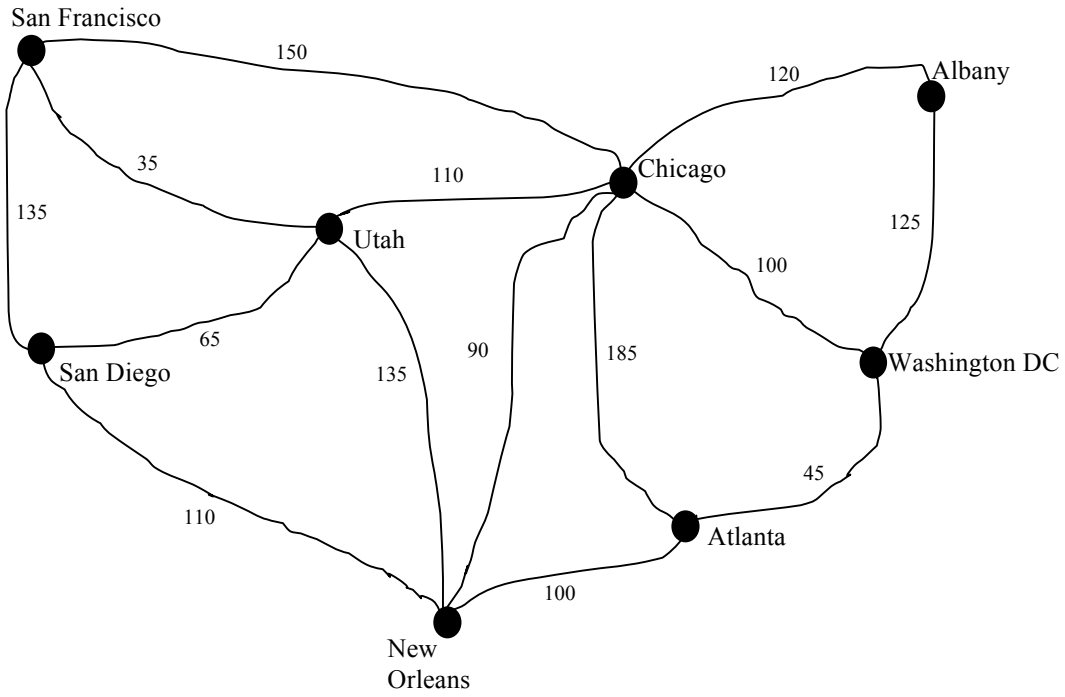
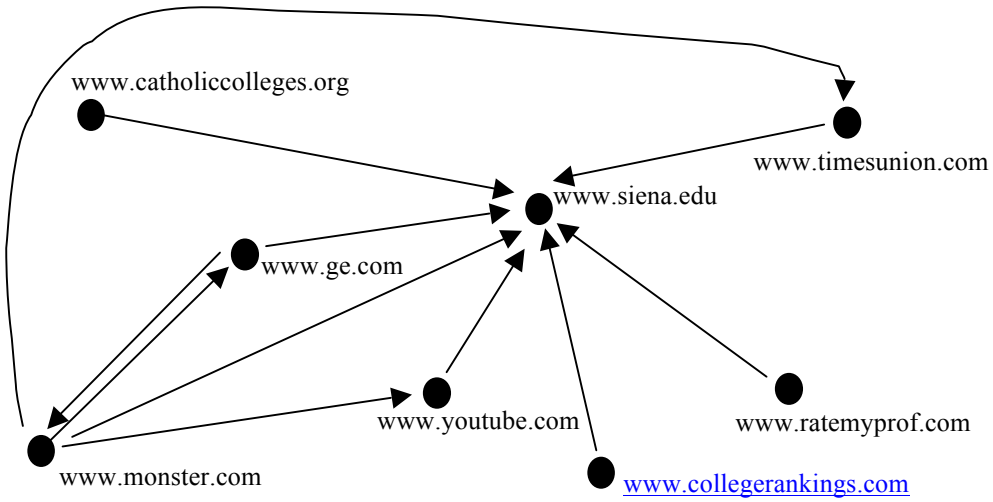
Graphs


San Francisco
150
120
Albany
35
110
Chicago
135
125
Utah
100
65
90
185
Washington DC
San Diego
135
45
110
Atlanta
100
New
Orleans


www.catholiccolleges.org
www.timesunion.com
www.siena.edu
www.ge.com
www.youtube.com
www.ratemyprof.com
www.monster.com
www.collegerankings.com

Graph:

Directed graph:

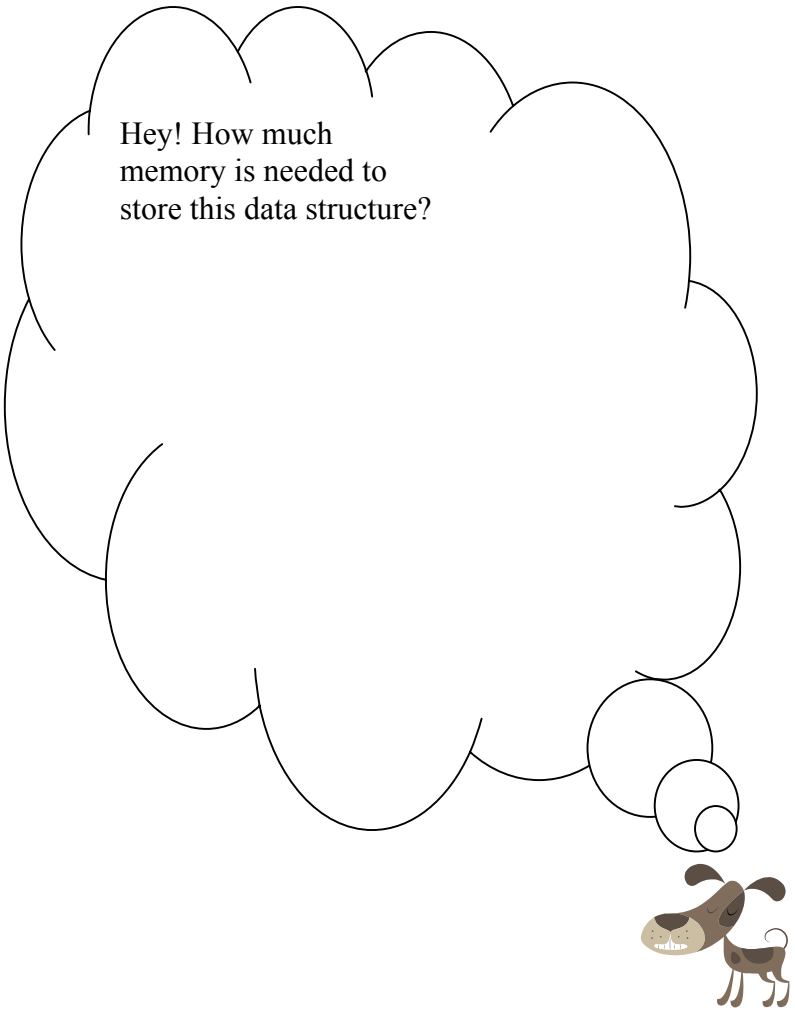
Undirected graph:

Weighted graph:

Adjacency Matrix Representation for Graphs

	0	1	2	3	4
0	F	T	F	T	F
1	T	F	T	T	F
2	F	T	F	T	F
3	T	T	T	F	T
4	F	F	F	T	F

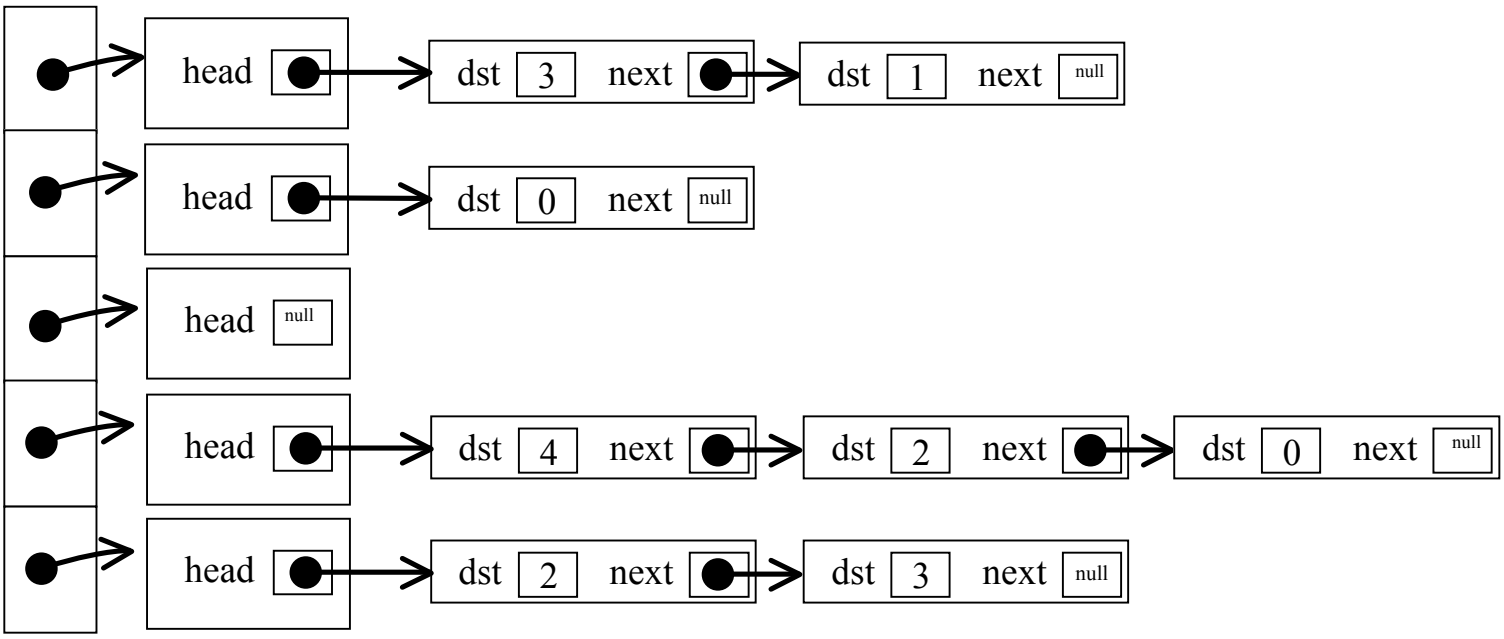
If adjmatrix[i][j] is true, then there is an edge from vertex i to vertex j. Below, draw the graph represented in the adjacency matrix above.



