Assignment no 6

Problem statement: There are flight paths between cities. If there is a flight between city A and city B then there is an edge between the cities. The cost of the edge can be the time that flight take to reach city B from A, or the amount of fuel used for the journey. Represent this as a graph. The node can be represented by airport name or name of the city. Use adjacency list representation of the graph or use adjacency matrix representation of the graph. Check whether the graph is connected or not. Justify the storage representation used.

```
#include <iostream>
#include <unordered_map>
#include <vector>
#include <set>

using namespace std;

class Graph {
   unordered_map<string, vector<pair<string, int>>> adjList;

public:
   void addEdge(string city1, string city2, int cost) {
      adjList[city1].push_back({city2, cost});
      adjList[city2].push_back({city1, cost}); // Since it's an undirected graph
   }

   void display() {
      for (auto& city : adjList) {
         cout << city.first << " -> ";
         for (auto& neighbor : city.second) {
            cout << "(" << neighbor.first << ", " << neighbor.second << ") ";
         }
         cout << endl;
```

```
        }
    }

    void dfs(string city, set<string>& visited) {
        visited.insert(city);
        for (auto& neighbor : adjList[city]) {
            if (visited.find(neighbor.first) == visited.end()) {
                dfs(neighbor.first, visited);
            }
        }
    }

    bool isConnected() {
        if (adjList.empty()) return false;

        set<string> visited;
        string startCity = adjList.begin()->first; // Pick any starting city

        dfs(startCity, visited);

        return visited.size() == adjList.size();
    }
};

int main() {
    Graph flights;
    flights.addEdge("Mumbai", "Delhi", 2);
    flights.addEdge("Delhi", "Bangalore", 3);
    flights.addEdge("Mumbai", "Chennai", 4);
    flights.addEdge("Chennai", "Kolkata", 5);
    flights.addEdge("Kolkata", "Delhi", 6);

    cout << "Adjacency List Representation:\n";
    flights.display();

    if (flights.isConnected()) {
        cout << "\nThe flight graph is connected.\n";
    } else {
        cout << "\nThe flight graph is NOT connected.\n";
    }

    return 0;
}
```

Output:

```
Adjacency List Representation:
Kolkata -> (Chennai, 5) (Delhi, 6)
Bangalore -> (Delhi, 3)
Delhi -> (Mumbai, 2) (Bangalore, 3) (Kolkata, 6)
Chennai -> (Mumbai, 4) (Kolkata, 5)
Mumbai -> (Delhi, 2) (Chennai, 4)
```