6. Implement Greedy search algorithm for Kruskal's Minimal Spanning Tree Algorithm

```
import java.io.*;
import java.lang.*;
import java.util.*;

class KruskalsMST11 {
	class Edge implements Comparable<Edge> {
		int src, dest, weight;
		public int compareTo(Edge compareEdge) {
			return this.weight - compareEdge.weight;
		}
	};

	class subset {
		int parent, rank;
	};

	private int V, E;
	private Edge edge[];

	KruskalsMST11(int v, int e, int graph[][]) {
		this.V = v;
		this.E = e;
		this.edge = new Edge[E];
	  int i = -1;
		for (int x = 0; x < v; x++)
		for (int y = x; y < v; y++)
		    if (graph[x][y] != 0) {
			edge[++i] = new Edge();
			edge[i].src = x;
			edge[i].dest = y;
			edge[i].weight = graph[x][y];
		    }
	}

	int find(subset subsets[], int i) {
		if (subsets[i].parent != i)
			subsets[i].parent = find(subsets, subsets[i].parent);
		return subsets[i].parent;
	}

	void Union(subset subsets[], int x, int y) {
		int xroot = find(subsets, x);
		int yroot = find(subsets, y);

		if (subsets[xroot].rank < subsets[yroot].rank)
			subsets[xroot].parent = yroot;
		else if (subsets[xroot].rank > subsets[yroot].rank)
			subsets[yroot].parent = xroot;
		else {
			subsets[yroot].parent = xroot;
			subsets[xroot].rank++;
		}
	}

	void KruskalMSTf() {
		Edge result[] = new Edge[V];
		int e = 0, i = 0;
```

```
            for (i = 0; i < V; ++i)
                  result[i] = new Edge();
            Arrays.sort(edge);

            subset subsets[] = new subset[V];
            for (i = 0; i < V; ++i)
                  subsets[i] = new subset();

            for (int v = 0; v < V; ++v) {
                  subsets[v].parent = v;
                  subsets[v].rank = 0;
            }

            i = 0;

            while (e < V - 1) {
                  Edge next_edge = edge[i++];

                  int x = find(subsets, next_edge.src);
                  int y = find(subsets, next_edge.dest);

                  if (x != y) {
                        result[e++] = next_edge;
                        Union(subsets, x, y);
                  }
            }

            System.out.println("\n\n\nKruskal's Minimum Spanning
Tree:\nEdge \tWeight");
            int minimumCost = 0;
            for (i = 0; i < e; ++i) {
                  System.out.printf("%d -- %d == %d\n", result[i].src,
result[i].dest, result[i].weight);
                  minimumCost += result[i].weight;
            }
        System.out.printf("Minimum Cost: %d", minimumCost);
      }

      public static void main(String ar[])
      {

            // A2.jpg
                        int graph[][] = new int[][] {
                        { 0, 4, 0, 0, 0, 0, 0, 8, 0},
                        { 4, 0, 8, 0, 0, 0, 0,11, 0},
                        { 0, 8, 0, 7, 0, 4, 0, 0, 2},
                        { 0, 0, 7, 0, 9,14, 0, 0, 0},
                        { 0, 0, 0, 9, 0,10, 0, 0, 0},
                        { 0, 0, 4,14,10, 0, 2, 0, 0},
                        { 0, 0, 0, 0, 0, 2, 0, 1, 6},
                        { 8,11, 0, 0, 0, 0, 1, 0, 7},
                        { 0, 0, 2, 0, 0, 0, 6, 7, 0}
                    };


                        KruskalsMST11 kruskalsMST = new KruskalsMST11(9,
14, graph);
                        kruskalsMST.KruskalMSTf();
      }
}
```