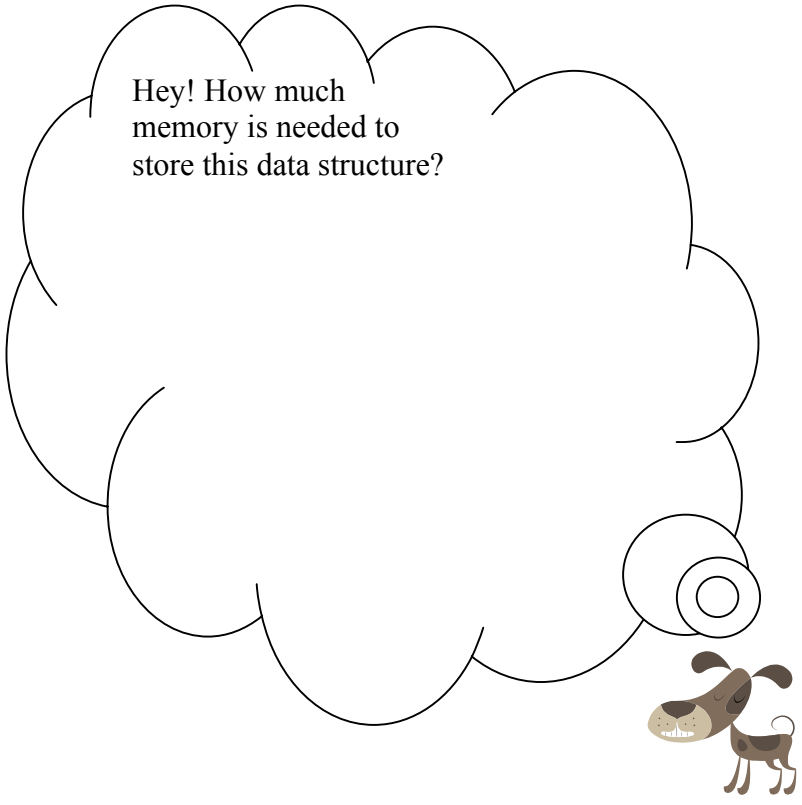
--- Implementation Adjacency Matrix ---

```
public class Digraph {
   private boolean[][] adjMatrix;


   public Digraph( int numVerts ) {









   }
   public void addEdge( int src, int dst ){









   }
   public void removeEdge( int src, int dst ) {









   }
```

Adjacency Lists Representation for Graphs



Each element in the 1D array represents a vertex in the graph. I.e., `vertices[i]` stores a reference to a Vertex object. The Vertex object stores a reference (called `head`) to the first node in a linked list; each node in the linked list represents a directed edge whose source vertex is `i` and whose destination vertex is stored in `dst`. Below, draw the graph represented by the adjacency lists data structure above.



--- Implementation Adjacency Lists ---

```
public class Digraph {
   private Vertex[] vertices;

   private class Vertex {
       private Edge head;
   }
   private class Edge {
       private int dst;
       private Edge next;
       private Edge(int d, Edge n) {dst=d; next=n;}
   }
```

Example of how the methods in this implementation of Digraph work:

```
Digraph g = new Digraph( 4 );
g.addEdge( 1, 3 );
g.addEdge( 1, 0 );
g.addEdge( 1, 2 );
g.removeEdge( 1, 0 );
```

--- Implementation Adjacency Lists ---


```
public class Digraph {
   private Vertex[] vertices;

   private class Vertex {
       private Edge head;
   }
   private class Edge {
       private int dst;
       private Edge next;
       private Edge(int d, Edge n) {dst=d; next=n;}
   }
   public Digraph( int numVerts ) {









   }
   public void addEdge( int src, int dst ) {









   }
```

--- Implementation Adjacency Lists ---

```
public void removeEdge( int src, int dst ) {
```

--- Implementation Adjacency Lists ---


```
public class Digraph {
   private Vertex[] vertices;

   private class Vertex {
       private Edge head;
   }
   private class Edge {
       private int dst;
       private Edge next;
       private Edge(int d, Edge n) {dst=d; next=n;}
   }
   public Digraph( int numVerts ) {









   }
   public void addEdge( int src, int dst ) {









   }
```

--- Implementation Adjacency Lists ---

```
public void removeEdge( int src, int dst ) {
```

	Adjacency Matrix	Adjacency Lists
memory		
addEdge(src,dst)		
removeEdge(src,dst)		