COMP251: DATA STRUCTURES & ALGORITHMS

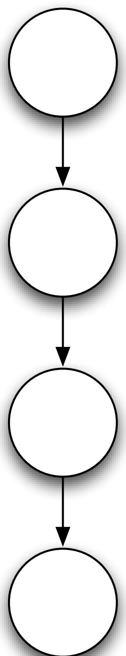
Instructor: Maryam Siahbani

Computer Information System
University of Fraser Valley

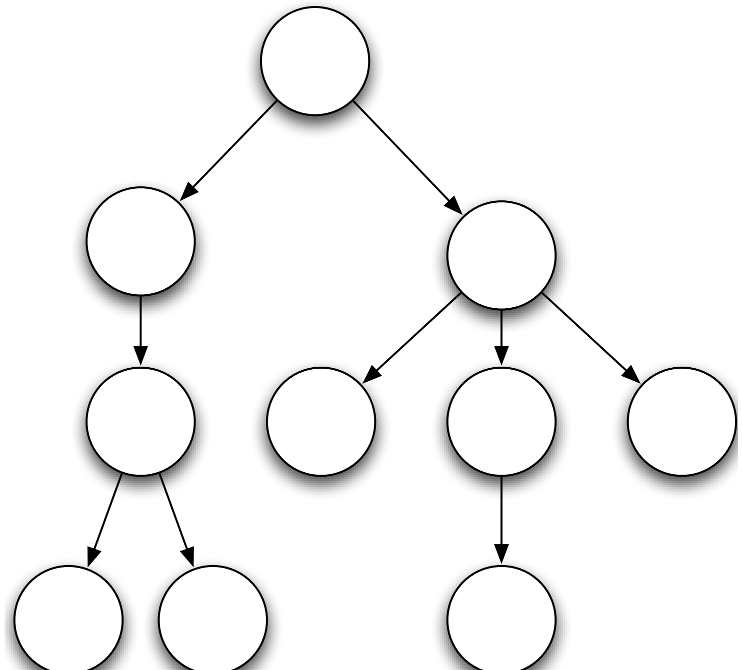
* Some slides from “Algorithms and Data Structures” by Douglas Wilhelm Harder

