fields Corner,Shawmut,red,2.0
16,Shawmut,Ashmont,red,2.0
17,Ashmont,Cedar Grove,red,1.5
18,Cedar Grove,Butler,red,1.5
19,Butler,Milton,red,1.5
20,Milton,Central Avenue,red,1.5
21,Central Avenue,Valley Road,red,1.5
22,Valley Road,Capen Street,red,1.5
23,Capen Street,Mattapan,red,1.0
24,JFK/UMass,North Quincy,red,9.0
25,North Quincy,Wollaston,red,2.5
26,Wollaston,Quincy Center,red,2.5

Fig.2.2

stations.py connections.py

```python
import pandas as pd


def main():
    #file_path = "C:\Users\patto\Desktop\课程\dv5642\HW\HW4\HW4-1\HW4-code\data.html"
    file_name = "data.html"
    table = pd.read_html(file_name)
    print("table count:", len(table))

    data_frame1 = table[0]
    data_frame2 = table[1]

    #print(data_frame1)
    data_frame2.to_csv("connections.csv", index = False)

if __name__ == "__main__":
    main()
```

Fig.2.3

connections.csv - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
#,From,To,Line,Minutes
1,Alewife,Davis,red,3.0
2,Davis,Porter Square,red,3.0
3,Porter Square,Harvard Square,red,3.0
4,Harvard Square,Central Square,red,3.5
5,Central Square,Kendall/MIT,red,3.5
6,Kendall/MIT,Charles/MGH,red,2.0
7,Charles/MGH,Park Street,red,2.0
8,Park Street,Downtown Crossing,red,1.5
9,Downtown Crossing,South Station,red,1.5
10,South Station,Broadway,red,2.5
11,Broadway,Andrew,red,2.5
12,Andrew,JFK/UMass,red,2.0
13,JFK/UMass,Savin Hill,red,2.5
14,Savin Hill,Fields Corner,red,2.5
15,Fields Corner,Shawmut,red,2.0
16,Shawmut,Ashmont,red,2.0
17,Ashmont,Cedar Grove,red,1.5
18,Cedar Grove,Butler,red,1.5
19,Butler,Milton,red,1.5
20,Milton,Central Avenue,red,1.5
21,Central Avenue,Valley Road,red,1.5
22,Valley Road,Capen Street,red,1.5
23,Capen Street,Mattapan,red,1.0
24,JFK/UMass,North Quincy,red,9.0
25,North Quincy,Wollaston,red,2.5
26,Wollaston,Quincy Center,red,2.5
```

Fig.2.4

In Fig.2.5 which showed I converted map.gif into mbtamap.png and reset 900*900 size, and then used mouse left button to locate the Boston MBTA each station on the map

and recorded into locations.csv Fig.2.7

Ex_1_2_2 | Table

```
int currentRow = -1;
PrintWriter writer;

void setup() {
  size(900, 900);
  mapImage = loadImage("mbtamap.png");

  // read the csv file
  nameTable = new Table("stations.csv");
  writer = createWriter("locations.csv");
  cursor(CROSS);
  println("Click the mouse to begin.");
}

```

Fig.2.5

```
void mousePressed() {
  if (currentRow != -1) {
    String abbrev = nameTable.getRowName(currentRow);
    writer.println(abbrev + "," + mouseX + "," + mouseY);
  }
```

