Kruskal.java

```
1   package com.example.graph;
2
3   import java.util.PriorityQueue;
4
5   public class Kruskal {
6       public int minCostConnectPoints(int[][] points) {
7   2       if (points == null || points.length == 0)
8               return 0;
9
10  2       PriorityQueue<int[]> heap = new PriorityQueue<>((a, b) -> a[0] - b[0]);
11  3       for (int i = 0; i < points.length - 1; i++) {
12  3           for (int j = i + 1; j < points.length; j++) {
13                  heap.add(new int[] { distance(points, i, j), i, j });
14              }
15          }
16          int minCost = 0;
17          UnionFind uf = new UnionFind(points.length);
18  1       while (!heap.isEmpty()) {
19              int[] edge = heap.poll();
20  1           if (uf.union(edge[1], edge[2])) {
21  1               minCost += edge[0];
22              }
23          }
24
25  1       return minCost;
26      }
27
28      public int distance(int[][] points, int a, int b) {
29  4       return Math.abs(points[a][0] - points[b][0]) + Math.abs(points[a][1] - points[b][1]);
30      }
31  }
32
33  class UnionFind {
34      int[] parent;
35      int[] rank;
36
37      public UnionFind(int n) {
38          this.rank = new int[n];
39          this.parent = new int[n];
40
41  2       for (int i = 0; i < n; i++)
42              parent[i] = i;
43      }
44
45      public int find(int x) {
46  1       if (parent[x] != x) {
47              parent[x] = find(parent[x]);
48          }
49  1       return parent[x];
50      }
51
52      // return true if we do not get a cycle
53      public boolean union(int x, int y) {
54          int parentX = find(x);
55          int parentY = find(y);
56  1       if (parentX == parentY)
57  1           return false;
58  1       if (rank[parentX] == rank[parentY]) {
59              parent[parentX] = parentY;
60  1           rank[parentY] += 1;
61  2       } else if (rank[parentX] < rank[parentY]) {
62              parent[parentX] = parentY;
63          } else {
64              parent[parentY] = parentX;
65          }
66  1       return true;
```

```
67      }
68  }
```

Mutations

7	1. negated conditional → KILLED 2. negated conditional → KILLED
10	1. Replaced integer subtraction with addition → KILLED 2. replaced int return with 0 for com/example/graph/Kruskal::lambda$minCostConnectPoints$0 → KILLED
11	1. Replaced integer subtraction with addition → SURVIVED 2. negated conditional → KILLED 3. changed conditional boundary → SURVIVED
12	1. changed conditional boundary → KILLED 2. Replaced integer addition with subtraction → KILLED 3. negated conditional → KILLED
18	1. negated conditional → KILLED
20	1. negated conditional → KILLED
21	1. Replaced integer addition with subtraction → KILLED
25	1. replaced int return with 0 for com/example/graph/Kruskal::minCostConnectPoints → KILLED
29	1. replaced int return with 0 for com/example/graph/Kruskal::distance → KILLED 2. Replaced integer subtraction with addition → KILLED 3. Replaced integer addition with subtraction → KILLED 4. Replaced integer subtraction with addition → KILLED
41	1. negated conditional → KILLED 2. changed conditional boundary → KILLED
46	1. negated conditional → KILLED
49	1. replaced int return with 0 for com/example/graph/UnionFind::find → KILLED
56	1. negated conditional → KILLED
57	1. replaced boolean return with true for com/example/graph/UnionFind::union → KILLED
58	1. negated conditional → SURVIVED
60	1. Replaced integer addition with subtraction → SURVIVED
61	1. changed conditional boundary → SURVIVED 2. negated conditional → SURVIVED
66	1. replaced boolean return with false for com/example/graph/UnionFind::union → KILLED

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NEGATE_CONDITIONALS
- NULL_RETURNS
- PRIMITIVE_RETURNS
- TRUE_RETURNS
- VOID_METHOD_CALLS

Tests examined

- com.example.graph.KruskalTest.testMinCostConnectPoints(com.example.graph.KruskalTest) (0 ms)

Report generated by PIT 1.15.0