Day31 – The Graph ADT

Chapter 10, 10.1 – Introduction to Graphs, Review Problem(s): ***7, 8, 10, 11***

Chapter 10, 10.2 – The Graph Interface

Chapter 10, 10.3 – Implementation of Graphs, Review Problem(s): ***16***

Graphs are data structures that contain a set of node (or vertices) and a set of edges between nodes and which relate the nodes to each other.

Trees, which we discussed in Chapter 07 and then again in Chapter 09 are subsets of Graphs. In particular, Trees are connected Graphs without any circuits.

Every Tree is a Graph, but not every Graph is a Tree.

Give two Graphs which are Trees, and two Graphs which are not Trees.

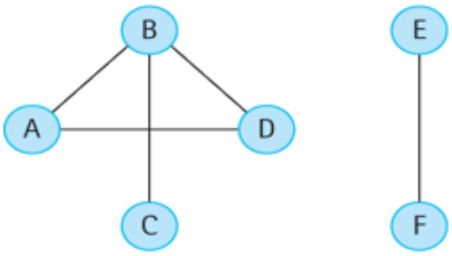
An ***undirected*** graph is a graph where the edges have no direction, and a ***directed graph*** is a graph in which each edge is directed from one vertex to another (or the same vertex).

Note the additional terms in the Chapter 10 PPT:

- Adjacent vertices
- Path
- Complete graph (***example***)
- Weighted graph (***example***)

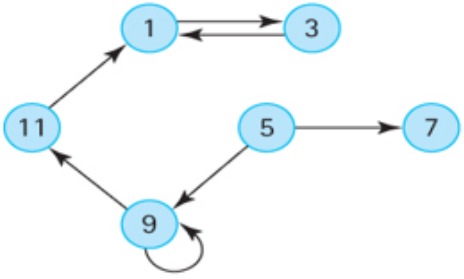
Chapter 10 PPT, pg 01-09.

(a) Graph1 is an undirected graph



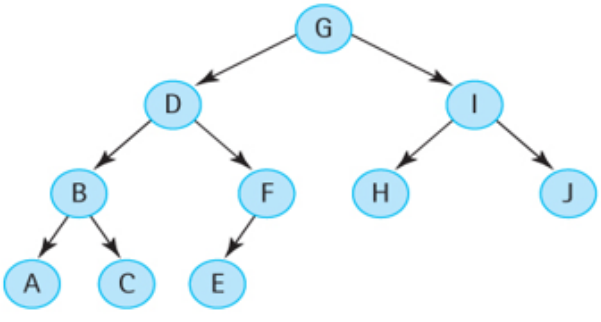
V(Graph1) = {A, B, C, D, E, F}
E(Graph1) = {(A, B), (A, D), (B, C), (B, D), (E, F)}

(b) Graph2 is a directed graph



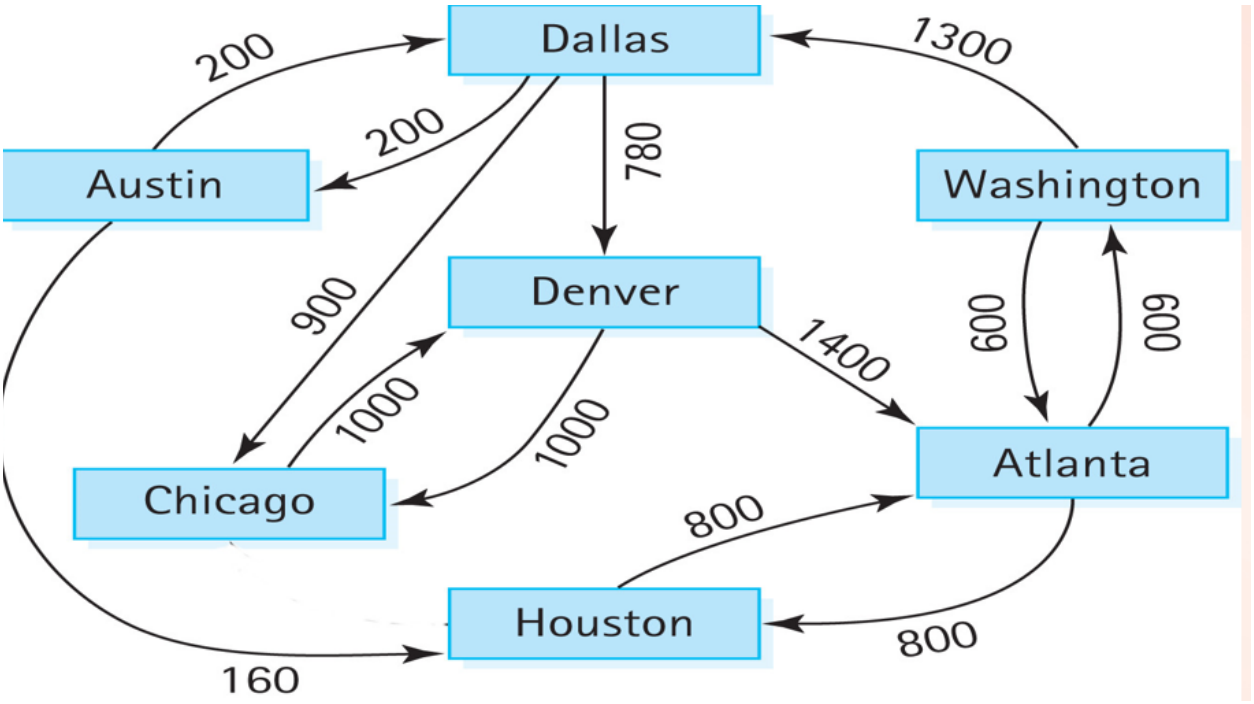
V(Graph2) = {1, 3, 5, 7, 9, 11}
E(Graph2) = {(1, 3), (3, 1), (5, 7), (5, 9), (9, 11), (9, 9), (11, 1)}