```
  // Write this table as a CSV file
  void write(PrintWriter writer) {
    for (int i = 0; i < rowCount; i++) {
      for (int j = 0; j < data[i].length; j++) {
        if (j != 0) {
          writer.print(",");
        }
        if (data[i][j] != null) {
          writer.print(data[i][j]);
        }
      }
      writer.println();
    }
    writer.flush();
  }
```

Fig.2.6

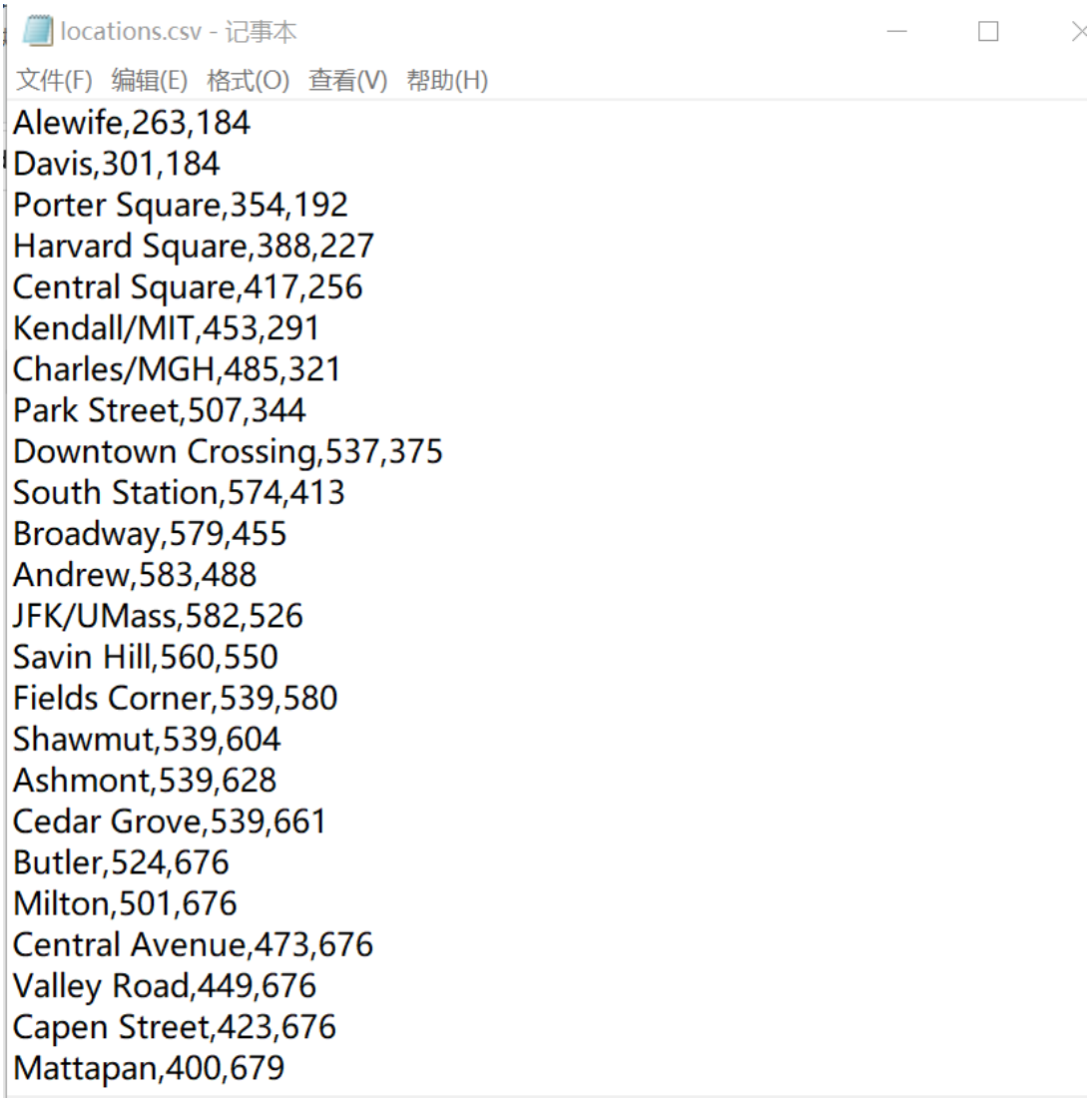

Fig.2.7

1.2.3 Visualization of the Network

In this question, we need to draw the picture of Boston MBTA map by using of the data we stored in *locations.csv.* In *connections.csv* file(Fig.2.4) we can get index from 1st column, and From station with 2nd column, To station with 3rd column. The 4th and 5th column are the line name and the spend time(unit minutes); According to the *locations.csv* we can see that the first column is station name, and the second column is X line position, the third column is Y line position. So I re-write the *addNode()* method, to identify float x, y by locations.csv 2nd and 3rd columns. And finally we draw the Fig.3.3 with my own data.

