Name : Ramzy Izza Warhana
NIM : 21/472698/PA/20322
Class : IUP CS1

Assignment 8 Lab Algorithm and Data Structures

Minimum Spanning Tree

Complete Source Code in zip format :

https://drive.google.com/file/d/1uxdE0zzROqrV1EtcMzDWd952trpjqxN0/view?usp=sharing

1. Test the code with the graph from 8.2.2 and modify the code such that it also calculate the total cost of the MST. Did you get the same result compared to the theoretical one?

Main.java

```java
package PrimsAlgo;
public class Main {
    Run | Debug
    public static void main(String[] args){

        MST t = new MST();
        int graph[][] = new int[][]{{0,5,0,5,4,0,0,0,0,0},
                                    {5,0,3,0,7,0,6,8,0,0},
                                    {0,3,0,0,0,0,0,0,5,0},
                                    {5,0,0,0,0,4,0,0,0,0},
                                    {4,7,0,0,0,5,0,0,0,0},
                                    {0,0,0,4,5,0,3,0,0,0},
                                    {0,6,0,0,0,3,0,6,0,6},
                                    {0,8,0,0,0,0,6,0,7,0},
                                    {0,0,5,0,0,0,0,7,0,4},
                                    {0,0,0,0,0,0,6,0,4,0}};
        t.primMST(graph);
    }
}
```

MST.java

```java
package PrimsAlgo;

public class MST {
    public int cost;
    int V, prev[], value[], spanning = 2, mstWeight = 0;
    boolean MSTset[];

    MST(int vertex) {
        prev = new int[vertex];
        V = vertex;
    }

    public int minKey() {
        int min = Integer.MAX_VALUE;
        int minIndex = -1;
        int v = 0;
        while(v != V){
            if(MSTset[v] == false && value[v] < min){
                min = value[v];
                minIndex = v;
            }
            v++;
        }
        return minIndex;
    }

    public void printMST(int graph[][]) {
        System.out.println(x: "Edge \tWeight");
        mstWeight = 0;
        for(int i = 1; i < V; i++) {
            System.out.println((prev[i] + 1) + " - " + (i + 1) + "\t" + graph[i][prev[i]]);
            mstWeight += graph[i][prev[i]];
        }
        System.out.println("Total weight of the MST : " + mstWeight);
        cost = mstWeight;
    }

    public void primMST(int graph[][]) {
        value = new int[V];
        MSTset = new boolean[V];

        for(int i = 0; i < V; i++) {
            value[i] = Integer.MAX_VALUE;
            MSTset[i] = false;
        }
        value[0] = 0;
        prev[0] = -1;

        for(int i = 0; i < V - 1; i++) {
            int u = minKey();
            MSTset[u] = true;
            for(int v = 0; v < V; v++) {
                if(graph[u][v] != 0 && MSTset[v] == false && graph[u][v] < value[v]) {
                    prev[v] = u;
                    value[v] = graph[u][v];
                }
            }
        }
        printMST(graph);
    }
}
```

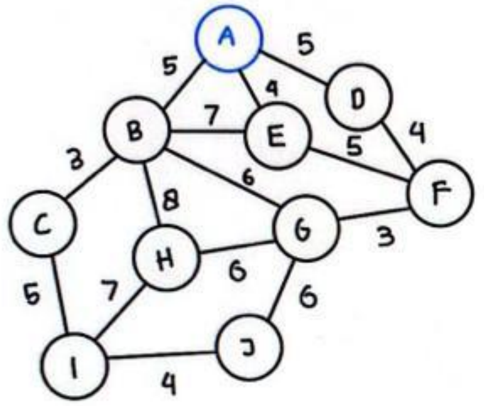
Theoretical Calculations:

Ramzy Izza W.
21/472698/PA/20322

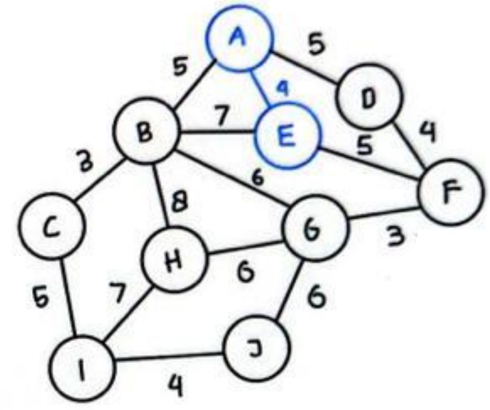
Prims Algorithm theory :

a. determine any starting vertex
b. find and select edge with smallest weight from the vertex that we've visited
c. Make sure that the edge will not form a cycle
d. repeat until all the vertex is visited

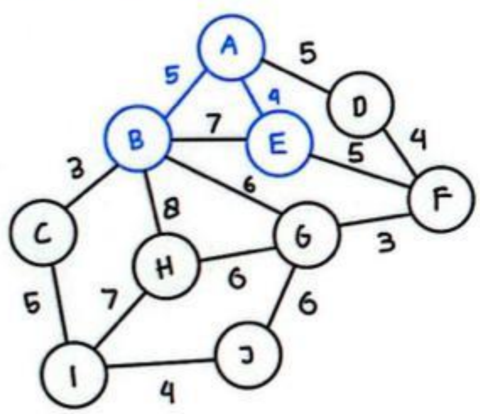
Step 1 ⇒ set A as starting node



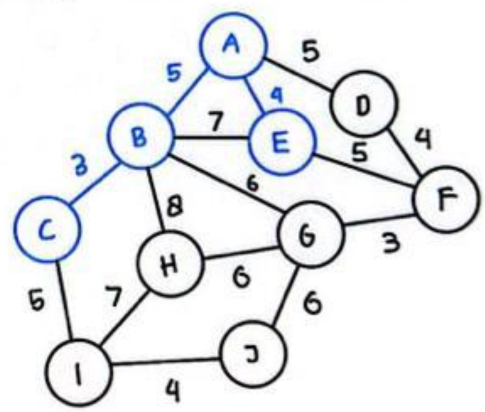
Step 2 ⇒ select edge with weight 4 (A-E)



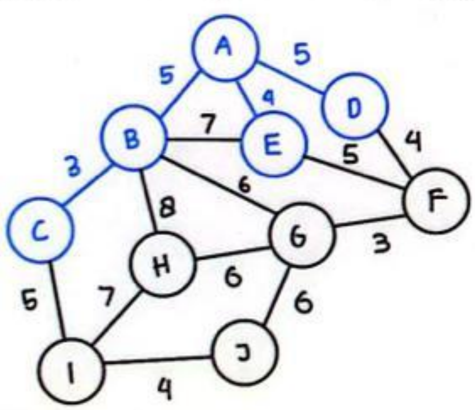
Step 3 ⇒ select edge with weight 5 (A-B)



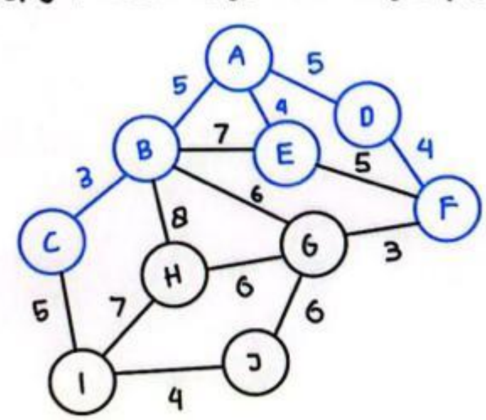
Step 4 ⇒ select edge with weight 3 (B-C)



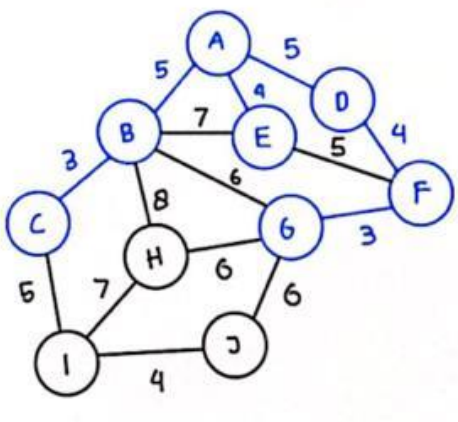
Step 5 ⇒ select edge with weight 5 (A-D)



