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

Prompt   Scratchpad   Our Solution(s)   Video Explanation   Run Code

Solution 1

```python
# Copyright © 2020 AlgoExpert, LLC. All rights reserved.

# 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
def airportConnections(airports, routes, startingAirport):
    airportGraph = createAirportGraph(airports, routes)
    unreachableAirportNodes = getUnreachableAirportNodes(airportGraph, airports, startingAirport)
    markUnreachableConnections(airportGraph, unreachableAirportNodes)
    return getMinNumberOfNewConnections(airportGraph, unreachableAirportNodes)


# O(a + r) time | O(a + r) space
def createAirportGraph(airports, routes):
    airportGraph = {}
    for airport in airports:
        airportGraph[airport] = AirportNode(airport)
    for route in routes:
        airport, connection = route
        airportGraph[airport].connections.append(connection)
    return airportGraph


# O(a + r) time | O(a) space
def getUnreachableAirportNodes(airportGraph, airports, startingAirport):
    visitedAirports = {}
    depthFirstTraverseAirports(airportGraph, startingAirport, visitedAirports)

    unreachableAirportNodes = []
    for airport in airports:
        if airport in visitedAirports:
            continue
        airportNode = airportGraph[airport]
        airportNode.isReachable = False
        unreachableAirportNodes.append(airportNode)
    return unreachableAirportNodes


def depthFirstTraverseAirports(airportGraph, airport, visitedAirports):
    if airport in visitedAirports:
        return
    visitedAirports[airport] = True
    connections = airportGraph[airport].connections
    for connection in connections:
        depthFirstTraverseAirports(airportGraph, connection, visitedAirports)


# O(a * (a + r)) time | O(a) space
def markUnreachableConnections(airportGraph, unreachableAirportNodes):
    for airportNode in unreachableAirportNodes:
        airport = airportNode.airport
        unreachableConnections = []
        depthFirstAddUnreachableConnections(airportGraph, airport, unreachableConnections, {})
        airportNode.unreachableConnections = unreachableConnections


def depthFirstAddUnreachableConnections(airportGraph, airport, unreachableConnections, visitedAirports):
    if airportGraph[airport].isReachable:
        return
    if airport in visitedAirports:
        return
    visitedAirports[airport] = True
    unreachableConnections.append(airport)
    connections = airportGraph[airport].connections
    for connection in connections:
        depthFirstAddUnreachableConnections(airportGraph, connection, unreachableConnections, visitedAirports)


# O(alog(a) + a + r) time | O(1) space
def getMinNumberOfNewConnections(airportGraph, unreachableAirportNodes):
    unreachableAirportNodes.sort(key=lambda airport: len(airport.unreachableConnections), reverse=True)

    numberOfNewConnections = 0
    for airportNode in unreachableAirportNodes:
        if airportNode.isReachable:
            continue
        numberOfNewConnections += 1
        for connection in airportNode.unreachableConnections:
            airportGraph[connection].isReachable = True
    return numberOfNewConnections


class AirportNode:
    def __init__(self, airport):
        self.airport = airport
        self.connections = []
        self.isReachable = True
        self.unreachableConnections = []
```