Introduction to Graphs

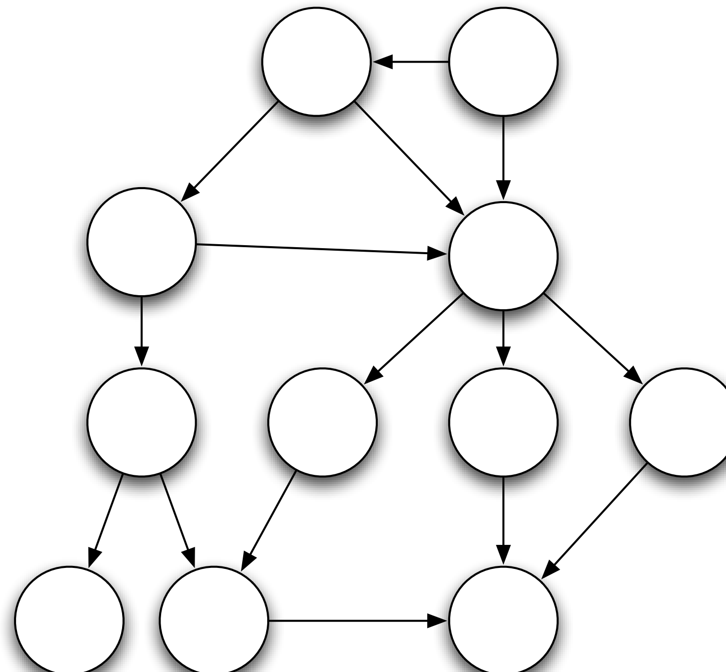
Graphs



Linked List

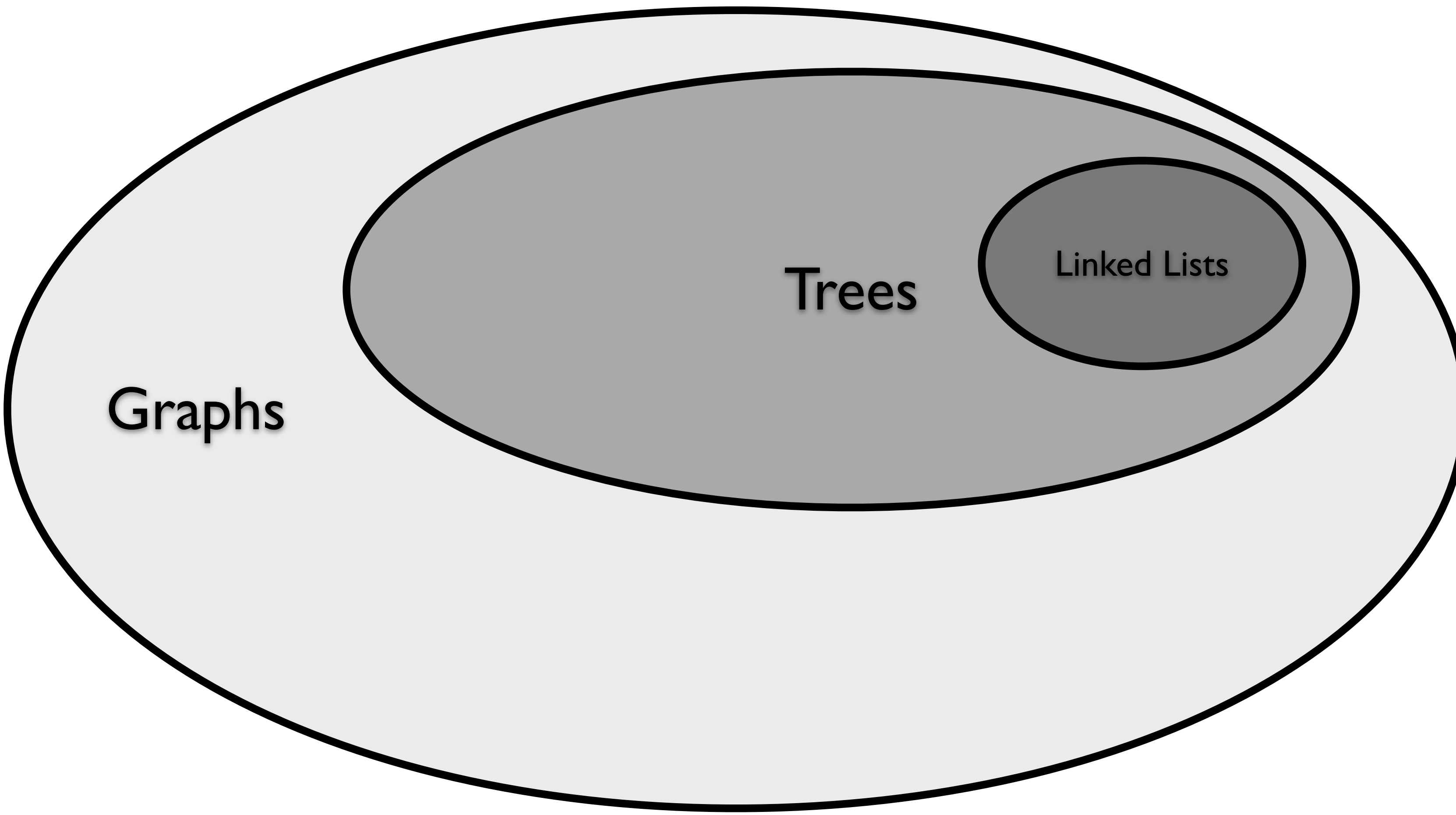


Tree



Graph

Graphs



Graphs

We will define an Undirected Graph ADT as a collection of *vertices*

$$V = \{v_1, v_2, ..., v_n\}$$

- The number of vertices is denoted by

$$|V| = n$$

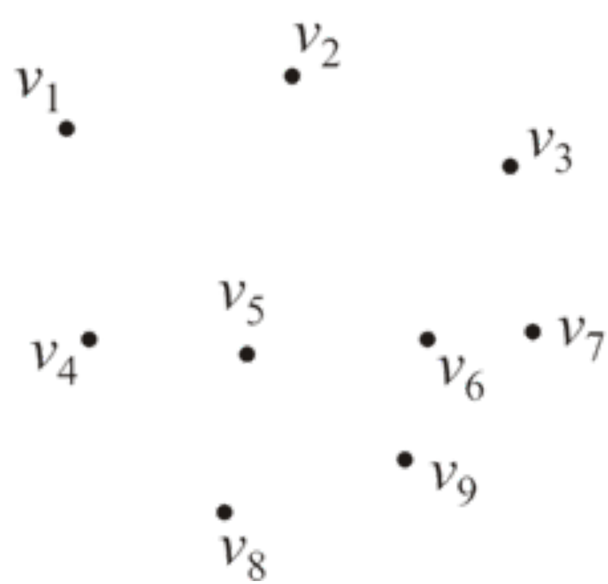
- Associated with this is a collection E of unordered pairs $\{v_i, v_j\}$ termed *edges* which connect the vertices

Graphs

Consider this collection of vertices

$$V = \{v_1, v_2, ..., v_9\}$$

where $|V| = n$

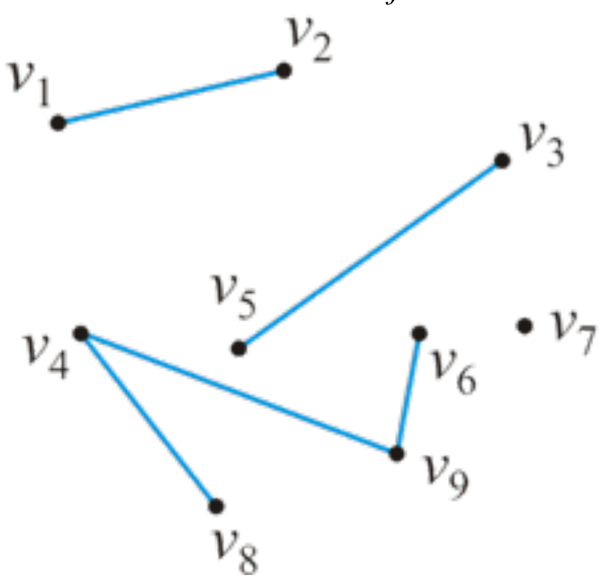


Undirected graphs

Associated with these vertices are $|E| = 5$ edges

$$E = \{\{v_1, v_2\}, \{v_3, v_5\}, \{v_4, v_8\}, \{v_4, v_9\}, \{v_6, v_9\}\}$$

– The pair $\{v_j, v_k\}$ indicates that both vertex v_j is adjacent to vertex v_k and vertex v_k is adjacent to vertex v_j



Undirected graphs

We will assume in this course that a vertex is never adjacent to itself

- For example, $\{v_1, v_1\}$ will not define an edge

The maximum number of edges in an undirected graph is

$$|E| \le \binom{|V|}{2} = \frac{|V|\left(|V|-1\right)}{2} = \mathrm{O}\left(|V|^2\right)$$

A ***complete graph*** has an edge between every pair of nodes.