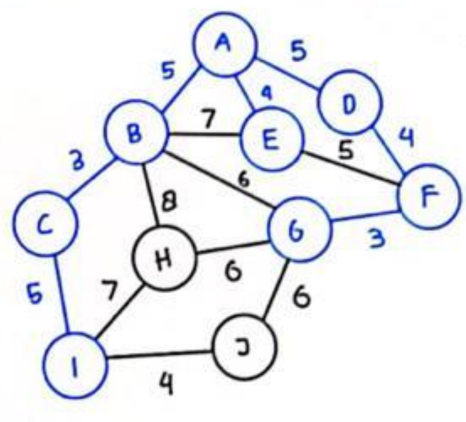
Step 6 ⇒ select edge with weight 4 (D-F)



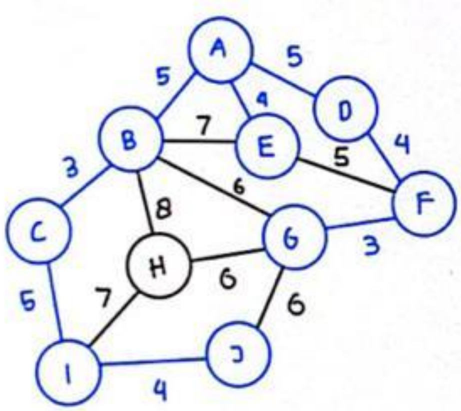
Step 7 ⇒ select edge with weight 3 (F - G)



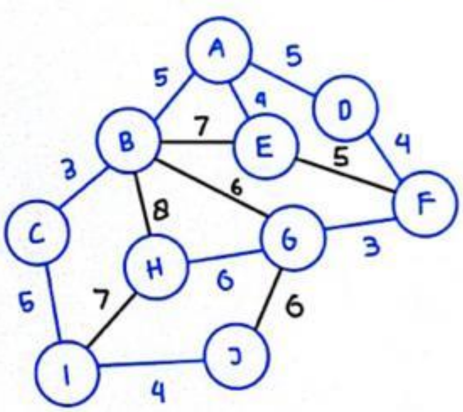
Step 8 ⇒ select edge with weight 5 (C - I)



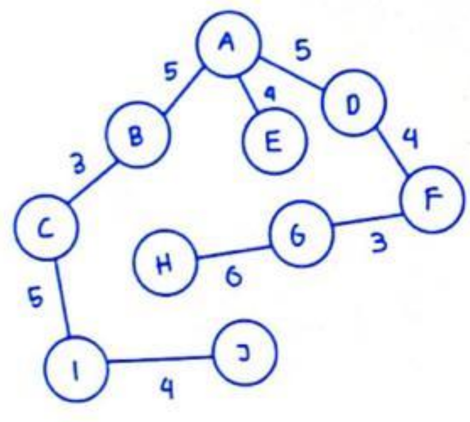
Step 9 ⇒ select edge with weight 4 (I - J)



Step 10 ⇒ select edge with weight 6 (H - G)



Step 11 ⇒ compute all the minimum weight



$5+5+4+4+3+3+6+5+4$

$= 39$

Hence, from graph 8.2.2 by using prims algorithm theorem it is shown that the total MST is equal to 39

Output Code:

```
Edge    Weight
1 - 2   5
2 - 3   3
1 - 4   5
1 - 5   4
4 - 6   4
6 - 7   3
7 - 8   6
3 - 9   5
9 - 10  4
Total MST: 39
```

From the screenshot given above, we can imply that calculating the prims algorithm manually has an equal value with the java implementation of prims algorithm according to the graph of 8.2.2