```
void loadData() {
 Table connectionsTable = new Table("connections.csv");
  for(int i=1; i<connectionsTable.getRowCount(); i++)
  {
    String fromPoint = connectionsTable.getString(i,1);
    String toPoint = connectionsTable.getString(i,2);
    String lineColor = connectionsTable.getString(i,3);
    float minutes = connectionsTable.getFloat(i,4);
    if(i<28){
      lineColor="r";
    }else if(i<45){
      lineColor="o";
    }else if(i<110){
      lineColor="g";
    }else{
      lineColor="b";
    }
```

Fig.3.1

```
Node addNode(String label) {
  Table nodesTable = new Table("locations.csv");
  int nodeIndex = nodesTable.getRowIndex(label);
  float x = nodesTable.getFloat(nodeIndex, 1);
  float y = nodesTable.getFloat(nodeIndex, 2);
  Node n = new Node(label, x, y, nodeCount);

  if (nodeCount == nodes.length) {
    nodes = (Node[]) expand(nodes);
  }
  nodeTable.put(label, n);
  nodes[nodeCount++] = n;
  return n;
}
```

Fig.3.2

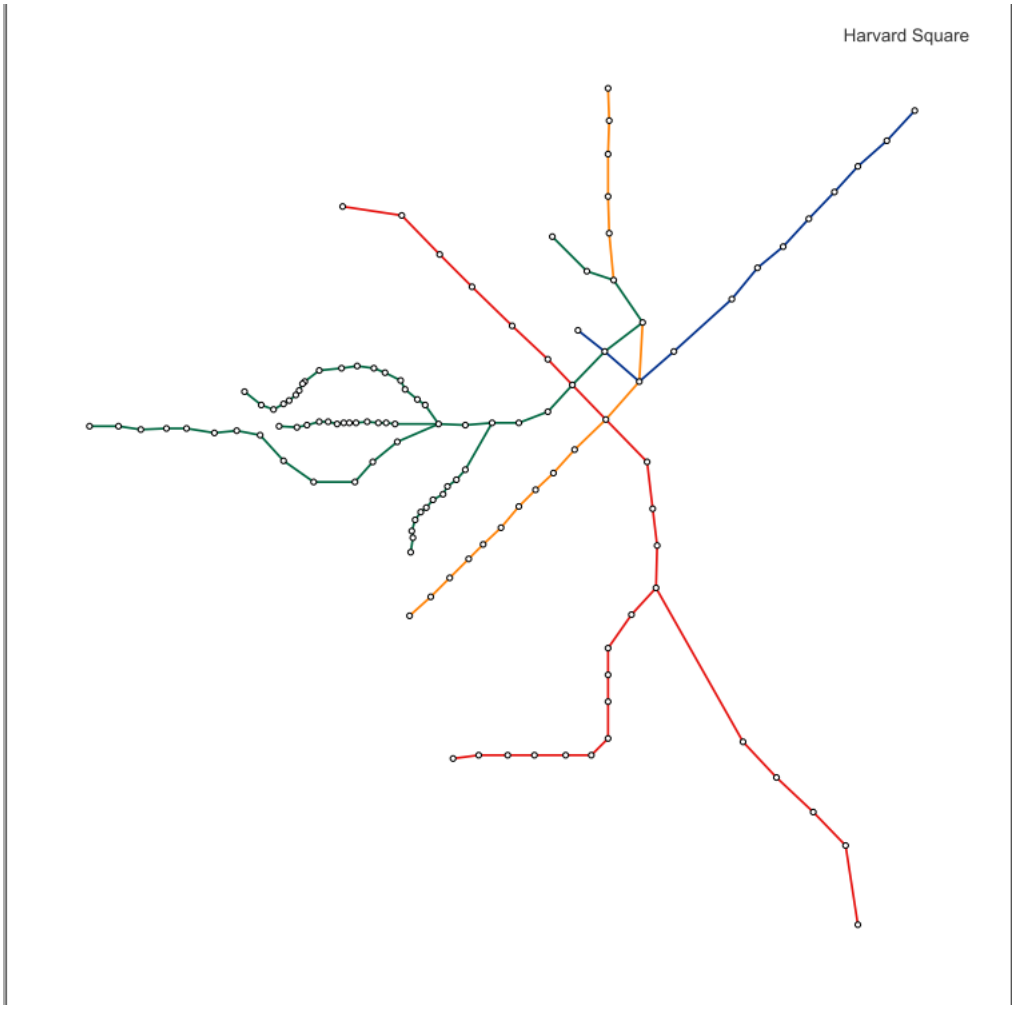

Fig.3.3

1.2.4 Shortest Path

In question 1.2.4; first I add two global arrays of type "*boolean*" which show as Fig.4.1, and also added *initializeActiveDataStructures()* in Fig.4.2 to monitored mouse movements. As for example, I chose Malden center station and Fenway station as from/to station and using Shortest Path file which wrote method Dijkstra's shortest path algorithm. The Fig.4.4 was the result of the Shortest Path.

EX_1_24 | Edge | Node | ShortestPath | Table

