

MST_Kruskal_Alg.java

```
1
2/* * MST Minimum Spanning Tree (Basic Form)
7import java.util.*;
8
9class MST_Kruskal_Alg{
10    static Node[] G;
11    static int sM;
12    static int N;
13    static int cnt = 0;
14    static int conCompCnt=0;
15    static Stack<Integer> s = new Stack<Integer>();
16    static List<Edge> E = new LinkedList<Edge>();
17    static List<Edge> MST = new LinkedList<Edge>();
18
19    static class DisjointSet{
20        int[] pset;
21        public DisjointSet(){
22            pset = new int[G.length];
23            for(int i = 0; i<pset.length; i++){
24                pset[i] = i;
25            }
26        }
27        public int findSet(int i){
28            return pset[i]==i ? pset[i] : (pset[i] =
findSet(pset[i]));
29        }
30        public void unionSet(int i, int j){
31            pset[findSet(i)] = findSet(j);
32        }
33        public boolean isSameSet(int i, int j){
34            return (findSet(i)==findSet(j));
35        }
36    }
37    static class Node {
38        List<Edge> adj;
39        int n;
40        public boolean visited;
41        int layer;
42        public Node(int N){
43            adj = new ArrayList<Edge>();
```

MST_Kruskal_Alg.java

```

44         n=N;
45         layer = -1;
46         visited = false;
47     }
48 }
49 static class Edge implements Comparable<Edge> {
50     int to, weight, from;
51     public Edge(int t, int w) {
52         to=t;
53         weight = w;
54     }
55     public Edge(int f, int t, int w) {
56         to=t;
57         weight = w;
58         from = f;
59     }
60     @Override
61     public int compareTo(Edge e) {
62         return this.weight - e.weight;
63     }
64 }
65 public static void makeGraph(int n) {
66     G = new Node[n];
67     for(int i =0; i<n; i++){
68         G[i]=new Node(i);
69     }
70 }
71 public static void addEdge(int u,int v, int w) {
72     G[u].adj.add(new Edge(u,v,w));
73     E.add(new Edge(u,v,w));
74     G[v].adj.add(new Edge(v,u,w));
75 }
76 public static int charN(char c) {
77     return c;
78 }
79 public static int MSTkruskal(DisjointSet ds) {
80     if(conComp(G) !=1) {
81         return -1;
82     }
83     Collections.sort(E);

```

MST_Kruskal_Alg.java

```

84         int cost = 0;
85         for(int i =0; i<E.size(); i++){
86             Edge e = E.get(i);
87             if(!(ds.isSameSet(e.from, e.to))){
88                 ds.unionSet(e.from, e.to);
89                 MST.add(e);
90                 cost+=e.weight;
91             }
92         }
93
94         return cost;
95     }
96     public static void dfs(int n) {
97         if(G[n].visited) {
98             return;
99         }
100         G[n].visited = true;
101         s.push(n);
102         cnt++;
103         for(Edge e : G[n].adj)
104         {
105             dfs(e.to);
106         }
107     }
108     public static int conComp(Node[] g){
109         for(int i = 0; i<g.length;i++){
110             if(!(g[i].visited)){
111                 conCompCnt++;
112                 dfs(i);
113             }
114         }
115         return conCompCnt;
116     }
117
118     public static void main(String[] args){
119         Scanner scan = new Scanner(System.in);
120         int K = Integer.parseInt(scan.nextLine());
121         int u = -1;
122         int v = -1;
123         int w = -1;

```

MST_Kruskal_Alg.java

```
124
125     for(int k =0; k<K; k++){
126         N = scan.nextInt();
127         makeGraph(N);
128         while((u = scan.nextInt()) !=-1){
129
130             v = scan.nextInt();
131             w = scan.nextInt();
132             addEdge(u,v,w);
133
134         }
135         DisjointSet ds = new DisjointSet();
136         System.out.println(MSTkruskal(ds));
137
138     }
139     scan.close();
140 }
141 }
142
```