

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 using System;
4 using System.Collections.Generic;
5
6 public class Program {
7
8     // O(a * (a + r) + a + r + alog(a)) time | O(a + r) space - where a is the number of airports and r is the number of routes
9     public static int AirportConnections(
10         List<string> airports,
11         List<List<string> > routes,
12         string startingAirport
13     ) {
14         Dictionary<string, AirportNode> airportGraph = createAirportGraph(airports, routes);
15         List<AirportNode> unreachableAirportNodes = getUnreachableAirportNodes(airportGraph,
16             airports,
17             startingAirport);
18         markUnreachableConnections(airportGraph, unreachableAirportNodes);
19         return getMinNumberOfNewConnections(airportGraph, unreachableAirportNodes);
20     }
21
22     // O(a + r) time | O(a + r) space
23     public static Dictionary<string, AirportNode> createAirportGraph(
24         List<string> airports,
25         List<List<string> > routes
26     ) {
27         Dictionary<string,
28             AirportNode> airportGraph = new Dictionary<string, AirportNode>();
29         foreach (string airport in airports) {
30             airportGraph.Add(airport, new AirportNode(airport));
31         }
32         foreach (List<string> route in routes) {
33             string airport = route[0];
34             string connection = route[1];
35             airportGraph[airport].connections.Add(connection);
36         }
37         return airportGraph;
38     }
39
40     // O(a + r) time | O(a) space
41     public static List<AirportNode> getUnreachableAirportNodes(
42         Dictionary<string, AirportNode> airportGraph,
43         List<string> airports,
44         string startingAirport
45     ) {
46         HashSet<string> visitedAirports = new HashSet<string>();
47         depthFirstTraverseAirports(airportGraph, startingAirport, visitedAirports);
48
49         List<AirportNode> unreachableAirportNodes = new List<AirportNode>();
50         foreach (string airport in airports) {
51             if (visitedAirports.Contains(airport)) continue;
52             AirportNode airportNode = airportGraph[airport];
53             airportNode.isReachable = false;
54             unreachableAirportNodes.Add(airportNode);
55         }
56         return unreachableAirportNodes;
57     }
58
59     public static void depthFirstTraverseAirports(
60         Dictionary<string, AirportNode> airportGraph,
61         string airport,
62         HashSet<string> visitedAirports
63     ) {
64         if (visitedAirports.Contains(airport)) return;
65         visitedAirports.Add(airport);
66         List<string> connections = airportGraph[airport].connections;
67         foreach (string connection in connections) {
68             depthFirstTraverseAirports(airportGraph, connection, visitedAirports);
69         }
70     }
71
72     // O(a * (a + r)) time | O(a) space
73     public static void markUnreachableConnections(
74         Dictionary<string, AirportNode> airportGraph,
75         List<AirportNode> unreachableAirportNodes
76     ) {
77         foreach (AirportNode airportNode in unreachableAirportNodes) {
78             string airport = airportNode.airport;
79             List<string> unreachableConnections = new List<string>();
80             HashSet<string> visitedAirports = new HashSet<string>();
81             depthFirstAddUnreachableConnections(airportGraph, airport,
82                 unreachableConnections,
83                 visitedAirports);
84             airportNode.unreachableConnections = unreachableConnections;
85         }
86     }
87
88     public static void depthFirstAddUnreachableConnections(
89         Dictionary<string, AirportNode> airportGraph,
90         string airport,
91         List<string> unreachableConnections,
92         HashSet<string> visitedAirports
93     ) {
94         if (airportGraph[airport].isReachable) return;
95         if (visitedAirports.Contains(airport)) return;
96         visitedAirports.Add(airport);
97         unreachableConnections.Add(airport);
98         List<string> connections = airportGraph[airport].connections;
99         foreach (string connection in connections) {
100             depthFirstAddUnreachableConnections(airportGraph, connection,
101                 unreachableConnections,
102                 visitedAirports);
103         }
104     }
105
106     // O(alog(a) + a + r) time | O(1) space
107     public static int getMinNumberOfNewConnections(
108         Dictionary<string, AirportNode> airportGraph,
109         List<AirportNode> unreachableAirportNodes
110     ) {
111         unreachableAirportNodes.Sort((a1,
112             a2) => a2.unreachableConnections.Count -
113             a1.unreachableConnections.Count);
114         int numberOfNewConnections = 0;
115         foreach (AirportNode airportNode in unreachableAirportNodes) {
```

```
116         if (airportNode.isReachable) continue;
117         numberOfNewConnections++;
118         foreach (string connection in airportNode.unreachableConnections) {
119             airportGraph[connection].isReachable = true;
120         }
121     }
122     return numberOfNewConnections;
123 }
124
125 public class AirportNode {
126     public string airport;
127     public List<string> connections;
128     public bool isReachable;
129     public List<string> unreachableConnections;
130
131     public AirportNode(string airport) {
132         this.airport = airport;
133         connections = new List<string>();
134         isReachable = true;
135         unreachableConnections = new List<string>();
136     }
137 }
138 }
139
```