```
1 import processing.pdf.*;
2 //for Short path
3 boolean[] activeNodes;
4 boolean[] activeEdges;
5 Node A;
6 Node B;
7 int numOfNodes;
8 float numOfMinutes;
```

Fig.4.1

```
// draw the text of short path
if(numOfNodes==1){
  textAlign(LEFT, BOTTOM);
  textSize(16);
  fill(50);
  text("From:",10, 30);
  text(A.label, 55, 30);
}
if(numOfNodes==2){
  String travelTime  = nf(numOfMinutes,2,1);  //convert minute to 3 digital. step1.2
  textAlign(LEFT, BOTTOM);
  textSize(16);
  fill(50);
  text("From:",10, 30);
  text(A.label, 55, 30);
  text("To:",10, 50);
  text(B.label, 55, 50);
  textSize(16);
  text("Travel Time:",10, 70);
```

Fig.4.2

```
//calculate the short path
if (mouseButton == RIGHT) {
  float closest = 5;
  if(numOfNodes == 2){
    numOfNodes = 0;
    numOfMinutes = 0;
    initializeActiveDataStructures();
  }
  for (int i = 0; i < nodeCount; i++) {
    Node n = nodes[i];
    float d = dist(mouseX, mouseY, n.x, n.y);
    if (d < closest) {
      if(numOfNodes == 0){
        A = n;
        numOfNodes++;
      }
      else if (numOfNodes == 1){
        B = n;
        numOfMinutes = shortestPath(A.getIndex(),B.getIndex());
```

Fig.4.3

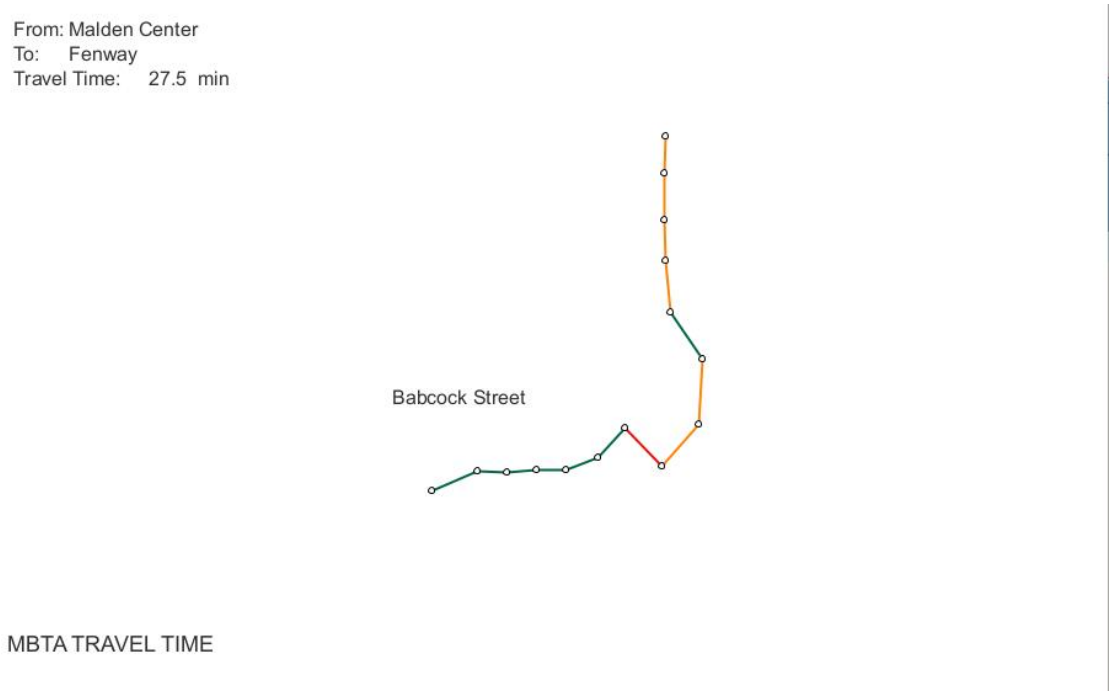

Fig.4.4

1.2.5 Color Effect

In this question, we need to keep the nonactive nodes and edges in the background and set them in gray color. To figure out this problem, I first set the grey color as Fig.4.1 shows. And then override the *draw()* method in Edge. And Node. File, first we need to identify what status they are. If the nodes and edges are active status, draw them with color