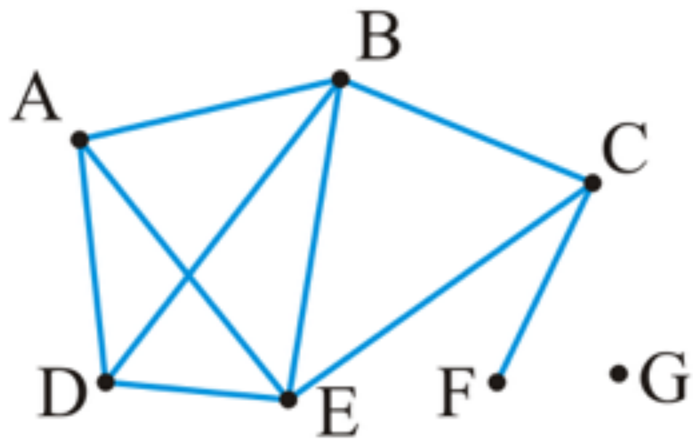
An undirected graph

Example: given the $|V| = 7$ vertices

$$V = \{A, B, C, D, E, F, G\}$$

and the $|E| = 9$ edges

$$E = \{\{A, B\}, \{A, D\}, \{A, E\}, \{B, C\}, \{B, D\}, \{B, E\}, \{C, E\}, \{C, F\}, \{D, E\}\}$$



Degree

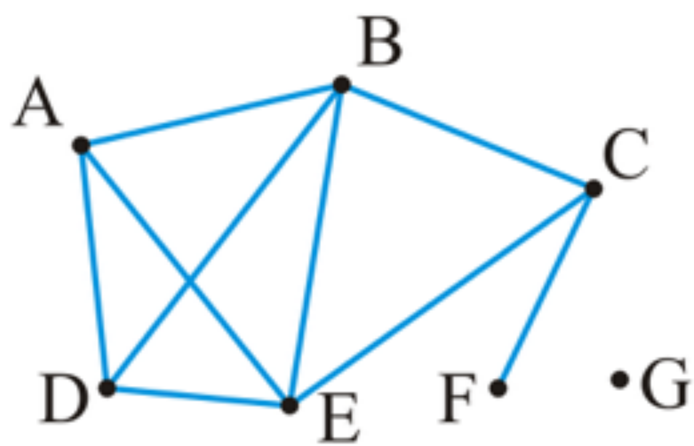
The degree of a vertex is defined as the number of adjacent vertices

degree(A) = degree(D) = degree(C) = 3

degree(B) = degree(E) = 4

degree(F) = 1

degree(G) = 0



Those vertices adjacent to a given vertex are its *neighbors*

Paths

A path in an undirected graph is an ordered sequence of vertices

$$(v_0, v_1, v_2, ..., v_k)$$

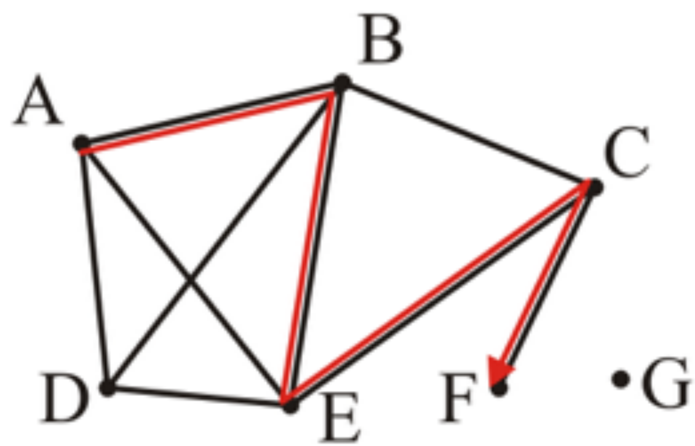
where $\{v_{j-1}, v_j\}$ is an edge for $j = 1, ..., k$

- Termed ***a path from*** v_0 to v_k
- The length of this path is k

Paths

A path of length 4:

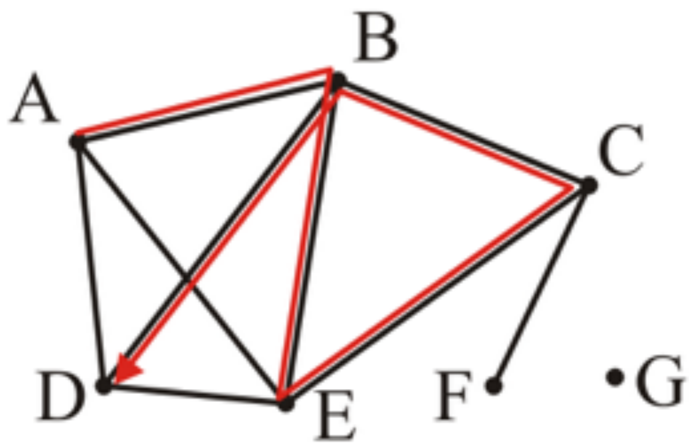
(A, B, E, C, F)



