AlgoExpert Quad Layout JavaScript 12px Sublime Monokai 00:00:00

Prompt Scratchpad Our Solution(s) Video Explanation Run Code

```
// O(a * (a + r) + a + r + alog(a)) time \mid O(a + r) space - where a is the number of airports and r is the number of routes
            function \ airportConnections (airports, \ routes, \ startingAirport) \ \{
                 const airportGraph = createAirportGraph(airports, routes);
                 const unreachableAirportNodes = getUnreachableAirportNodes(airportGraph, airports, startingAirport);
                 markUnreachableConnections(airportGraph, unreachableAirportNodes);
                 return getMinNumberOfNewConnections(airportGraph, unreachableAirportNodes);
  9 }
 10
            // O(a + r) time | O(a + r) space
            function createAirportGraph(airports, routes) {
 13
                const airportGraph = {};
 14
                 for (const airport of airports) {
                       airportGraph[airport] = new AirportNode(airport);
15
 16
 17
                  for (const route of routes) \{
 18
                      const [airport, connection] = route;
 19
                       airportGraph[airport].connections.push(connection);
20
                 return airportGraph;
22
23
 24
            // O(a + r) time | O(a) space
             function \ getUnreachable AirportNodes (airportGraph, airports, startingAirport) \ \{ constant of the constan
 26
                 const visitedAirports = {};
27
                 depthFirstTraverseAirports(airportGraph, startingAirport, visitedAirports);
28
29
                  const unreachableAirportNodes = [];
 30
                  for (const airport of airports) {
 31
                       if (airport in visitedAirports) continue;
 32
                       const airportNode = airportGraph[airport];
                       airportNode.isReachable = false;
 33
 34
                       unreachableAirportNodes.push(airportNode);
 35
                 return unreachableAirportNodes;
 36
 37 }
 38
             function \ depth First Traverse Airports (airport Graph, \ airport, \ visited Airports) \ \{
 39
                 if (airport in visitedAirports) return;
 40
                 visitedAirports[airport] = true;
41
                 const {connections} = airportGraph[airport];
42
43
                  for (const connection of connections) \{
                       depthFirstTraverseAirports(airportGraph, connection, visitedAirports);
44
 45
 46 }
47
48 // 0(a * (a + r)) time | 0(a) space
49
           function markUnreachableConnections(airportGraph, unreachableAirportNodes) {
50
                 for (const airportNode of unreachableAirportNodes) {
 51
                       const {airport} = airportNode;
 52
                       const unreachableConnections = [];
 53
                       \tt depthFirstAddUnreachableConnections(airportGraph, airport, unreachableConnections, \{\}); \\
 54
                       airportNode.unreachableConnections = unreachableConnections;
55
56 }
57
 58 \quad function \ depth First Add Unreachable Connections (airport Graph, \ airport, \ unreachable Connections, \ visited Airports) \ \{ (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1), (1,1),
 59
                 if (airportGraph[airport].isReachable) return;
                 if (airport in visitedAirports) return;
60
61
                 visitedAirports[airport] = true;
                 unreachableConnections.push(airport);
62
63
                 const {connections} = airportGraph[airport];
64
                  for (const connection of connections) {
65
                       \tt depthFirstAddUnreachableConnections (airportGraph, connection, unreachableConnections, visitedAirports); \\
 66
67 }
68
           // O(alog(a) + a + r) time | O(1) space
69
 70
             function \ get \verb|MinNumberOfNewConnections| (airportGraph, \ unreachable AirportNodes)| \{ (airportGraph, \ unreachable AirportNodes) \} \} (airportGraph, \ unreachable AirportNodes)| \} (airportGraph)| \} (airportGraph)| \} (airportGraph)| \} (airportGraph
                 unreachableAirportNodes.sort((a1, a2) => a2.unreachableConnections.length - a1.unreachableConnections.length);
 72
 73
                  let numberOfNewConnections = 0;
 74
                  for (const airportNode of unreachableAirportNodes) {
 75
                       if (airportNode.isReachable) continue;
 76
                       numberOfNewConnections++;
 77
                       for (const connection of airportNode.unreachableConnections) {
 78
                            airportGraph[connection].isReachable = true;
 79
 80
81
                 return numberOfNewConnections;
82 }
83
            class AirportNode {
                 constructor(airport) {
85
86
                       this.airport = airport;
 87
                       this.connections = [];
 88
                       this.isReachable = true;
                       this.unreachableConnections = [];
 89
90
91 }
92
           exports.airportConnections = airportConnections;
```

Solution 1