red, green, blue and orange as usual, or draw them with grey color. To do this, I also downloaded the Integrator class. But, however, I didn't used it really. The Fig.5.4 was the result of this question. We can see I chose Harvard Square and Mass Ave as from/to station. After finding its shortest path, the map presented shortest path with red and orange color, and other nodes and edges presented with grey color.

```
static final color red=#E61310; //"r"=(230,19,16)
static final color green=#016842;//"g"=(1,104,66)
static final color blue=#00308C; //"b"=(0,48,140)
static final color orange=#FF8305;//"o"=(255,131,5)
static final color grey=#A5A3A3;//"grey"=(191,191,191)

class Edge {
  Node from;
  Node to;
  float minutes;
  String colo;

  Edge(Node from, Node to, float minutes, String colo) {
    this.from = from;
    this.to = to;
    this.minutes = minutes;
    this.colo = colo;
  }
```

Fig.5.1

EX_1_25 | Edge | Integrator | Node | ShortestPath | Table

```
    this.index = index;
  }

  int getIndex() {
    return index;
  }

  void draw(boolean activeStatus) {
    stroke(0);
    strokeWeight(1);
     if(activeStatus){
      fill(255);
    }else{
      fill(230);
    }
    ellipse(x, y, 5, 5);
  }
}
```

Fig.5.2

EX_1_25 | Edge | Integrator | Node | ShortestPath | Table

```
  void draw(boolean activeStatus) {
    if(activeStatus){
      switch(colo){
      case "r":
        stroke(red);
        break;
      case "g":
        stroke(green);
        break;
      case "b":
        stroke(blue);
        break;
      case "o":
        stroke(orange);
        break;
    }
  }
```

