CPSC535-Project-1 Report

Index of Documentation:

Content in Github:

- Source Code
- README.md: contains the name of the team members.
- LICENSE: MIT License
- Pom.xml: contains the project information and configuration details that is used to build the project by maven.
- Project Report

1. Your names, CSUF-supplied email address(es), and an indication that the submission is for project 1.

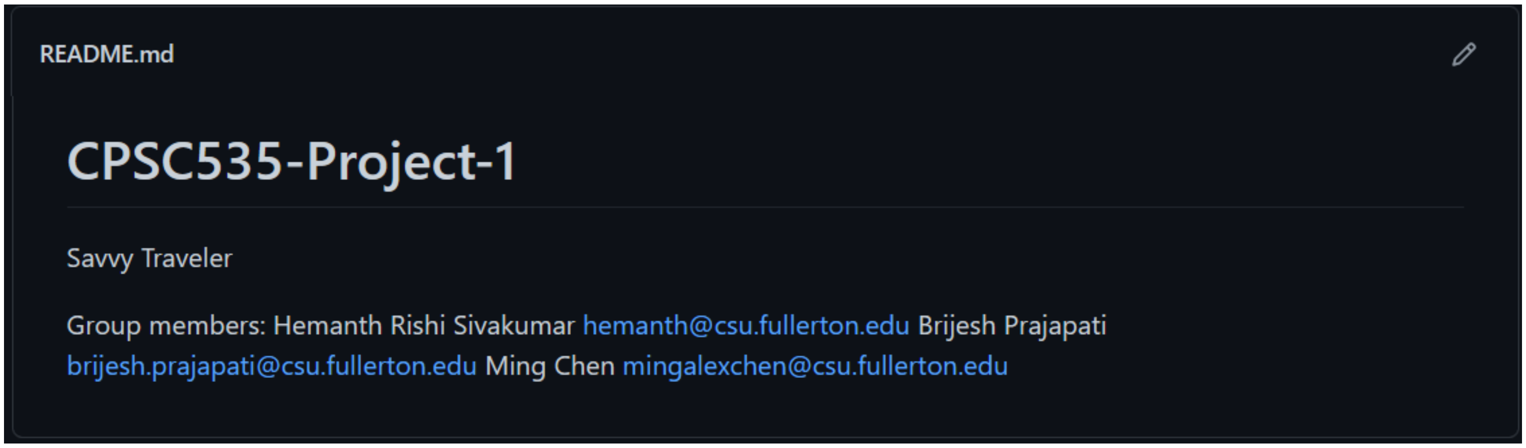
Project 1 Team Members:

Hemanth Rishi Sivakumar hemanth@csu.fullerton.edu

Brijesh Prajapati brijesh.prajapati@csu.fullerton.edu

Ming Chen mingalexchen@csu.fullerton.edu

2. A full-screen screenshot with your group member names shown clearly. One way to make your names appear in Atom is to simply open your README.md.


README.md

CPSC535-Project-1

Savvy Traveler

Group members: Hemanth Rishi Sivakumar hemanth@csu.fullerton.edu Brijesh Prajapati brijesh.prajapati@csu.fullerton.edu Ming Chen mingalexchen@csu.fullerton.edu

3. The pseudocode for the algorithms

Input/Output

Input:

- Graph with vertices and edges
- Start vertex- is the source vertex.
- End vertex- is the destination vertex.

Output:

- Returns the path with the highest probability from the source vertex to the end vertex.
- Returns the probability to the end vertex from the source vertex.
- Returns the most reliable destination vertex in the graph