Paths

A path of length 5:

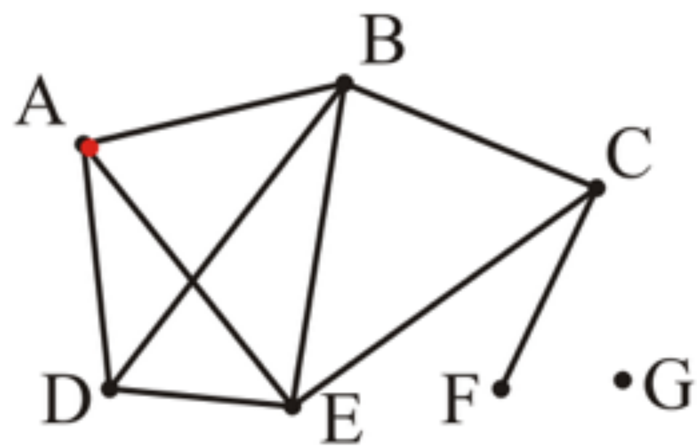
(A, B, E, C, B, D)



Paths

A *trivial* path of length 0:

(A)



Simple paths

A ***simple path*** has no repetitions other than perhaps the first and last vertices

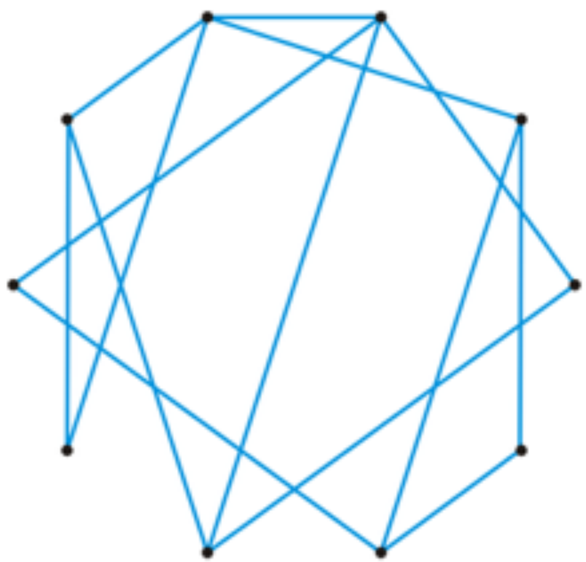
A ***simple cycle*** is a simple path of at least two vertices with the first and last vertices equal

- Note: these definitions are not universal

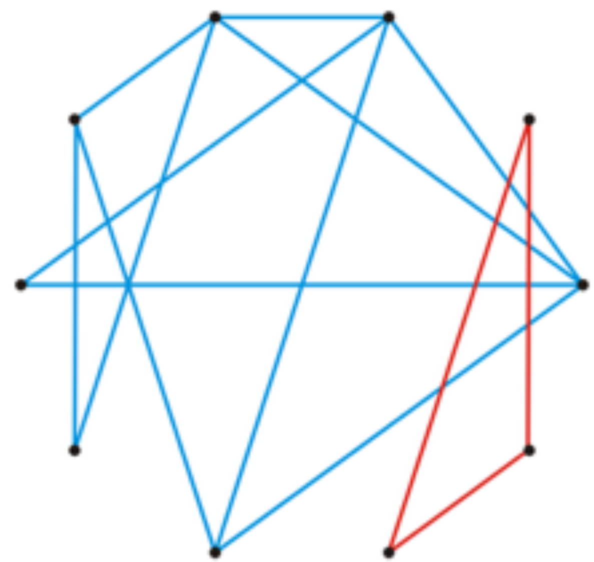
Connectedness

Two vertices v_i, v_j are said to be ***connected*** if there exists a path from v_i to v_j