```
  else{
        switch(colo){
      case "r":
        stroke(grey);
        break;
      case "g":
        stroke(grey);
        break;
      case "b":
        stroke(grey);
        break;
      case "o":
        stroke(grey);
        break;
      }
```

Fig.5.3

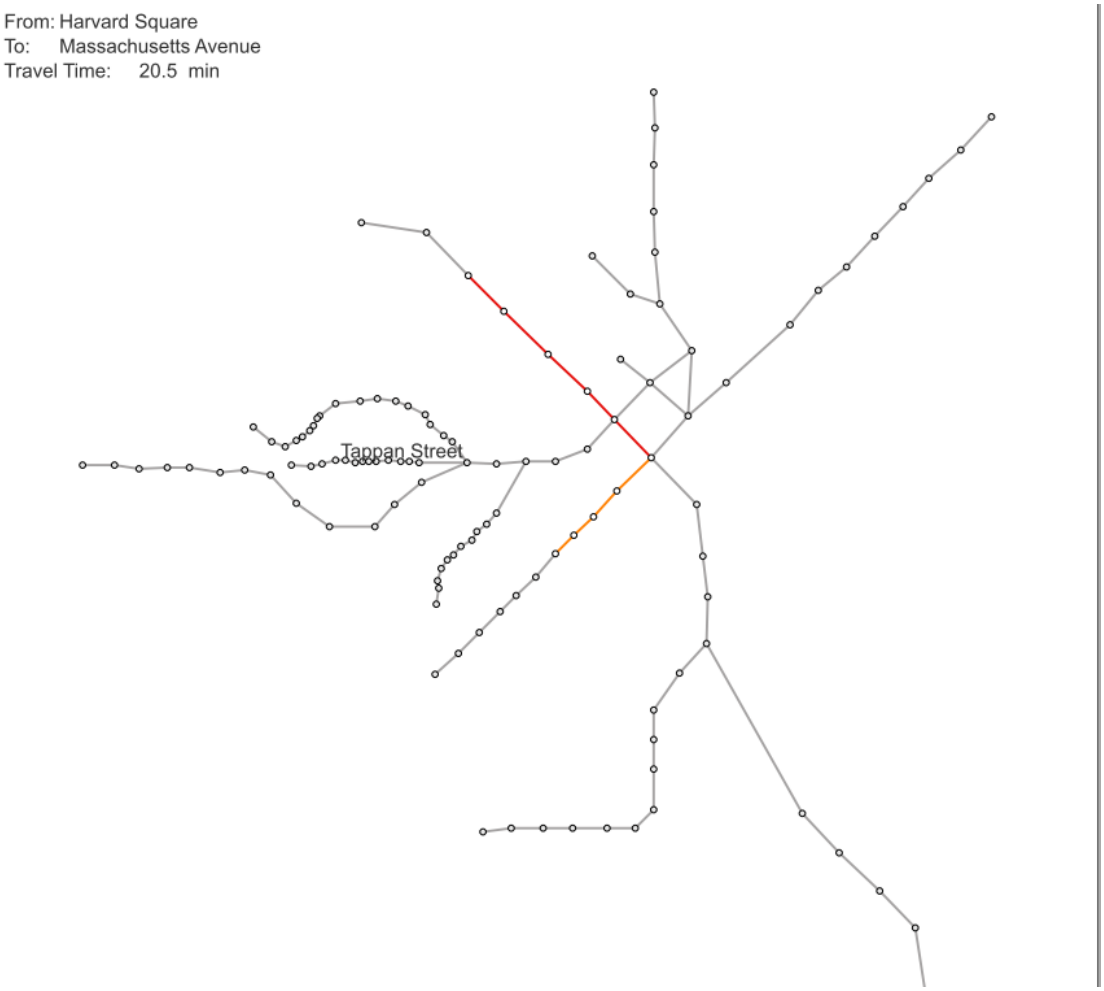
From: Harvard Square
To: Massachusetts Avenue
Travel Time: 20.5 min
Tappan Street

Fig.5.4