1 /\*\*  
 2 \* @author Amar Bessedik Designs the Kruskal's algorithm for finding minimum  
 3 \* spanning trees of graphs.  
 4 \*/  
 5 public class Kruskal  
 6 {  
 7 private MinimumSpanningTree mst;// MST object, to potentially hold (V - 1) edges.  
 8 private DisjointSet ds;//To hold edges each in a disjoint set.  
 9 private HeapSort hs;//Needed to sort the edges of a graph.  
10 private int N;//Number of vertices  
11 private int u, v, wt, u\_set, v\_set;//Edge params: vertex1, vertex2, weight & sets.  
12   
13 //Constructor   
14 public Kruskal(Graph G)  
15 {  
16 this.N = G.getVertices();// # vertices  
17 this.hs = new HeapSort(); // heapSort instance  
18 this.ds = new DisjointSet(N);//Disjoint set of capacity N.  
19 this.mst = new MinimumSpanningTree(G);// MST instance.  
20 }//end constructor  
21   
22 /\*\*  
23 \* Kruskal's function.  
24 \* @param V # number of vertices.  
25 \* @param E array of graph edges.  
26 \*/  
27 public void kruskal(int V, Edge[] E)  
28 {  
29 Edge e;//Shortest edge yet to be considered.  
30 int count = 0;//counter of the graph's edges.  
31 int n = E.length;//# of edges.  
32   
33 //Sort Edges in inceasinding order of weight.  
34 hs.heapSort(E, n);  
35   
36 while ((count < n) && !mst.satisfied())  
37 {  
38 e = E[count++];//Shortest edge yet to consider.  
39 get\_parameters(e);//Get the edge's parameters.  
40   
41 u\_set = ds.find2(u);//The label of vertex u.  
42 v\_set = ds.find2(v);//The label of vertex v.  
43   
44 //If adding the edge to the MST would create a cycle.  
45 if (u\_set == v\_set)  
46 continue;  
47   
48 //Otherwise - NO CYCLE  
49 ds.merge(u\_set, v\_set); // merge the sets into one disjoint set.  
50 mst.add(e);// Add edge to MST  
51 mst.update(wt);// update total weight.   
52   
53 }//end while  
54 //Show results according to weither there is an MST or not.   
55 mst.output();  
56 }//end kruskal  
57   
58 /\*\*  
59 \* @param e gets vertices and weight of e.  
60 \*/  
61 private void get\_parameters(Edge e)  
62 {  
63 this.u = e.getVertex1();// get first vertex  
64 this.v = e.getVertex2();// get second vertex.  
65 this.wt = e.getWeight();// get edge's weight.  
66 }//end extract\_parameters  
67 }//end Kruskal's Class  
68