Airport Graph

Graph representation of airports in the US:

Your program needs to represent a graph with vertices/nodes that represent airports of the 20 cities in the US. Each node has a name — the name of the airport. For each node, record the information about the airport of that city.

Example:

```
    airports = [
        Airport("ATL", "Atlanta", "GA", 84.3880, 33.7490),
        Airport("AUS", "Austin", "TX", 97.7431, 30.2672),
        Airport("BOS", "Boston", "MA", 71.0096, 42.3656),
        Airport("BWI", "Baltimore", "MD", 76.6413, 39.2904),
        Airport("DCA", "Washington D.C.", "VA", 77.0369, 38.9072),
        Airport("DEN", "Denver", "CO", 104.9903, 39.7392),
        Airport("DFW", "Dallas", "TX", 96.7967, 32.7767),
        Airport("DTW", "Detroit", "MI", 83.0458, 42.3314),
        Airport("EWR", "Newark", "NJ", 74.1745, 40.7357),
        Airport("IAD", "Washington D.C.", "VA", 77.4558, 38.8821),
        Airport("IAH", "Houston", "TX", 95.3698, 29.7604),
        Airport("JFK", "New York City", "NY", 73.7781, 40.6413),
        Airport("LAS", "Las Vegas", "NV", 115.1398, 36.1699),
        Airport("LAX", "Los Angeles", "CA", 118.2437, 34.0522),
        Airport("MDW", "Chicago", "IL", 87.6298, 41.8781),
        Airport("MIA", "Miami", "FL", 80.1918, 25.7617),
        Airport("MSP", "Minneapolis and Saint Paul", "MN", 93.2650,
44.9778),
        Airport("PDX", "Portland", "OR", 122.6762, 45.5051),
        Airport("PHL", "Philadelphia", "PA", 75.1652, 39.9526),
        Airport("RDU", "Raleigh and Durham", "NC", 78.7870, 35.7796),
        Airport("SAN", "San Diego", "CA", 117.1611, 32.7157),
        Airport("SEA", "Seattle", "WA", 122.3321, 47.6062),
        Airport("SFO", "San Fransisco", "CA", 122.4194, 37.7749),
        Airport("SLC", "Salt Lake City", "UT", 111.8910, 40.7608)
    ]
```

Each edge represents a bi-directional connection between two airport connected with a direct delta flight. Each edge has a value that represents the distance between those the airports of those two cities. The distances between two airports are based on the geographic distance. (See a separate announcement for a shortcut you can use to compute this distance.)

```
    neighbours = [
    ["RDU", "AUS", "BOS", "BWI", "DCA", "DEN", "DFW", "DTW", "EWR", "IAD",
"IAH", "JFK", "LAS", "LAX", "MDW", "MIA", "MSP", "PDX", "PHL", "SAN",
"SEA", "SFO", "SLC"],
```

```
    ["ATL", "BOS", "DTW", "JFK", "RDU", "LAS", "LAX", "MSP", "SEA",
"SLC"],
    ["SLC", "RDU", "AUS", "ATL", "DTW", "MSP", "SEA", "LAX", "DEN", "JFK",
"IAD", "MIA", "PHL", "IAH", "LAS"],
    ["ATL", "DTW", "MSP", "JFK", "SLC", "BOS"],
    ["ATL", "JFK", "BOS", "DTW", "MSP", "SLC"],
    ["ATL", "MSP", "SLC", "DTW"],
    ["ATL", "MSP", "DTW", "SLC"],
    ["ATL", "MSP", "DTW", "SLC", "LAX"],
    ["BOS", "JFK", "ATL", "RDU", "MIA", "LAX", "SFO", "SEA"],
    ["ATL", "BOS", "DTW", "MSP", "RDU", "SLC"],
    ["ATL", "DTW", "SLC"],
    ["ATL", "DTW", "MSP", "SLC"],
    ["LAX", "ATL", "MIA", "DFW", "BOS", "SFO", "DTW", "SEA", "MSP"],
    ["ATL", "MSP", "DTW", "SLC", "LAX", "SEA", "SFO", "DFW"],
    ["JFK", "SFO", "SLC", "ATL", "LAS", "MSP", "BOS", "DTW", "SEA",
"PDX"],
    ["ATL", "DTW", "MSP"],
    ["ATL", "JFK", "LAX", "BOS", "DFW", "DCA"],
    ["ATL", "DEN", "DFW", "SEA", "LAS", "LAX"],
    ["SEA", "LAX", "SFO", "SLC", "MSP", "ATL", "DTW"],
    ["ATL", "BOS", "JFK", "DTW", "MSP", "SLC"],
    ["ATL", "BOS", "DTW", "MSP", "LAX", "SLC"],
    ["MSP", "SEA", "SLC", "ATL", "DTW", "JFK"],
    ["LAX", "SLC", "ATL", "MSP", "SFO", "DFW", "DEN", "LAS"],
    ["LAX", "JFK", "SAN", "SEA", "DEN", "SLC", "DFW", "PDX"],
    ["LAX", "DEN", "PHX", "SEA", "ATL", "MSP", "LAS"]
    ]
```

You can use **Haversine formula** to compute the distance between two geographic locations based on their latitude and longitude. Here is a sample code:

```python
import math

def haversine(lon1, lat1, lon2, lat2):
    # Radius of the Earth in kilometers
    R = 6371.0

    # Convert latitude and longitude from degrees to radians
    lat1_rad = math.radians(lat1)
    lon1_rad = math.radians(lon1)
    lat2_rad = math.radians(lat2)
    lon2_rad = math.radians(lon2)

    # Differences in coordinates
    dlat = lat2_rad - lat1_rad
    dlon = lon2_rad - lon1_rad

    # Haversine formula
    a = math.sin(dlat / 2)**2 + math.cos(lat1_rad) * math.cos(lat2_rad) *
math.sin(dlon / 2)**2
```

```
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))

    # Distance in kilometers
    distance = R * c
    return distance

# Example usage
lat1 = 52.5200  # Latitude of Berlin
lon1 = 13.4050  # Longitude of Berlin
lat2 = 48.8566  # Latitude of Paris
lon2 = 2.3522   # Longitude of Paris

distance = haversine(lat1, lon1, lat2, lon2)
print(f"Distance: {distance:.2f} km")
```

Algorithms:

You need to implement 3 algorithm to find the path between two airports:

1. Breadth First Search algorithm
2. Depth First Search algorithm, and
3. Dijkstra shortest path algorithm.

Requirements:

The customer wants a program that can find the path between two airports. The program should have the following features:

1. The User should be able to select the origin and the destination of the desired trip.
2. The User should be able to select which of the 3 algorithms is to be used for finding a path from source to destination.
3. The user should be able to see the resulting path and the total cost of the path.

Resources:

You cand find our helper code on Canvas. The code contains the following:

1. A simple GUI for selecting the origin and destination point for the graph traversal algorithm between the 20 airports, and selecting one of the three algorithms to be executed.
2. It contains a comments on what needs to be implemented in separates classes.