Solution 1

```csharp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

using System;
using System.Collections.Generic;

public class Program {

  // 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
  public static int AirportConnections(
    List<string> airports,
    List<List<string> > routes,
    string startingAirport
    ) {
    Dictionary<string, AirportNode> airportGraph = createAirportGraph(airports, routes);
    List<AirportNode> unreachableAirportNodes = getUnreachableAirportNodes(airportGraph,
        airports,
        startingAirport);
    markUnreachableConnections(airportGraph, unreachableAirportNodes);
    return getMinNumberOfNewConnections(airportGraph, unreachableAirportNodes);
  }

  // O(a + r) time | O(a + r) space
  public static Dictionary<string, AirportNode> createAirportGraph(
    List<string> airports,
    List<List<string> > routes
    ) {
    Dictionary<string,
      AirportNode> airportGraph = new Dictionary<string, AirportNode>();
    foreach (string airport in airports) {
      airportGraph.Add(airport, new AirportNode(airport));
    }
    foreach (List<string> route in routes) {
      string airport = route[0];
      string connection = route[1];
      airportGraph[airport].connections.Add(connection);
    }
    return airportGraph;
  }

  // O(a + r) time | O(a) space
  public static List<AirportNode> getUnreachableAirportNodes(
    Dictionary<string, AirportNode> airportGraph,
    List<string> airports,
    string startingAirport
    ) {
    HashSet<string> visitedAirports = new HashSet<string>();
    depthFirstTraverseAirports(airportGraph, startingAirport, visitedAirports);

    List<AirportNode> unreachableAirportNodes = new List<AirportNode>();
    foreach (string airport in airports) {
      if (visitedAirports.Contains(airport)) continue;
      AirportNode airportNode = airportGraph[airport];
      airportNode.isReachable = false;
      unreachableAirportNodes.Add(airportNode);
    }
    return unreachableAirportNodes;
  }

  public static void depthFirstTraverseAirports(
    Dictionary<string, AirportNode> airportGraph,
    string airport,
    HashSet<string> visitedAirports
    ) {
    if (visitedAirports.Contains(airport)) return;
    visitedAirports.Add(airport);
    List<string> connections = airportGraph[airport].connections;
    foreach (string connection in connections) {
      depthFirstTraverseAirports(airportGraph, connection, visitedAirports);
    }
  }

  // O(a * (a + r)) time | O(a) space
  public static void markUnreachableConnections(
    Dictionary<string, AirportNode> airportGraph,
    List<AirportNode> unreachableAirportNodes
    ) {
    foreach (AirportNode airportNode in unreachableAirportNodes) {
      string airport = airportNode.airport;
      List<string> unreachableConnections = new List<string>();
      HashSet<string> visitedAirports = new HashSet<string>();
      depthFirstAddUnreachableConnections(airportGraph, airport,
        unreachableConnections,
        visitedAirports);
      airportNode.unreachableConnections = unreachableConnections;
    }
  }

  public static void depthFirstAddUnreachableConnections(
    Dictionary<string, AirportNode> airportGraph,
    string airport,
    List<string> unreachableConnections,
    HashSet<string> visitedAirports
    ) {
    if (airportGraph[airport].isReachable) return;
    if (visitedAirports.Contains(airport)) return;
    visitedAirports.Add(airport);
    unreachableConnections.Add(airport);
    List<string> connections = airportGraph[airport].connections;
    foreach (string connection in connections) {
      depthFirstAddUnreachableConnections(airportGraph, connection,
        unreachableConnections,
        visitedAirports);
    }
  }

  // O(alog(a) + a + r) time | O(1) space
  public static int getMinNumberOfNewConnections(
    Dictionary<string, AirportNode> airportGraph,
    List<AirportNode> unreachableAirportNodes
    ) {
    unreachableAirportNodes.Sort((a1,
      a2) => a2.unreachableConnections.Count -
      a1.unreachableConnections.Count);
    int numberOfNewConnections = 0;
    foreach (AirportNode airportNode in unreachableAirportNodes) {
      if (airportNode.isReachable) continue;
      numberOfNewConnections++;
      foreach (string connection in airportNode.unreachableConnections) {
        airportGraph[connection].isReachable = true;
      }
    }
    return numberOfNewConnections;
  }

  public class AirportNode {
    public string airport;
    public List<string> connections;
    public bool isReachable;
    public List<string> unreachableConnections;

    public AirportNode(string airport) {
      this.airport = airport;
      connections = new List<string>();
      isReachable = true;
      unreachableConnections = new List<string>();
    }
  }
}
```