Read Chap S7.1-7.4

**Definition** A graph \( G \) is a finite nonempty set of vertices \( V \) and a set of edges \( E \).

Example: \( G = (V,E), V = \{0, 1, 2, 3, 4, 5\}, E = \{(1, 2), (1, 3), (2, 3), (2, 4), (5, 0)\} \)

A **directed** graph is a graph where the edges are ordered.

Example: \( G = (V,E), V = \{1, 2, 3, 4, 5\}, E = \{(1, 2), (3, 1), (2, 3), (2, 1), (2, 4)\} \)

**Trees**

A tree is a connected directed acyclic graph such that:

- There is exactly one vertex (root) that has no edges entering.
- Each vertex except the root has exactly one entering edge.
- There is a path from the root to each node.

**Graphs model lots of problems**
Implementation 1: Adjacency Matrix

Let \( A \) be a matrix of weights \( w(i,j) \).

If edge \((i,j)\) then \( A[i][j] = w(i,j) \) and \( A[j][i] = w(i,j) \)

Otherwise, \( A[i][j] = A[j][i] = 0 \)

If no weights on edges, then an edge means weight is 1, no edge has weight 0.

Example:

Adjacency matrix of first graph on handout.

\[
\begin{array}{c|ccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 \\
\hline
0 &  &  &  &  &  & \\
1 &  &  &  &  &  & \\
2 &  &  &  &  &  & \\
3 &  &  &  &  &  & \\
4 &  &  &  &  &  & \\
5 &  &  &  &  &  & \\
\end{array}
\]

Implementation 2: Adjacency list

Some Operations on Graph G

- InsertVertex(v)
- InsertEdge(v1, v2)
- DeleteVertex(v)
- DeleteEdge(v1, v2)
- IsEdge(v1, v2)
Depth First Search (DFS)
Print the nodes in a graph in depth-first order, going as deep as you can on a path, and then backtracking.

```c
struct nodeg {
    int data;
    int mark;
    nodea * adj;
}

struct nodea {
    int data;
    nodea * adj;
}

Vector<nodeg> graph(SIZE);    // an array of nodes with adjacency lists

DFS(v)
    mark v as visited
    print v
    for each unvisited vertex u adjacent to v
        DFS(u)

main()
    mark all nodes as unmarked
    DFS(v) on a vertex v
```

Breadth-First Search (BFS)
Print the nodes in a graph in breadth-first order, fanning out, visiting closest nodes first.
BFS(v)
Queue <vertex> Q;  // where vertex is int or string
unmark all vertices;
Q.Enqueue(v);  // for some vertex v
mark v as visited

while (!Q.IsEmpty())
  w = Q.GetFront()
  Q.Dequeue()
  for each unvisited u adj. to w
    mark u as visited
    Q.Enqueue(u)

Bridges of Konigsberg

Traveling Rock Band

Problem: A local rock band wants to go on tour, and they will travel by bus. They don’t like riding on the bus, so they give you a list of the n cities they plan to visit, and your task is to schedule the tour so that they visit each city exactly once and travel the fewest miles.

<table>
<thead>
<tr>
<th>No. of Cities</th>
<th>No. of Tours</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>