DSA ASSIGNMENT - 11

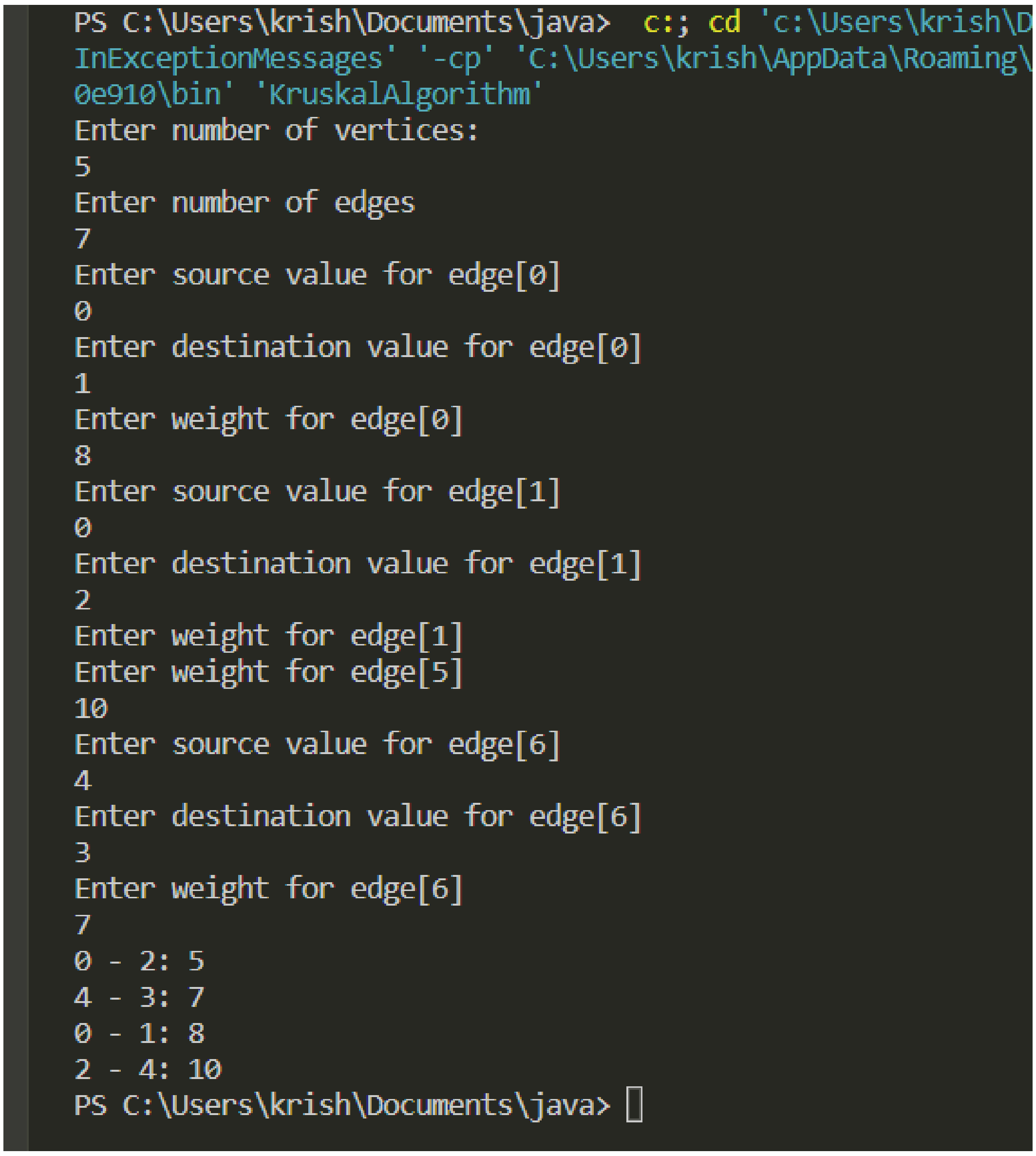
21BCE7371
RADHA KRISHNA GARG

<u>INPUT</u>

```
import java.util.Arrays;
import java.util.Scanner;

public class KruskalAlgorithm {
    class Edge implements Comparable<Edge> {
        int source, destination, weight;

        public int compareTo(Edge edgeToCompare) {
            return this.weight - edgeToCompare.weight;
        }
    };

    class Subset {
        int parent, value;
    };


    int vertices, edges;
    Edge edgeArray[];


    KruskalAlgorithm(int vertices, int edges) {
        this.vertices = vertices;
        this.edges = edges;
        edgeArray = new Edge[this.edges];
        for (int i = 0; i < edges; ++i)
            edgeArray[i] = new Edge();
    }


    void applyKruskal() {


        Edge finalResult[] = new Edge[vertices];
        int newEdge = 0;
        int index = 0;
        for (index = 0; index < vertices; ++index)
```

```
            finalResult[index] = new Edge();


        Arrays.sort(edgeArray);


        Subset subsetArray[] = new Subset[vertices];


        for (index = 0; index < vertices; ++index)
            subsetArray[index] = new Subset();


        for (int vertex = 0; vertex < vertices; ++vertex) {
            subsetArray[vertex].parent = vertex;
            subsetArray[vertex].value = 0;
        }
        index = 0;


        while (newEdge < vertices - 1) {

            Edge nextEdge = new Edge();
            nextEdge = edgeArray[index++];

            int nextSource = findSetOfElement(subsetArray,
nextEdge.source);
            int nextDestination = findSetOfElement(subsetArray,
nextEdge.destination);


            if (nextSource != nextDestination) {
                finalResult[newEdge++] = nextEdge;
                performUnion(subsetArray, nextSource, nextDestination);
            }
        }
        for (index = 0; index < newEdge; ++index)
            System.out.println(finalResult[index].source + " - " +
finalResult[index].destination + ": " + finalResult[index].weight);
    }


    int findSetOfElement(Subset subsetArray[], int i) {
        if (subsetArray[i].parent != i)
            subsetArray[i].parent = findSetOfElement(subsetArray,