Algorithm Pseudocode

Finding the highest probability path from a given startVertex to an endVertex

```
set the weight of startVertex to 1.0
add the startVertex to the priorityQueue
add the startVertex to the the list of visitedVertices
While(priorityQueue is not empty)
    currentVertex ← get the first vertex element from the priorityQueue (vertex element with the highest probability will be returned.)
    For all the edges of currentVertex
        adjacentVertex ← get the toVertex of the edge
        If list of visited vertices does not contain adjacentVertex
            newWeight ← currentVertex's weight * edge's cost
            If newWeight is greater than adjacentVertex's weight
              remove the adjacent Vertex from the priorityQueue
              setWeight of the adjacentVertex to newWeight
                  set adjacentVertex's PreviouseVertex property to currentVertex
              add the adjacentVertex to the priorityQueue

add currentVertex to the list of visitedVertices
For vertices from endVertex to startVertex
    add vertex's previousVertex to the shortestPath list
    if vertex is equal to startVertext
        break loop
reverse shortestPath list
return shortestPath list and probability of endVertex
```

Finding the most reliable travel destination in the graph

```
max ← 0
For fromVertex in vertices
    findHighestProbabilityPath from fromVertex to all other vertices
    For toVertex in vertices
        sum ← sum + get toVertex’s weight

    sum ← sum – 1
    If sum is greater than max
        maxVertex ← fromVertex
        max ← sum

resetGraphVertices’s weights and previousVertex
return maxVertex
```

4. A brief description of how to run the code.

Description of how to execute the code

1. Clone repository to a directory using the following command in git “git clone https://github.com/CSUF-CPSC-Bein-SP22/savvy-traveler-wanderers.git”
2. Navigate to the following directory “\savvy-traveler-wanderers\src\main\java”
3. Open command prompt.
4. Compile the project using the following command “javac Main.java”
5. Run the project using following command “java Main example1” where ‘example1’ is the graph from example 1. ‘example1’ can be replaced with ‘example2’, ‘example3’, ’example4’ and so on to construct graph of those examples and perform graph operations on them.

Instruction on how to change input value

Input values are hard coded in the program.

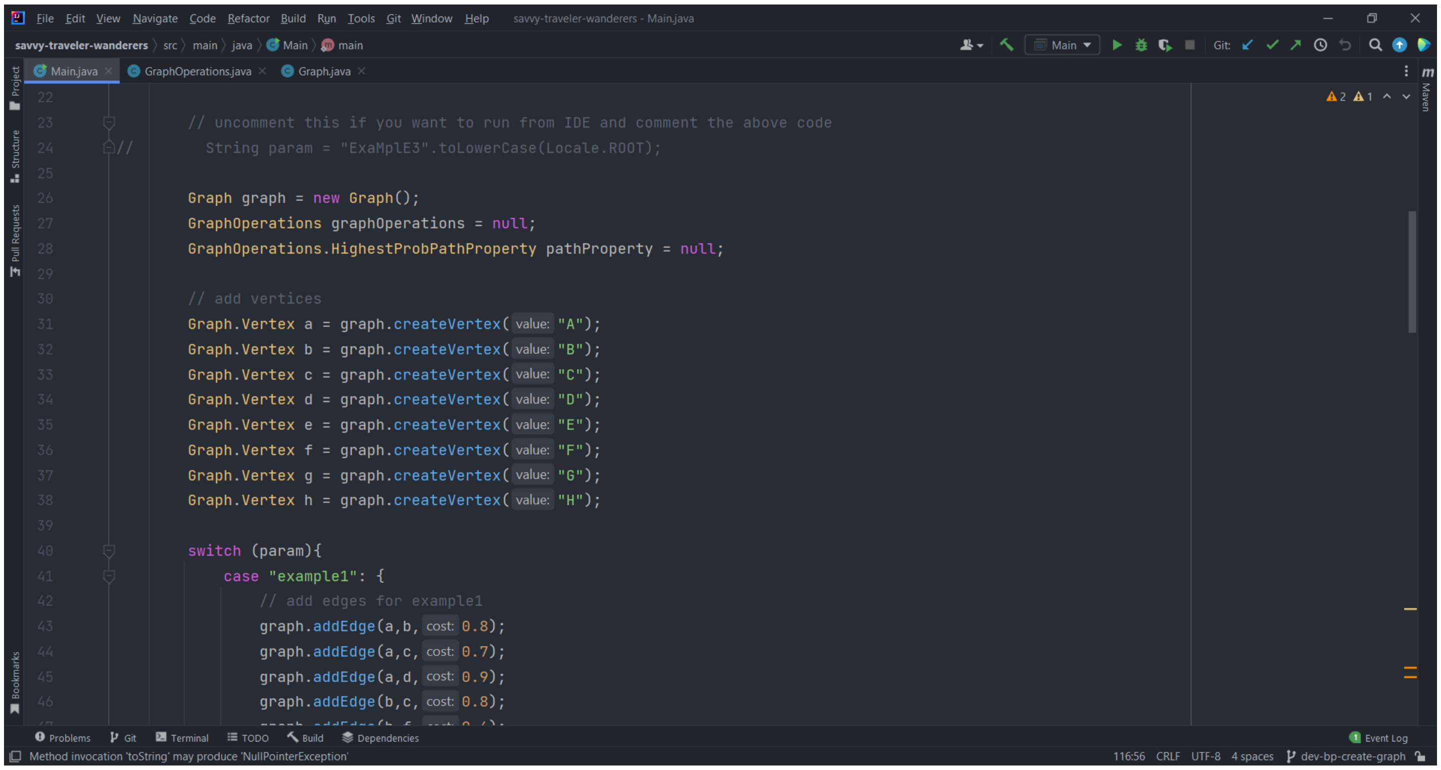
You may add vertices as you wish below the comments that says add vertices in Main.java file. You must also add edges and edge values as you wish in switch cases of the same file. There are some examples in that file, and it should be easy to add inputs there.

