DESIGN ANALYSIS AND ALGORITHMS

LAB ASSIGNMENT – 1

NAME : PRATHAPANI SATWIKA

REG.NO. : 20BCD7160

1) USING JAVA IMPLEMENT THE KRUSKALS ALGORITHMS

CODE :

```
package Lab1a;
import java.util.*;
public class Kruskals {
            static class Edge {
                    int src, destination, weight;
                    public Edge(int src, int destination, int weight) {
                          this.src = src;
                        this.destination = destination;
                        this.weight = weight;
                    }
             }
             static class Graph1 {
                    int vertices;
                    ArrayList<Edge> aEdges = new ArrayList<>();
                    Graph1(int vertices) {
                          this.vertices = vertices;
                    }
                    public void addEdge(int src, int destination, int weight) {
                          Edge edge = new Edge(src, destination, weight);
                        aEdges.add(edge);
                    }
                    public void kruskalMST(){
                    PriorityQueue<Edge> pq = new PriorityQueue<>(aEdges.size(),
                                Comparator.comparingInt(o -> o.weight));
               for (int i = 0; i <aEdges.size() ; i++) {
                    pq.add(aEdges.get(i));
                    }
               int [] parent = new int[vertices];
               makeSet(parent);
                    ArrayList<Edge> mst = new ArrayList<>();
                    int index = 0;
                    while(index < (vertices-1)){
                          Edge edge = pq.remove();
                        int x_set = find(parent, edge.src);
                        int y_set = find(parent, edge.destination);
                        if(x_set==y_set){
                        }
                        else {
                           mst.add(edge);
                            index++;
```

```
                        union(parent,x_set,y_set);
                    }
                }
                System.out.println("Minimum Spanning Tree(MST): ");
                printGraph(mst);
            }
            public void makeSet(int [] parent){
                for (int i = 0; i <vertices ; i++) {
                    parent[i] = i;
                }
            }
            public int find(int [] parent, int vertex){
                if(parent[vertex]!=vertex)
                return find(parent, parent[vertex]);;
                return vertex;
            }
          public void union(int [] parent, int x, int y){
                int x_set_parent = find(parent, x);
                int y_set_parent = find(parent, y);
                parent[y_set_parent] = x_set_parent;
            }
            public void printGraph(ArrayList<Edge> edgeList){
                for (int i = 0; i <edgeList.size() ; i++) {
                    Edge edge = edgeList.get(i);
                   System.out.println("Edge-" + i + " src: " + edge.src + "
destination: " + edge.destination +" weight: " + edge.weight);
                }
            }
        }
            public static void main(String[] args) {
                int vertices = 6;
                Graph1 graph = new Graph1(vertices);
                graph.addEdge(0, 1, 2);
                graph.addEdge(1, 2, 3);
                graph.addEdge(3, 5, 4);
                graph.addEdge(5, 4, 2);
                graph.addEdge(1, 3, 4);
                graph.addEdge(3, 5, 2);
                graph.addEdge(4, 5, 6);
                graph.kruskalMST();

            }
        }
```

```java
package Lab1a;
import java.util.*;
public class Kruskals {
         static class Edge {
             int src, destination, weight;
             public Edge(int src, int destination, int weight) {
                 this.src = src;
                 this.destination = destination;
                 this.weight = weight;
             }
         }
         static class Graph1 {
             int vertices;
             ArrayList<Edge> aEdges = new ArrayList<>();
             Graph1(int vertices) {
                 this.vertices = vertices;
             }
             public void addEdge(int src, int destination, int weight) {
                 Edge edge = new Edge(src, destination, weight);
                 aEdges.add(edge);
             }
             public void kruskalMST(){
             PriorityQueue<Edge> pq = new PriorityQueue<>(aEdges.size(),
                     Comparator.comparingInt(o -> o.weight));
             for (int i = 0; i <aEdges.size() ; i++) {
                 pq.add(aEdges.get(i));
             }
             int [] parent = new int[vertices];
             makeSet(parent);
             ArrayList<Edge> mst = new ArrayList<>();
             int index = 0;
             while(index < (vertices-1)){
                 Edge edge = pq.remove();
                 int x_set = find(parent, edge.src);
                 int y_set = find(parent, edge.destination);
                 if(x_set==y_set){
                 }
                 else {
                     mst.add(edge);
                     index++;
                     union(parent,x_set,y_set);
                 }
             }
```

```
43              }
44              System.out.println("Minimum Spanning Tree(MST): ");
45              printGraph(mst);
46          }
47          public void makeSet(int [] parent){
48              for (int i = 0; i <vertices ; i++) {
49                  parent[i] = i;
50              }
51          }
52          public int find(int [] parent, int vertex){
53              if(parent[vertex]!=vertex)
54              return find(parent, parent[vertex]);;
55              return vertex;
56          }
57          public void union(int [] parent, int x, int y){
58              int x_set_parent = find(parent, x);
59              int y_set_parent = find(parent, y);
60              parent[y_set_parent] = x_set_parent;
61          }
62          public void printGraph(ArrayList<Edge> edgeList){
63              for (int i = 0; i <edgeList.size() ; i++) {
64                  Edge edge = edgeList.get(i);
65                  System.out.println("Edge-" + i + " src: " + edge.src + " destination: " +
66                  edge.destination +" weight: " + edge.weight);
67              }
68          }
69      }
70          public static void main(String[] args) {
71              int vertices = 6;
72              Graph1 graph = new Graph1(vertices);
73              graph.addEdge(0, 1, 2);
74              graph.addEdge(1, 2, 3);
75              graph.addEdge(3, 5, 4);
76              graph.addEdge(5, 4, 2);
77              graph.addEdge(1, 3, 4);
78              graph.addEdge(3, 5, 2);
79              graph.addEdge(4, 5, 6);
80              graph.kruskalMST();
81
82          }
83      }
```