---

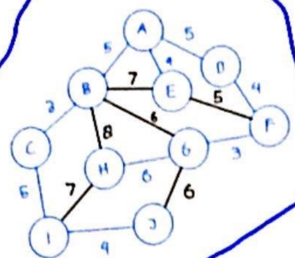
2. Another spanning tree (non-minimum) can be created from an MST by changing the edge connections. Write a program that output the number of spanning trees that can be created by changing the position of one edge only. Then, among these new spanning trees, output the minimum cost. Test your code on the graph from 8.2.2. Check the correctness by comparing the result with the one performed through manual calculation

Output

```
Problem 1 : Implement Graph 8.2.2 into adjacency martrix using prims algorithm
Edge    Weight
1 - 2   5
2 - 3   3
1 - 4   5
1 - 5   4
4 - 6   4
6 - 7   3
7 - 8   6
3 - 9   5
9 - 10  4
Total weight of the MST : 39

Problem 2 : Find other possible spanning tree non minimum by moving only one edge only

Among all spanning trees with total of 28 that had
been form from moving one edge, there exist one spanning tree that has minimum cost with value of 39
Therefore, the amount of possible non-minimum spanning trees is : 27
and from all of the non spanning trees, the lowest cost is  : 39

All the results from above implementation has been proved using manual calculation given on the lab report
```