After adding input values manually into code, select the case you want to test by using the example name when running Main.

EX: java Main example1 runs data of example1 that is hard coded in the program, and java Main example4 runs data of example4 etc.

Here are some helping pictures

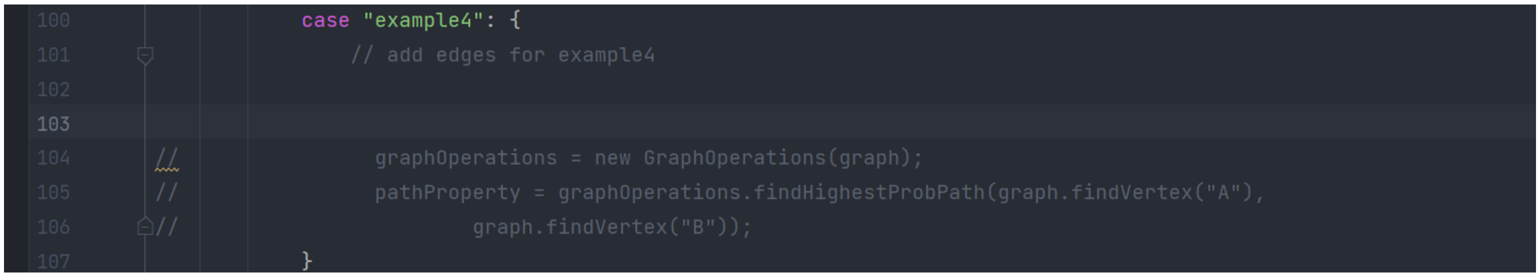
- Add vertices or remove vertices for new graph in the following location of Main.java file:


File Edit View Navigate Code Refactor Build Run Tools Git Window Help savvy-traveler-wanderers - Main.java
savvy-traveler-wanderers src main java Main main
Main.java GraphOperations.java Graph.java
Problems Git Terminal TODO Build Dependencies Event Log
116:56 CRLF UTF-8 4 spaces dev-bp-create-graph

```
        // uncomment this if you want to run from IDE and comment the above code
//        String param = "ExaMplE3".toLowerCase(Locale.ROOT);

        Graph graph = new Graph();
        GraphOperations graphOperations = null;
        GraphOperations.HighestProbPathProperty pathProperty = null;

        // add vertices
        Graph.Vertex a = graph.createVertex( value: "A");
        Graph.Vertex b = graph.createVertex( value: "B");
        Graph.Vertex c = graph.createVertex( value: "C");
        Graph.Vertex d = graph.createVertex( value: "D");
        Graph.Vertex e = graph.createVertex( value: "E");
        Graph.Vertex f = graph.createVertex( value: "F");
        Graph.Vertex g = graph.createVertex( value: "G");
        Graph.Vertex h = graph.createVertex( value: "H");

        switch (param){
            case "example1": {
                // add edges for example1
                graph.addEdge(a,b, cost: 0.8);
                graph.addEdge(a,c, cost: 0.7);
                graph.addEdge(a,d, cost: 0.9);
                graph.addEdge(b,c, cost: 0.8);
```

