CriticalConnections.java

```
1   package com.example.graph;
2
3   import java.util.*;
4
5   public class CriticalConnections {
6       public List<List<Integer>> criticalConnections(int n, List<List<Integer>> connections) {
7           List<Set<Integer>> adj = new ArrayList<>();
8           int[] parent = new int[n];
9   2       for (int i = 0; i < n; i++) {
10              adj.add(new HashSet<Integer>()); // We use hashset so that removal of edge is quick
11              parent[i] = -1;
12          }
13          for (List<Integer> edge : connections) {
14              adj.get(edge.get(0)).add(edge.get(1));
15              adj.get(edge.get(1)).add(edge.get(0));
16          }
17
18          Stack<Integer> stack = new Stack<>();
19          boolean[] visited = new boolean[n];
20  2       for (int i = 0; i < n; i++) {
21  1           if (!visited[i]) {
22  1               getOrder(adj, stack, parent, visited, i); // Fill stack for ordering
23              }
24          }
25
26  2       for (int i = 0; i < n; i++) {
27  1           if (parent[i] != -1) {
28                  adj.get(parent[i]).remove(i); // This is similar to the case where we have to build the tr
29                                                // as per Kosaraju's Algo
30              }
31          }
32  1       Arrays.fill(visited, false);
33
34          List<List<Integer>> criticals = new ArrayList<>();
35  1       while (!stack.isEmpty()) {
36              int v = stack.pop();
37  1           if (!visited[v]) {
38  1               if (parent[v] != -1) {
39                      criticals.add(Arrays.asList(parent[v], v)); // Whenever we pop from stack and the comp
40                                                                  // unvisited it means a new SCC
41                  }
42  1               dfs(adj, visited, v);
43              }
44          }
45
46  1       return criticals;
47      }
48
49      private void getOrder(List<Set<Integer>> adj, Stack<Integer> stack, int[] parent, boolean[] visited, i
50          visited[s] = true;
51
52          for (int n : adj.get(s)) {
53  1           if (!visited[n]) {
54                  parent[n] = s;
55  1               getOrder(adj, stack, parent, visited, n);
56              }
57          }
58
59          stack.push(s);
60      }
61
62      private void dfs(List<Set<Integer>> adj, boolean[] visited, int s) {
63          visited[s] = true;
64
65          for (int n : adj.get(s)) {
66  1           if (!visited[n]) {
67  1               dfs(adj, visited, n);
68              }
69          }
70
71      }
72  }
```

Mutations

9	1. changed conditional boundary → KILLED 2. negated conditional → KILLED
20	1. changed conditional boundary → KILLED 2. negated conditional → KILLED
21	1. negated conditional → KILLED
22	1. removed call to com/example/graph/CriticalConnections::getOrder → KILLED
26	1. changed conditional boundary → KILLED 2. negated conditional → KILLED
27	1. negated conditional → KILLED
32	1. removed call to java/util/Arrays::fill → KILLED
35	1. negated conditional → KILLED
37	1. negated conditional → KILLED
38	1. negated conditional → KILLED
42	1. removed call to com/example/graph/CriticalConnections::dfs → KILLED
46	1. replaced return value with Collections.emptyList for com/example/graph/CriticalConnections::criticalCo KILLED
53	1. negated conditional → KILLED
55	1. removed call to com/example/graph/CriticalConnections::getOrder → KILLED
66	1. negated conditional → KILLED
67	1. removed call to com/example/graph/CriticalConnections::dfs → 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.CriticalConnectionsTest.testCriticalConnections(com.example.graph.CriticalConnectionsTest) (0 ms)

Report generated by PIT 1.15.0