LAB ASSESSMENT 3

NAME: KAMRAN ANSARI

REG NO: 22MCA0223

Write a program to find the missing value of the given graph when Minimum Spanning tree total cost is given for the given graph MST cost is 17 try to find the missing edge value (F,E). By finding the MST without (F,E) edge value obtained is 15. Subtract the value with total MST to find the missing edge value.

DisjointSet.java

```
public class DisjointSet {
    private int parentArray[];
    private int numberOfElements;

    DisjointSet(int numberOfElements) {
        this.numberOfElements = numberOfElements;
        parentArray = new int[numberOfElements];

        for (int i = 0; i < numberOfElements; i++) {
            parentArray[i] = i;
        }
    }

    int parent(int e) {
        while (parentArray[e] != e) {
            e = parentArray[e];
        }

        return e;
    }
```

```
    void union(int e1, int e2) {
        int parente1 = parent(e1);
        int parente2 = parent(e2);

        if (parente1 < parente2) {
            parentArray[parente2] = parente1;
        } else {
            parentArray[parente1] = parente2;
        }
    }
}
```

Graph.java

```
import java.util.ArrayList;

public class Graph {
    private int adjacencyMatrix[][];
    public int totalVertices;

    Graph(int totalVertices) {
        this.totalVertices = totalVertices;
        adjacencyMatrix = new int[totalVertices][totalVertices];
    }

    void addEdge(int from, int to, int weight) {
        adjacencyMatrix[from][to] = weight;
    }

    void printGraph() {
```

```
        System.out.println("Adjacency Matrix: ");
        System.out.print("  ");
        for (int i = 0; i < totalVertices; i++) {
            System.out.print(((char) (65 + i)) + "\t");
        }
        for (int i = 0; i < totalVertices; i++) {
            System.out.println("");
            System.out.print(((char) (65 + i)) + " ");
            for (int j = 0; j < totalVertices; j++) {
                System.out.print(adjacencyMatrix[i][j] + "\t");
            }
        }
    }

    int mstCost(int toExclude) {
        ArrayList<ArrayList<Integer>> edgeList = new ArrayList<>();
        DisjointSet componentSet = new DisjointSet(totalVertices);

        for (int i = 0; i < totalVertices; i++) {
            for (int j = i + 1; j < totalVertices; j++) {
                if (adjacencyMatrix[i][j] == 0 || i == toExclude ||
j == toExclude) {
                    continue;
                }
                ArrayList<Integer> triple = new ArrayList<>();
                triple.add(adjacencyMatrix[i][j]);
                triple.add(i);
                triple.add(j);
                edgeList.add(triple);
            }
```

```
        }

        for (int i = 0; i < edgeList.size(); i++) {
            ArrayList<Integer> smallest = edgeList.get(i);
            int smallestIdx = i;
            for (int j = i + 1; j < edgeList.size(); j++) {
                if (edgeList.get(j).get(0) < smallest.get(0)) {
                    smallest = edgeList.get(j);
                    smallestIdx = j;
                }
            }
            edgeList.set(smallestIdx, edgeList.get(i));
            edgeList.set(i, smallest);
        }

        int cost = 0;
        for (int i = 0; i < edgeList.size(); i++) {
            if (componentSet.parent(edgeList.get(i).get(1)) ==
componentSet.parent(edgeList.get(i).get(2))) {
                continue;
            }

            componentSet.union(edgeList.get(i).get(1),
edgeList.get(i).get(2));
            cost += edgeList.get(i).get(0);
        }

        return cost;
    }
}
```

Main.java

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        int totalVertices;
        int totalCost;

        System.out.println("Enter number of vertices: ");
        totalVertices = Integer.valueOf(scan.nextLine());
        Graph graph = new Graph(totalVertices);

        for (int i = 0; i < totalVertices; i++) {
            for (int j = 0; j <= i; j++) {
                System.out.println("Enter weight of edge between " +
((char) (65 + i)) + " to " + ((char) (65 + j)));
                int weight = Integer.valueOf(scan.nextLine());
                graph.addEdge(i, j, weight);
                graph.addEdge(j, i, weight);
            }
        }

        graph.printGraph();

        System.out.println("");
        System.out.println("Enter minimum spanning tree total cost:
");
        totalCost = Integer.valueOf(scan.nextLine());
        System.out.println("Enter node to ignore: ");
```

```
        int nodeToIgnore = scan.nextLine().charAt(0) - 'A';


        int costWithoutNode = graph.mstCost(nodeToIgnore);
        System.out.println(
                "Minimum spanning tree cost without node " + ((char)
(nodeToIgnore + 65)) + " -> " + costWithoutNode);
        System.out.println("Weight of missing edge -> " + totalCost
+ " - " + costWithoutNode + " = "
                + (totalCost - costWithoutNode));
    }
}
```

```
Enter number of vertices:
6
Enter weight of edge between A to A
0
Enter weight of edge between B to A
7
Enter weight of edge between B to B
0
Enter weight of edge between C to A
8
Enter weight of edge between C to B
3
Enter weight of edge between C to C
0
Enter weight of edge between D to A
0
Enter weight of edge between D to B
6
Enter weight of edge between D to C
4
Enter weight of edge between D to D
0
Enter weight of edge between E to A
0
Enter weight of edge between E to B
0
```

```
Enter weight of edge between E to C
0
Enter weight of edge between E to D
5
Enter weight of edge between E to E
0
Enter weight of edge between F to A
0
Enter weight of edge between F to B
5
Enter weight of edge between F to C
3
Enter weight of edge between F to D
2
Enter weight of edge between F to E
0
Enter weight of edge between F to F
0
```

```
Adjacency Matrix:
  A     B       C       D       E       F
A 0     7       8       0       0       0
B 7     0       3       6       0       5
C 8     3       0       4       0       3
D 0     6       4       0       5       2
E 0     0       0       5       0       0
F 0     5       3       2       0       0
Enter minimum spanning tree total cost:
17
Enter node to ignore:
E
Minimum spanning tree cost without node E -> 15
Weight of missing edge -> 17 - 15 = 2
```