

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 package main
4
5 import "sort"
6
7 type AirportNode struct {
8     Airport      string
9     Connections  []string
10    IsReachable   bool
11    UnreachableConnections []string
12 }
13
14 func NewAirportNode(airport string) *AirportNode {
15     return &AirportNode{
16         Airport:      airport,
17         Connections:  []string{},
18         IsReachable:   true,
19         UnreachableConnections: []string{},
20     }
21 }
22
23 // 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
24 func AirportConnections(airports []string, routes [][]string, startingAirport string) int {
25     airportGraph := createAirportGraph(airports, routes)
26     unreachableAirportNodes := getUnreachableAirportNodes(airportGraph, airports, startingAirport)
27     markUnreachableConnections(airportGraph, unreachableAirportNodes)
28     return getMinNumberOfNewConnections(airportGraph, unreachableAirportNodes)
29 }
30
31 // O(a + r) time | O(a + r) space
32 func createAirportGraph(airports []string, routes [][]string) map[string]*AirportNode {
33     airportGraph := map[string]*AirportNode{}
34     for _, airport := range airports {
35         airportGraph[airport] = NewAirportNode(airport)
36     }
37     for _, route := range routes {
38         airport, connection := route[0], route[1]
39         airportGraph[airport].Connections = append(airportGraph[airport].Connections, connection)
40     }
41     return airportGraph
42 }
43
44 // O(a + r) time | O(a) space
45 func getUnreachableAirportNodes(
46     airportGraph map[string]*AirportNode, airports []string, startingAirport string,
47 ) []*AirportNode {
48     visitedAirports := map[string]bool{}
49     depthFirstTraverseAirports(airportGraph, startingAirport, visitedAirports)
50
51     unreachableAirportNodes := []*AirportNode{}
52     for _, airport := range airports {
53         if _, found := visitedAirports[airport]; found {
54             continue
55         }
56         airportNode := airportGraph[airport]
57         airportNode.IsReachable = false
58         unreachableAirportNodes = append(unreachableAirportNodes, airportNode)
59     }
60     return unreachableAirportNodes
61 }
62
63 func depthFirstTraverseAirports(
64     airportGraph map[string]*AirportNode, airport string, visitedAirports map[string]bool,
65 ) {
66     if _, found := visitedAirports[airport]; found {
67         return
68     }
69     visitedAirports[airport] = true
70     connections := airportGraph[airport].Connections
71     for _, connection := range connections {
72         depthFirstTraverseAirports(airportGraph, connection, visitedAirports)
73     }
74 }
75
76 // O(a * (a + r)) time | O(a) space
77 func markUnreachableConnections(
78     airportGraph map[string]*AirportNode, unreachableAirportNodes []*AirportNode,
79 ) {
80     for _, airportNode := range unreachableAirportNodes {
81         airport := airportNode.Airport
82         unreachableConnections := []string{}
83         visitedAirports := map[string]bool{}
84         depthFirstAddUnreachableConnections(airportGraph, airport, &unreachableConnections, visitedAirports)
85         airportNode.UnreachableConnections = unreachableConnections
86     }
87     return
88 }
89
90 func depthFirstAddUnreachableConnections(
91     airportGraph map[string]*AirportNode, airport string,
92     unreachableConnections *[]string, visitedAirports map[string]bool,
93 ) {
94     if airportGraph[airport].IsReachable {
95         return
96     } else if _, found := visitedAirports[airport]; found {
97         return
98     }
99     visitedAirports[airport] = true
100    *unreachableConnections = append(*unreachableConnections, airport)
101    connections := airportGraph[airport].Connections
102    for _, connection := range connections {
103        depthFirstAddUnreachableConnections(airportGraph, connection, unreachableConnections, visitedAirports)
104    }
105 }
106
107 // O(alog(a) + a + r) time | O(1) space
108 func getMinNumberOfNewConnections(
109     airportGraph map[string]*AirportNode, unreachableAirportNodes []*AirportNode,
110 ) int {
111     sort.SliceStable(unreachableAirportNodes, func(i, j int) bool {
112         a1, a2 := unreachableAirportNodes[i], unreachableAirportNodes[j]
113         return len(a1.UnreachableConnections) > len(a2.UnreachableConnections)
114     })
115     numberOfNewConnections := 0
```

```
116     for _, node := range unreachableAirportNodes {
117         if node.IsReachable {
118             continue
119         }
120         numberOfNewConnections++
121         for _, connection := range node.UnreachableConnections {
122             airportGraph[connection].IsReachable = true
123         }
124     }
125     return numberOfNewConnections
126 }
127
```