Graph.java

```
1      package com.hongchuan.app;
2      // Java program for Kruskal's algorithm to find Minimum
3      // Spanning Tree of a given connected, undirected and
4      // weighted graph
5      import java.util.*;
6      import java.lang.*;
7      import java.io.*;
8
9      public class Graph
10     {
11             int totalW = 0;
12             // A class to represent a graph edge
13             public class Edge implements Comparable<Edge>
14             {
15                     int src, dest, weight;
16
17                     // Comparator function used for sorting edges
18                     // based on their weight
19                     public int compareTo(Edge compareEdge)
20                     {
21  2                          return this.weight-compareEdge.weight;
22                     }
23             };
24
25             // A class to represent a subset for union-find
26             public class subset
27             {
28                     int parent, rank;
29             };
30
31             int V, E; // V-> no. of vertices & E->no.of edges
32             Edge edge[]; // collection of all edges
33
34             // Creates a graph with V vertices and E edges
35             public Graph(int v, int e)
36             {
37                     V = v;
38                     E = e;
39                     edge = new Edge[E];
40  3                  for (int i=0; i<e; ++i)
41                             edge[i] = new Edge();
42             }
43             public int getTotalW(){
44  1                  return totalW;
45             }
46             // A utility function to find set of an element i
47             // (uses path compression technique)
48             public int find(subset subsets[], int i)
49             {
50                     // find root and make root as parent of i (path compression)
51  1              if (subsets[i].parent != i)
52                             subsets[i].parent = find(subsets, subsets[i].parent);
53
54  1              return subsets[i].parent;
55             }
56
57             // A function that does union of two sets of x and y
58             // (uses union by rank)
59             public void Union(subset subsets[], int x, int y)
60             {
61                     int xroot = find(subsets, x);
62                     int yroot = find(subsets, y);
```

```
 63
 64                        // Attach smaller rank tree under root of high rank tree
 65                        // (Union by Rank)
 66 2                     if (subsets[xroot].rank < subsets[yroot].rank)
 67                                subsets[xroot].parent = yroot;
 68 2                     else if (subsets[xroot].rank > subsets[yroot].rank)
 69                                subsets[yroot].parent = xroot;
 70
 71                        // If ranks are same, then make one as root and increment
 72                        // its rank by one
 73                        else
 74                        {
 75                                subsets[yroot].parent = xroot;
 76 1                             subsets[xroot].rank++;
 77                        }
 78              }
 79
 80          // The main function to construct MST using Kruskal's algorithm
 81          public void KruskalMST()
 82          {
 83                  Edge result[] = new Edge[V]; // Tnis will store the resultant MST
 84                  int e = 0; // An index variable, used for result[]
 85                  int i = 0; // An index variable, used for sorted edges
 86 3               for (i=0; i<V; ++i)
 87                          result[i] = new Edge();
 88
 89                  // Step 1: Sort all the edges in non-decreasing order of their
 90                  // weight. If we are not allowed to change the given graph, we
 91                  // can create a copy of array of edges
 92 1               Arrays.sort(edge);
 93
 94                  // Allocate memory for creating V ssubsets
 95                  subset subsets[] = new subset[V];
 96 3               for(i=0; i<V; ++i)
 97                          subsets[i]=new subset();
 98
 99                  // Create V subsets with single elements
100 3               for (int v = 0; v < V; ++v)
101                  {
102                          subsets[v].parent = v;
103                          subsets[v].rank = 0;
104                  }
105
106                  i = 0; // Index used to pick next edge
107
108                  // Number of edges to be taken is equal to V-1
109 3               while (e < V - 1)
110                  {
111                          // Step 2: Pick the smallest edge. And increment
112                          // the index for next iteration
113                          Edge next_edge = new Edge();
114 1                       next_edge = edge[i++];
115
116                          int x = find(subsets, next_edge.src);
117                          int y = find(subsets, next_edge.dest);
118
119                          // If including this edge does't cause cycle,
120                          // include it in result and increment the index
121                          // of result for next edge
122 1                       if (x != y)
123                          {
124 1                               result[e++] = next_edge;
125 1                               Union(subsets, x, y);
126                          }
127                          // Else discard the next_edge
128                  }
```

```
129
130                    // print the contents of result[] to display
131                    // the built MST
132                    // System.out.println("Following are the edges in " +
133                    //                                      "the constructed MST");
134 3                 for (i = 0; i < e; ++i)
135 1                     System.out.println(result[i].src+" -- " +
136                             result[i].dest+" == " + result[i].weight);
137 1                     totalW += result[i].weight;
138          }
139
140          // Driver Program
141          // public static void main (String[] args)
142          // {
143
144          //      /* Let us create following weighted graph
145          //                   10
146          //             0--------1
147          //             | \      |
148          //      6| 5\ |15
149          //             |       \ |
150          //             2--------3
151          //                   4         */
152          //      int V = 4; // Number of vertices in graph
153          //      int E = 5; // Number of edges in graph
154          //      Graph graph = new Graph(V, E);
155
156          //      // add edge 0-1
157          //      graph.edge[0].src = 0;
158          //      graph.edge[0].dest = 1;
159          //      graph.edge[0].weight = 10;
160
161          //      // add edge 0-2
162          //      graph.edge[1].src = 0;
163          //      graph.edge[1].dest = 2;
164          //      graph.edge[1].weight = 6;
165
166          //      // add edge 0-3
167          //      graph.edge[2].src = 0;
168          //      graph.edge[2].dest = 3;
169          //      graph.edge[2].weight = 5;
170
171          //      // add edge 1-3
172          //      graph.edge[3].src = 1;
173          //      graph.edge[3].dest = 3;
174          //      graph.edge[3].weight = 15;
175
176          //      // add edge 2-3
177          //      graph.edge[4].src = 2;
178          //      graph.edge[4].dest = 3;
179          //      graph.edge[4].weight = 4;
180
181          //      graph.KruskalMST();
182          // }
183  }
184  //This code is contributed by Aakash Hasija
```