(c) Graph3 is a directed graph



V(Graph3) = {A, B, C, D, E, F, G, H, I, J}
E(Graph3) = {(G, D), (G, I), (D, B), (D, F), (I, H), (I, J), (B, A), (B, C), (F, E)}

Consider the weighted graph:



What kind of questions might we ask about a weighted graph?

1. Does a path exist between vertex A and vertex D?
   or
   Can we fly from Atlanta to Denver?

2. What is the total weight along this path from A to D?
   or
   What is the total distance from Atlanta to Denver?

3. What is the shortest path from A to D?
   or
   What is the shortest route from Atlanta to Denver?

4. If I start at vertex A, where can I go?
   or
   What cities are accessible if I start in Atlanta?

5. How many connected components are in the graph?
   or
   What groups of cities are connected to each other?

Graphs can be implemented using either Arrays or Lists. In these implementations we specify and implement a small set of useful graph operations. While many

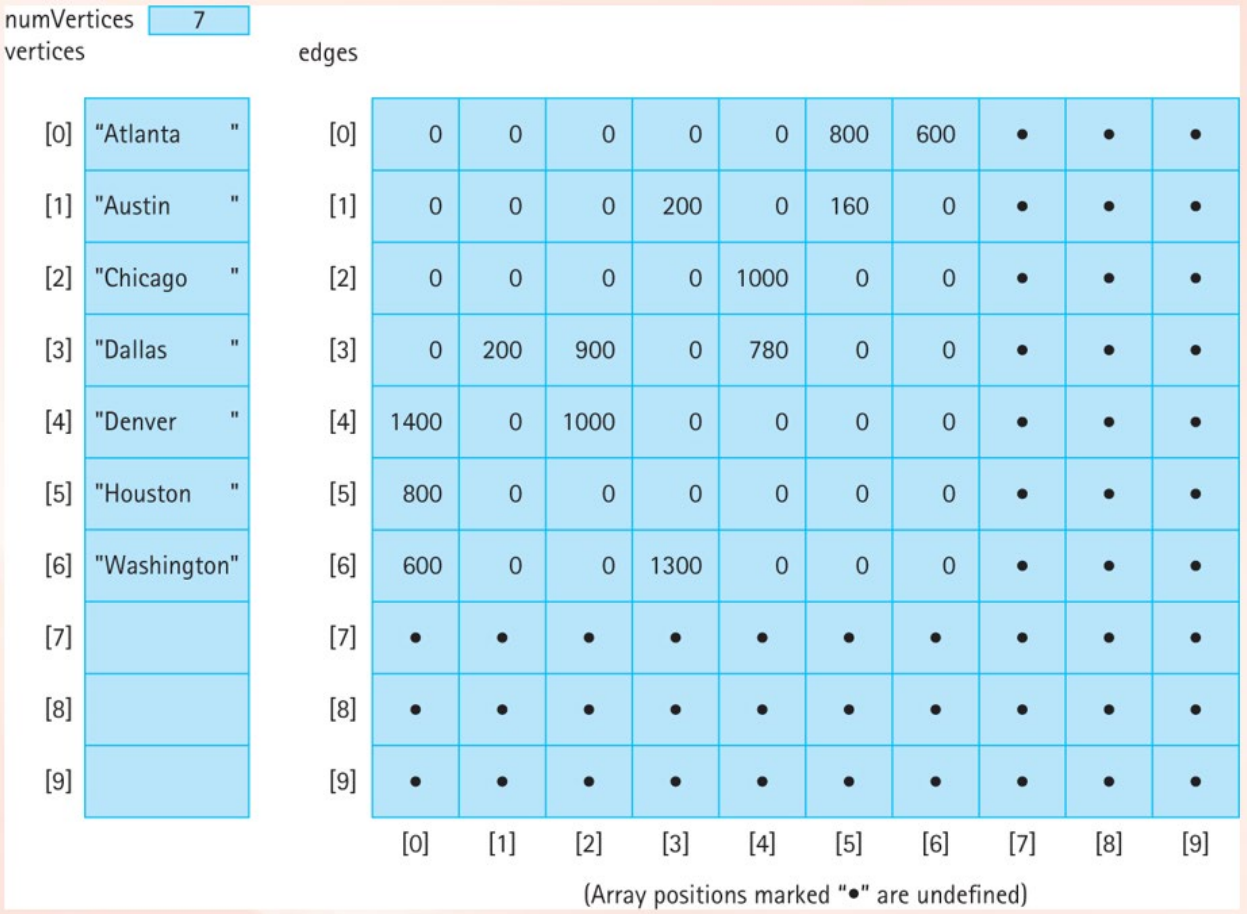
other graph operations could be implemented, the ones here were chosen to help answer questions as those above.