Method invocation 'toString' may produce 'NullPointerException'

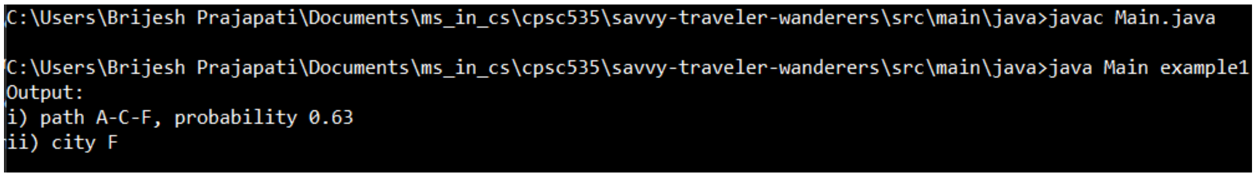
- Add edges of the graph in the following location and uncomment the graphOperations and pathProperty variables. Moreover, in the pathProperty edit the start vertex and end vertex.



```java
            case "example4": {
                // add edges for example4


//                graphOperations = new GraphOperations(graph);
//                pathProperty = graphOperations.findHighestProbPath(graph.findVertex("A"),
//                        graph.findVertex("B"));
            }
```

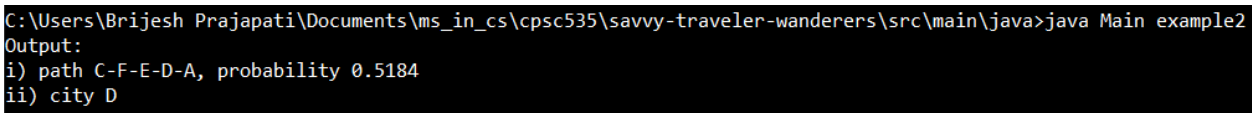
5. Three snapshots of code executing for the three given examples.

- Example 1 screen shot



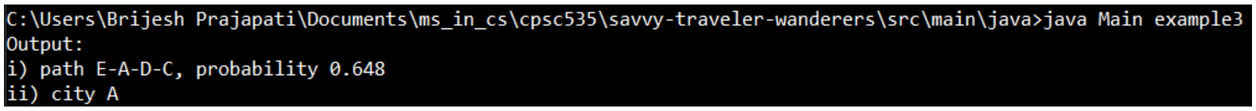
```
C:\Users\Brijesh Prajapati\Documents\ms_in_cs\cpsc535\savvy-traveler-wanderers\src\main\java>javac Main.java

C:\Users\Brijesh Prajapati\Documents\ms_in_cs\cpsc535\savvy-traveler-wanderers\src\main\java>java Main example1
Output:
i) path A-C-F, probability 0.63
ii) city F
```

- Example 2 screen shot



```
C:\Users\Brijesh Prajapati\Documents\ms_in_cs\cpsc535\savvy-traveler-wanderers\src\main\java>java Main example2
Output:
i) path C-F-E-D-A, probability 0.5184
ii) city D
```

- Example 3 screen shot



```
C:\Users\Brijesh Prajapati\Documents\ms_in_cs\cpsc535\savvy-traveler-wanderers\src\main\java>java Main example3
Output:
i) path E-A-D-C, probability 0.648
ii) city A
```