AccountsMerge.java

```
1   package com.example.graph;
2
3   import java.util.*;
4
5   class AccountsMerge {
6       private class Node {
7           String parent;
8           String owner;
9           int rank;
10
11          public Node(String owner, String parent, int rank) {
12              this.owner = owner;
13              this.parent = parent;
14              this.rank = rank;
15          }
16      }
17
18      // Union by Rank and Path Compression
19      private class AccountMergeUnionFind {
20          private Map<String, Node> emailToNode;
21
22          public AccountMergeUnionFind() {
23              this.emailToNode = new HashMap<>();
24          }
25
26          public void addEmail(String owner, String email) {
27  1           if (!emailToNode.containsKey(email)) {
28                  emailToNode.put(email, new Node(owner, email, 0));
29              }
30          }
31
32          public String findParent(String email) {
33              Node curNode = emailToNode.get(email);
34  1           if (!email.equals(curNode.parent)) {
35                  curNode.parent = findParent(curNode.parent);
36              }
37  1           return curNode.parent;
38          }
39
40          public void union(String email1, String email2) {
41              String parent1 = findParent(email1);
42              String parent2 = findParent(email2);
43
44  1           if (parent1.equals(parent2)) {
45                  return;
46              }
47
48              Node node1 = emailToNode.get(parent1);
49              Node node2 = emailToNode.get(parent2);
50
51  2           if (node1.rank >= node2.rank) {
52                  node2.parent = parent1;
53  1               if (node1.rank == node2.rank) {
54  1                   node1.rank++;
55                  }
```

```
56                  } else {
57                      node1.parent = parent2;
58                  }
59              }
60
61              public String getOwner(String email) {
62  1               return emailToNode.get(email).owner;
63              }
64          }
65
66          public List<List<String>> accountsMerge(List<List<String>> accounts) {
67  1           if (accounts == null) {
68                  throw new IllegalArgumentException("Input is null");
69              }
70  1           if (accounts.size() == 0) {
71  1               return new ArrayList<>();
72              }
73
74              AccountMergeUnionFind uf = new AccountMergeUnionFind();
75
76              // Populating UnionFind with all emails.
77              for (List<String> account : accounts) {
78  2               if (account.size() <= 1) {
79                      continue;
80                  }
81                  String owner = account.get(0);
82  1               uf.addEmail(owner, account.get(1));
83
84  2               for (int i = 2; i < account.size(); i++) {
85  1                   uf.addEmail(owner, account.get(i));
86  1                   uf.union(account.get(1), account.get(i));
87                  }
88              }
89
90              // Creating merged Groups with sorted emails
91              Map<String, TreeSet<String>> groups = new HashMap<>();
92              for (List<String> account : accounts) {
93  2               if (account.size() <= 1) {
94                      continue;
95                  }
96                  String parent = uf.findParent(account.get(1));
97  1               if (!groups.containsKey(parent)) {
98                      groups.put(parent, new TreeSet<String>());
99                  }
100                 TreeSet<String> curGroup = groups.get(parent);
101 2               for (int i = 1; i < account.size(); i++) {
102                     curGroup.add(account.get(i));
103                 }
104             }
105
106             // Creating final result list
107             List<List<String>> result = new ArrayList<>();
108             for (Map.Entry<String, TreeSet<String>> group : groups.entrySet()) {
109                 List<String> account = new ArrayList<>();
110                 account.add(uf.getOwner(group.getKey()));
111                 account.addAll(group.getValue());
112                 result.add(account);
113             }
114
```

```
115 1         return result;
116       }
117   }
```

Mutations

27	1. negated conditional → KILLED
34	1. negated conditional → KILLED
37	1. replaced return value with "" for com/example/graph/AccountsMerge$AccountMergeUnionFind::findParent → KILLED
44	1. negated conditional → KILLED
51	1. changed conditional boundary → KILLED 2. negated conditional → KILLED
53	1. negated conditional → SURVIVED
54	1. Replaced integer addition with subtraction → SURVIVED
62	1. replaced return value with "" for com/example/graph/AccountsMerge$AccountMergeUnionFind::getOwner → KILLED
67	1. negated conditional → KILLED
70	1. negated conditional → KILLED
71	1. replaced return value with Collections.emptyList for com/example/graph/AccountsMerge::accountsMerge → SURVIVED
78	1. negated conditional → KILLED 2. changed conditional boundary → SURVIVED
82	1. removed call to com/example/graph/AccountsMerge$AccountMergeUnionFind::addEmail → KILLED
84	1. changed conditional boundary → KILLED 2. negated conditional → KILLED
85	1. removed call to com/example/graph/AccountsMerge$AccountMergeUnionFind::addEmail → KILLED
86	1. removed call to com/example/graph/AccountsMerge$AccountMergeUnionFind::union → KILLED
93	1. negated conditional → KILLED 2. changed conditional boundary → SURVIVED
97	1. negated conditional → KILLED
101	1. negated conditional → KILLED 2. changed conditional boundary → KILLED
115	1. replaced return value with Collections.emptyList for com/example/graph/AccountsMerge::accountsMerge → 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.AccountsMergeTest.testAccountsMerge(com.example.graph.AccountsMergeTest) (2 ms)

Report generated by PIT 1.15.0