Methods in the Graph ADT include:

- isEmpty()
- isFull()
- addVertex()
- hasVertex()
- addEdge()
- weightIs()

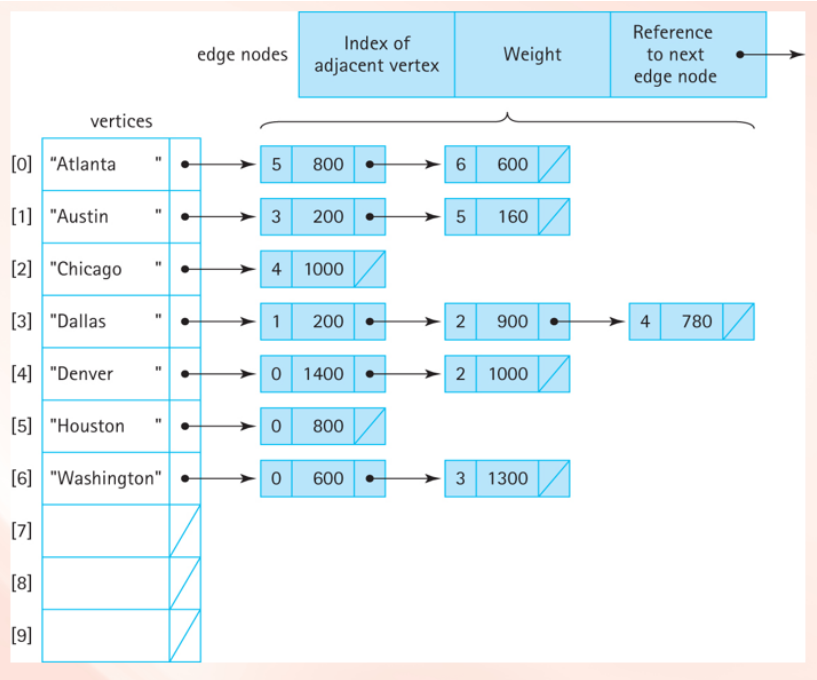
In an Array-Based implementation, an ***Adjacency Matrix*** is used. In a graph with ***N*** vertices the Adjacency Matrix would be an ***N x N*** table, where non-zero entries represent edges between two nodes.

The previous graph of air routes would be represented with the following Adjacency Matrix. Here the non-zero entries are the weights, which is distance for this example.

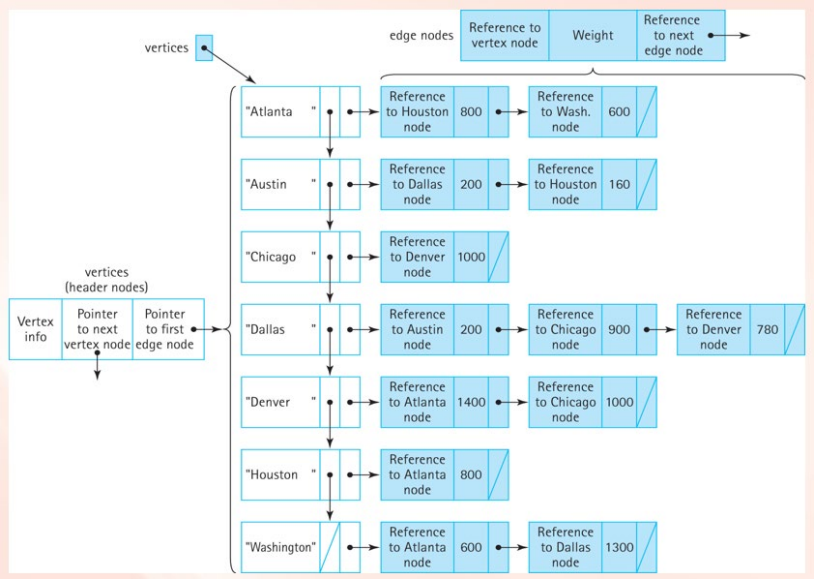


In an List-Based implementation, an ***Adjacency List*** is used. It is a linked list that identifies all vertices adjacent to a particular vertex.

We could use an array of vertices that each contain a reference to a linked list of nodes:



Or we could use a linked list of vertices that each contain a reference to a linked list of nodes:



Chapter 10 PPT, pg 15-25.

Here we will expect you to understand graphs, operations on graphs, and the structures used to implement them.