OUTPUT :

```
Minimum Spanning Tree(MST):
Edge-0 src: 0 destination: 1 weight: 2
Edge-1 src: 5 destination: 4 weight: 2
Edge-2 src: 3 destination: 5 weight: 2
Edge-3 src: 1 destination: 2 weight: 3
Edge-4 src: 1 destination: 3 weight: 4
```

2. USING JAVA IMPLEMENT THE PRIMS ALGORITHMS

CODE :

```
package Lab1b;
import java.util.*;
public class Prims {
        public void Prim(int G[][], int V) {
          int INF = 9999999;
          int no_edge;
          boolean[] selected = new boolean[V];
          Arrays.fill(selected, false);
          no_edge = 0;
          selected[0] = true;
          System.out.println("Edge : Weight");
          while (no_edge < V - 1) {
            int min = INF;
            int x = 0;
            int y = 0;

            for (int i = 0; i < V; i++) {
              if (selected[i] == true) {
                for (int j = 0; j < V; j++) {
                  if (!selected[j] && G[i][j] != 0) {
                    if (min > G[i][j]) {
                      min = G[i][j];
                      x = i;
                      y = j;
                    }
                  }
                }
              }
            }
            System.out.println(x + " - " + y + " :  " + G[x][y]);
            selected[y] = true;
            no_edge++;
          }
        }
        public static void main(String[] args) {
          Prims g = new Prims();
          int V = 6;
int[][] G = { { 0, 5, 85, 0, 0, 10}, { 8, 0, 45, 18, 24, 22 }, { 50, 65, 0, 55, 84,
62 },
                    { 0, 18, 56, 0, 33, 25 },{ 0, 46, 76, 42, 0, 66 },{3, 60, 86, 44,
34, 89}};
          g.Prim(G, V);
        }
}
```

```java
package Lab1b;
import java.util.*;
public class Prims {
      public void Prim(int G[][], int V) {
        int INF = 9999999;
        int no_edge;
        boolean[] selected = new boolean[V];
        Arrays.fill(selected, false);
        no_edge = 0;
        selected[0] = true;
        System.out.println("Edge : Weight");
        while (no_edge < V - 1) {
          int min = INF;
          int x = 0;
          int y = 0;

          for (int i = 0; i < V; i++) {
            if (selected[i] == true) {
              for (int j = 0; j < V; j++) {
                if (!selected[j] && G[i][j] != 0) {
                  if (min > G[i][j]) {
                    min = G[i][j];
                    x = i;
                    y = j;
                  }
                }
              }
            }
          }
          System.out.println(x + " - " + y + " :  " + G[x][y]);
          selected[y] = true;
          no_edge++;
        }
      }
      public static void main(String[] args) {
        Prims g = new Prims();
        int V = 6;
int[][] G = { { 0, 5, 85, 0, 0, 10}, { 8, 0, 45, 18, 24, 22 }, { 50, 65, 0, 55, 84, 62 },
                { 0, 18, 56, 0, 33, 25 },{ 0, 46, 76, 42, 0, 66 },{3, 60, 86, 44, 34, 89}};
        g.Prim(G, V);
      }
    }
```

OUTPUT :

```
<terminated> Prims [Java Applicati
Edge : Weight
0 - 1 :  5
0 - 5 :  10
1 - 3 :  18
1 - 4 :  24
1 - 2 :  45
```