

**CIS 211: Data Structures | Delaware Technical Community College | Wilmington Campus**  
**Assignment 10 - Graph ADT**

**Code will be graded on:**

- 70% Code quality/Correct usage of programming concepts
  - 50% Correct implementation/usage of graph ADT
  - 20% Correct implementation of logic/helper methods for application
- 10% Correct answer/Program runs
- 5% Descriptive variable names
- 5% User friendliness (good prompts and outputs)
- 5% Comments (minimum required comments and human readable clarification for any unclear or unintuitive code)
- 5% Proper indentation/formatting






**Introduction:**

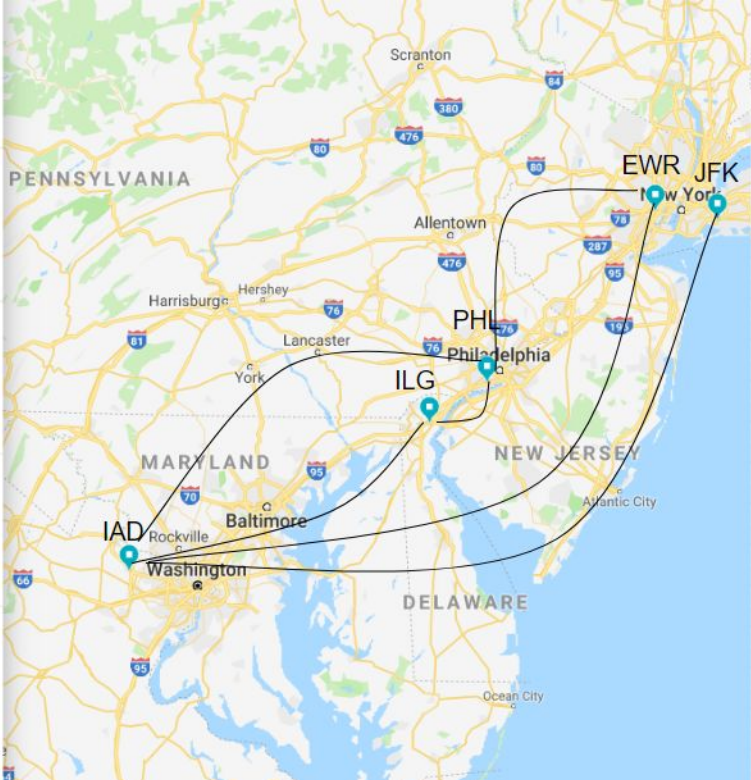
Throughout the semester, we've explored various ADTs that are very rigid in nature. Stacks, queues, binary search trees, while extremely useful, are somewhat limited in their application in real world data. The graph data structure is a very flexible ADT that can organize data into connected nodes with virtually no restrictions. Graphs give us a very useful ADT that can facilitate storage and algorithms for map-like data.

**Assignment:**

Implement and initialize a unweighted, undirected graph to store the information of possible flights between local airports.

Then write a method that will take one parameter for vertex and return a list of adjacent vertices. Write a driver program that tests a few airports to make sure the correct list of adjacent airports is returned.

<b>Newark Liberty International Airport</b> 3.1 ★★★★★ (5,310) Airport · Newark Long-standing NYC-area hub	
<b>Philadelphia International Airport</b> 3.3 ★★★★★ (2,805) Airport · Philadelphia Hub with rail service & kids' play areas	
<b>New Castle Airport</b> 4.0 ★★★★★ (45) Airport · New Castle	
<b>Dulles International Airport</b> 3.9 ★★★★★ (3,008) Airport · Dulles Airport serving Washington, D.C. area	
<b>John F. Kennedy International Airport</b> 3.7 ★★★★★ (8,932) Airport Airport serving the New York City area	



### Sample test code:

```

ArrayList<String> airportList = new ArrayList<>();
airportList.add("JFK");
airportList.add("EWR");
airportList.add("PHL");
airportList.add("ILG");
airportList.add("IAD");

Airports airports = new Airports(airportList);

airports.addRoute("JFK", "IAD");
airports.addRoute("IAD", "ILG");
airports.addRoute("ILG", "PHL");
airports.addRoute("IAD", "PHL");
airports.addRoute("IAD", "EWR");
airports.addRoute("PHL", "EWR");

// print the adjacency list representation of
// the whole graph
airports.printGraph();

```

```
// print the adjacency of selected airports
airports.printAdjacent("PHL");
airports.printAdjacent("ILG");
```

**Output:**

Printing whole graph as adjacency list:

JFK-->

IAD

EWR-->

IAD    PHL

PHL-->

ILG    IAD    EWR

ILG-->

IAD    PHL

IAD-->

JFK    ILG    PHL    EWR

Adjacency list of airport PHL

ILG    IAD    EWR

Adjacency list of airport ILG

IAD    PHL

**Resources:**

- <https://www.geeksforgeeks.org/graph-and-its-representations/>