Ramzy Izza W.
21/472698/PA/20322

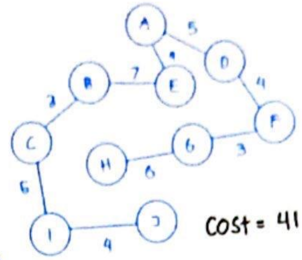
MST



Cost = 39

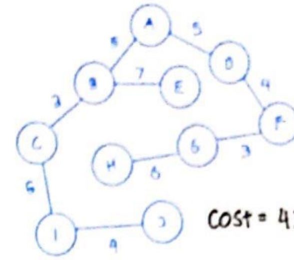
Observe edge B-E

Change A-B to B-E



Cost = 41

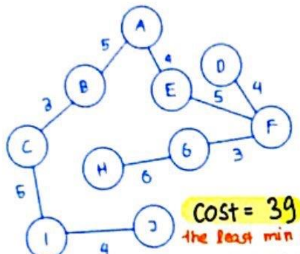
Change A-E to B-E



Cost = 42

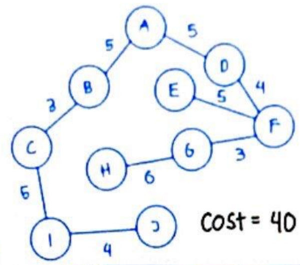
Observe edge E-F

Change A-D to E-F



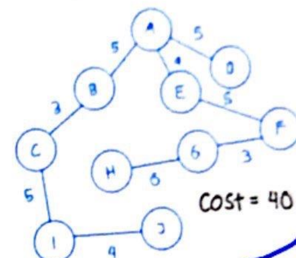
Cost = 39
the least min cost

Change A-E to E-F



Cost = 40

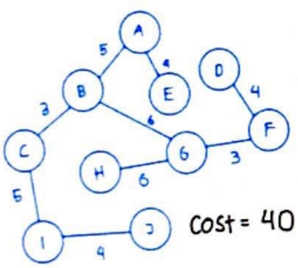
Change D-F to E-F



Cost = 40

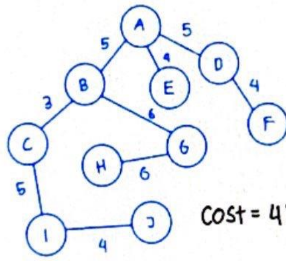
Observe edge B-G

Change A-D to B-G



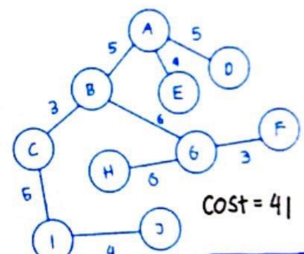
Cost = 40

Change G-F to B-G



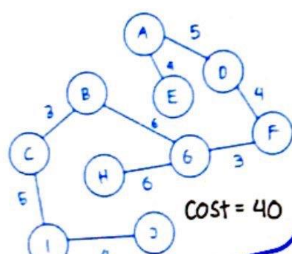
Cost = 42

Change D-F to B-G



Cost = 41

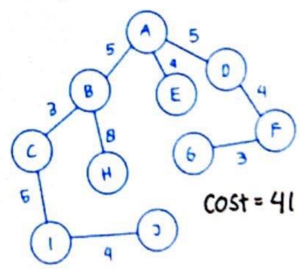
Change B-A to B-G



Cost = 40

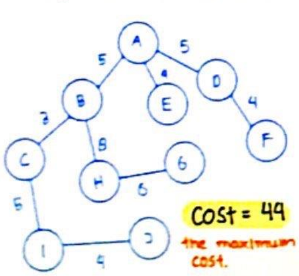
Observe edge B-H

Change H-G to B-H



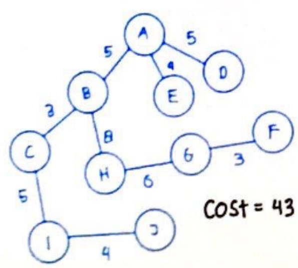
Cost = 41

Change G-F to B-H



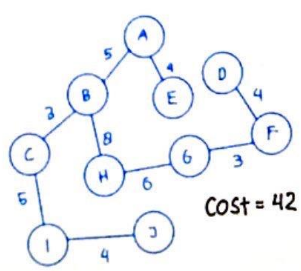
Cost = 49
the maximum cost.