Mutations

21   1. Replaced integer subtraction with addition → SURVIVED
     2. replaced return of integer sized value with (x == 0 ? 1 : 0) → SURVIVED
     1. changed conditional boundary → KILLED
40   2. Changed increment from 1 to -1 → KILLED
     3. negated conditional → KILLED
44   1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
51   1. negated conditional → KILLED
54   1. replaced return of integer sized value with (x == 0 ? 1 : 0) → SURVIVED

```
 66   1. changed conditional boundary → SURVIVED
      2. negated conditional → SURVIVED
 68   1. changed conditional boundary → SURVIVED
      2. negated conditional → SURVIVED
 76   1. Replaced integer addition with subtraction → SURVIVED
      1. changed conditional boundary → KILLED
 86   2. Changed increment from 1 to -1 → KILLED
      3. negated conditional → KILLED
 92   1. removed call to java/util/Arrays::sort → SURVIVED
      1. changed conditional boundary → KILLED
 96   2. Changed increment from 1 to -1 → KILLED
      3. negated conditional → KILLED
      1. changed conditional boundary → KILLED
100   2. Changed increment from 1 to -1 → KILLED
      3. negated conditional → KILLED
      1. changed conditional boundary → KILLED
109   2. Replaced integer subtraction with addition → KILLED
      3. negated conditional → SURVIVED
114   1. Changed increment from 1 to -1 → KILLED
122   1. negated conditional → KILLED
124   1. Changed increment from 1 to -1 → KILLED
125   1. removed call to com/hongchuan/app/Graph::Union → SURVIVED
      1. changed conditional boundary → KILLED
134   2. Changed increment from 1 to -1 → KILLED
      3. negated conditional → SURVIVED
135   1. removed call to java/io/PrintStream::println → SURVIVED
137   1. Replaced integer addition with subtraction → SURVIVED
```

Active mutators

- INCREMENTS_MUTATOR
- VOID_METHOD_CALL_MUTATOR
- RETURN_VALS_MUTATOR
- MATH_MUTATOR
- NEGATE_CONDITIONALS_MUTATOR
- INVERT_NEGS_MUTATOR
- CONDITIONALS_BOUNDARY_MUTATOR

Tests examined

- com.hongchuan.app.GraphTest.testCase2(com.hongchuan.app.GraphTest) (1 ms)
- com.hongchuan.app.GraphTest.testCase1(com.hongchuan.app.GraphTest) (7 ms)

Report generated by [PIT](#) 1.4.3