# CriticalConnections.java

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

### Mutations

```
10 1. changed conditional boundary → KILLED
   2. Changed increment from 1 to -1 → KILLED
   3. negated conditional → KILLED
21 1. changed conditional boundary → KILLED
   2. Changed increment from 1 to -1 → KILLED
   3. negated conditional → KILLED
22 1. negated conditional → KILLED
23 1. removed call to com/example/CriticalConnections::getOrder → KILLED
27 1. changed conditional boundary → KILLED
   2. Changed increment from 1 to -1 → KILLED
   3. negated conditional → KILLED
28 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
41 1. removed call to com/example/CriticalConnections::dfs → KILLED
45 1. replaced return value with Collections.emptyList for com/example/CriticalConnections::criticalConnections → KILLED
52 1. negated conditional → KILLED
54 1. removed call to com/example/CriticalConnections::getOrder → KILLED
65 1. negated conditional → KILLED
66 1. removed call to com/example/CriticalConnections::dfs → KILLED
```

## Active mutators

- BOOLEAN_FALSE_RETURN
- BOOLEAN_TRUE_RETURN
- CONDITIONALS_BOUNDARY_MUTATOR
- EMPTY_RETURN_VALUES
- INCREMENTS_MUTATOR
- INVERT_NEGS_MUTATOR

- MATH_MUTATOR
- NEGATE_CONDITIONALS_MUTATOR
- NULL_RETURN_VALUES
- PRIMITIVE_RETURN_VALS_MUTATOR
- VOID_METHOD_CALL_MUTATOR

## Tests examined

- com.example.CriticalConnectionsTest.testCriticalConnections(com.example.CriticalConnectionsTest) (0 ms)

Report generated by PIT 1.5.0