Change D-F to B-H



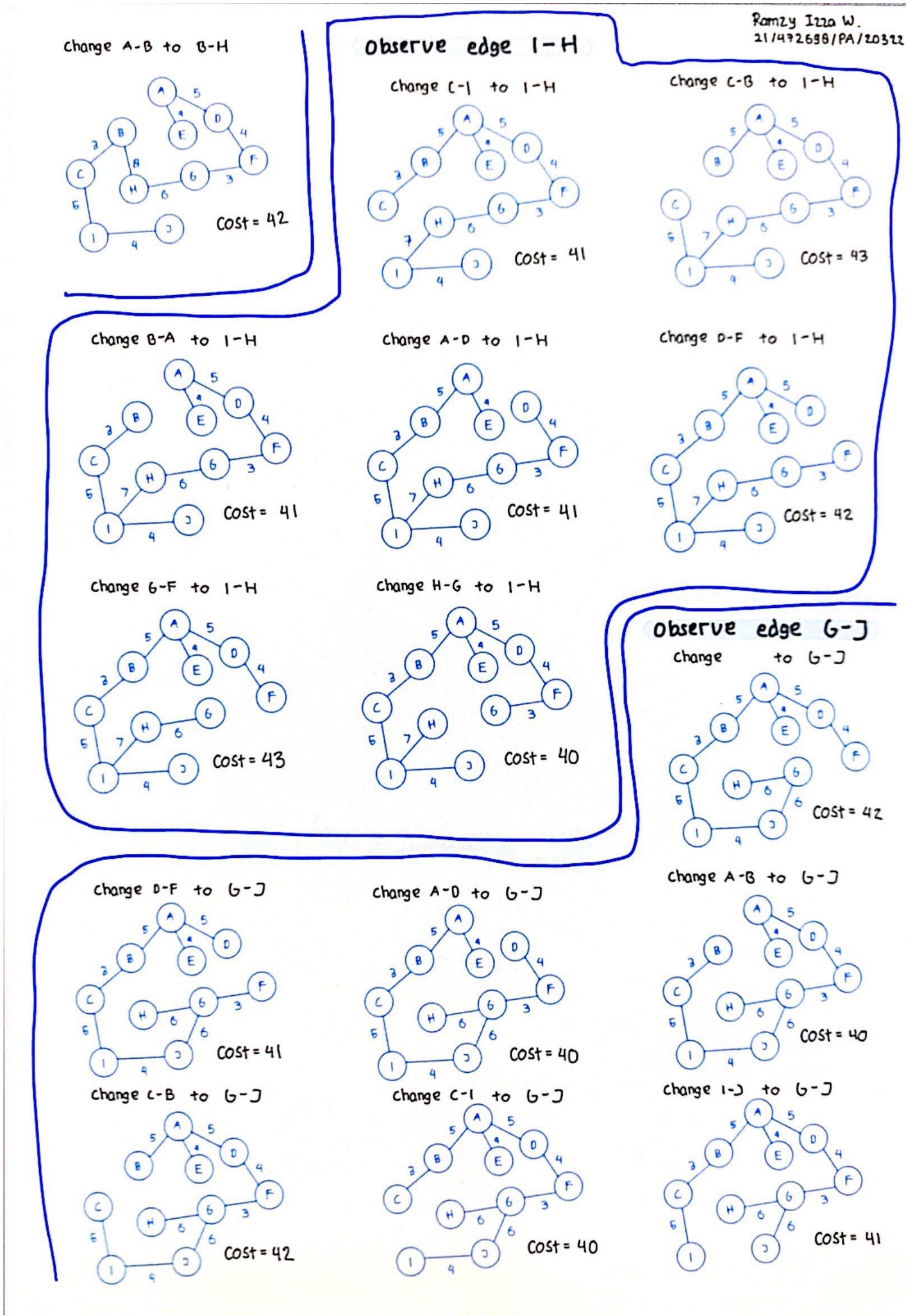
Cost = 43

Change A-D to B-H



Cost = 42

From the manual calculation that I have written here, it is shown that there are 28 possibilities of spanning tree that could be form by changing at most 1 edge only to another vertex (non minimum spanning). Calculating all of those 28 spanning trees, we get the minimum cost is equal to 39 and the maximum cost will be 44. Hence there exist 1 minimum spanning tree among the 28 spanning trees making the total of spanning tree (non-minimum) to be 27



FindOtherST.java

```java
package PrimsAlgo;

public class FindOtherST {

    int V, mstWeight, findMinimumCost;
    int prev[];
    boolean connect;

    FindOtherST(int V, int previous[], int mstWeight){
        this.V = V;
        connect = false;
        prev = previous;
        this.mstWeight = mstWeight;
    }

    public void nonMinimumST(int adjacencyMatrix[][]) {
        int totalSpanningTree = 2, i = 0, j = 0, k = 0;
        int temp[] = new int[V];
        findMinimumCost = mstWeight++;

        while(i != V){
            temp = prev.clone();
            temp[i] = -1;
            for(j = 0; j < V; j++) {
                for(k = 0; k < V; k++) {
                    if(temp[k] == -1 || adjacencyMatrix[i][j] == adjacencyMatrix[k][prev[k]] || adjacencyMatrix[i][j] == 0)
                        continue;
                    else if(adjacencyMatrix[i][j] == adjacencyMatrix[k][prev[k]])
                        continue;
                    else if(adjacencyMatrix[i][j] == 0)
                        continue;
                    else{
                        if(isConnect(i, j, k: 0, temp)) {
                            totalSpanningTree++;
                            if(findMinimumCost > mstWeight + 1 && mstWeight - adjacencyMatrix[i][prev[i]] + adjacencyMatrix[i][j] <= findMinimumCost) {
                                findMinimumCost = mstWeight + adjacencyMatrix[i][prev[i]] - adjacencyMatrix[i][j];
                            }
                        }
                    }
                }
            }
            i++;
        }
        System.out.println("Among all spanning trees with total of "+ (int)(totalSpanningTree+1) + " that had\nbeen form from moving one edge, there exist one spanning tree that has m
        System.out.println("Therefore, the amount of possible non-minimum spanning trees is : " + totalSpanningTree);
        System.out.println("and from all of the non spanning trees, the lowest cost is  : " + findMinimumCost);
    }

    public boolean isConnect(int i, int j, int k, int temp[]) {
        for(; k < V; k++) {
            if(k == temp[i]) {
                connect = true; continue;
            }
            else if(k == i || k == j) {
                connect = true; continue;
            }
            connect = false;
        }
        return connect;
    }
}
```