subsetArray[i].parent);
        return subsetArray[i].parent;
    }


    void performUnion(Subset subsetArray[], int sourceRoot, int
destinationRoot) {

        int nextSourceRoot = findSetOfElement(subsetArray, sourceRoot);
        int nextDestinationRoot = findSetOfElement(subsetArray,
destinationRoot);

        if (subsetArray[nextSourceRoot].value <
subsetArray[nextDestinationRoot].value)
            subsetArray[nextSourceRoot].parent = nextDestinationRoot;
        else if (subsetArray[nextSourceRoot].value >
subsetArray[nextDestinationRoot].value)
            subsetArray[nextDestinationRoot].parent = nextSourceRoot;
        else {
            subsetArray[nextDestinationRoot].parent = nextSourceRoot;
            subsetArray[nextSourceRoot].value++;
        }
    }


    public static void main(String[] args) {

        int v, e;

        Scanner sc = new Scanner(System.in);


        System.out.println("Enter number of vertices: ");


        v = sc.nextInt();


        System.out.println("Enter number of edges");

        e = sc.nextInt();
        KruskalAlgorithm graph = new KruskalAlgorithm(v, e);
        for(int i = 0; i < e; i++){
            System.out.println("Enter source value for edge["+ i +"]");
            graph.edgeArray[i].source = sc.nextInt();
            System.out.println("Enter destination value for edge["+ i
+"]");
            graph.edgeArray[i].destination = sc.nextInt();
            System.out.println("Enter weight for edge["+i+"]");
            graph.edgeArray[i].weight = sc.nextInt();
        }
        graph.applyKruskal();
    }
}
```

OUTPUT



```
PS C:\Users\krish\Documents\java>  c:; cd 'c:\Users\krish\D
InExceptionMessages' '-cp' 'C:\Users\krish\AppData\Roaming\
0e910\bin' 'KruskalAlgorithm'
Enter number of vertices:
5
Enter number of edges
7
Enter source value for edge[0]
0
Enter destination value for edge[0]
1
Enter weight for edge[0]
8
Enter source value for edge[1]
0
Enter destination value for edge[1]
2
Enter weight for edge[1]
Enter weight for edge[5]
10
Enter source value for edge[6]
4
Enter destination value for edge[6]
3
Enter weight for edge[6]
7
0 - 2: 5
4 - 3: 7
0 - 1: 8
2 - 4: 10
PS C:\Users\krish\Documents\java> 
```