A graph is connected if there exists a path between any two vertices



A connected graph



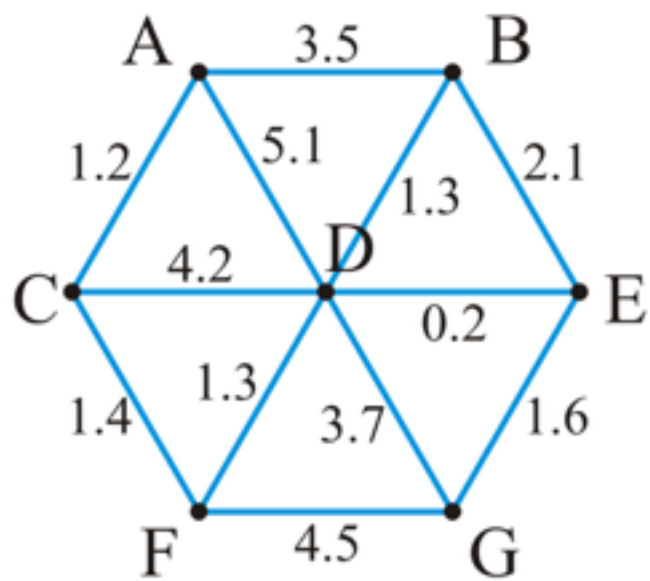
An unconnected graph

Weighted graphs

A weight may be associated with each edge in a graph

- This could represent distance, energy consumption, cost, etc.
- Such a graph is called a ***weighted graph***

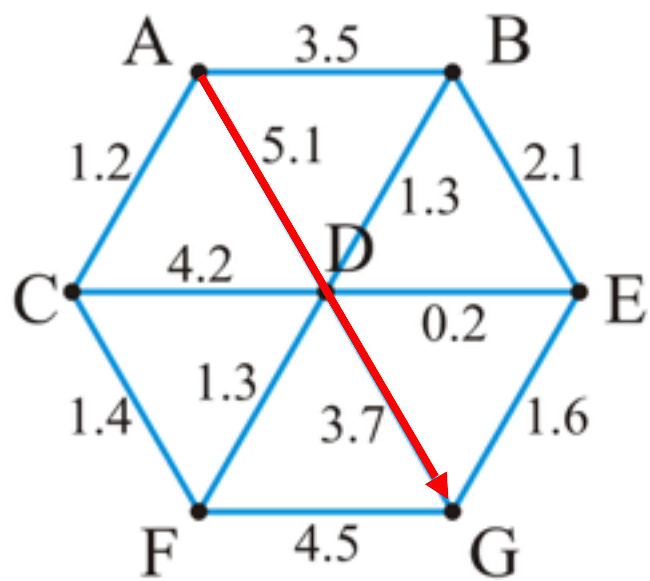
Pictorially, we will represent weights by numbers next to the edges



Weighted graphs

The *length* of a path within a weighted graph is the sum of all of the edges which make up the path

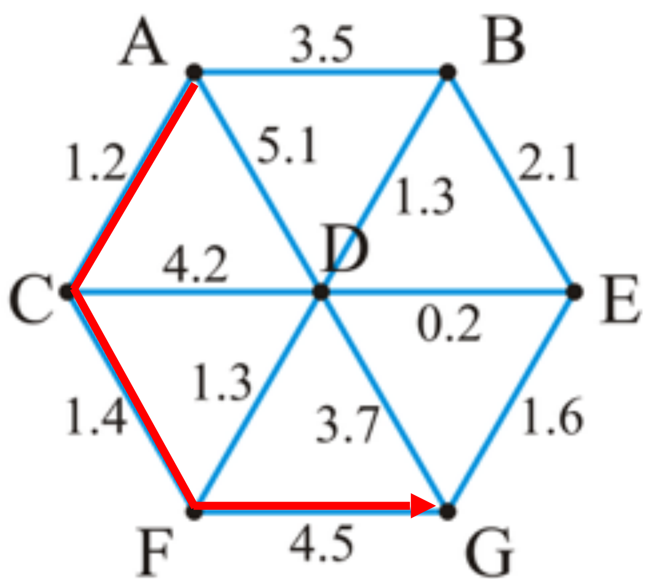
– The length of the path (A, D, G) in the following graph is $5.1 + 3.7 = 8.8$



Weighted graphs

Different paths may have different weights

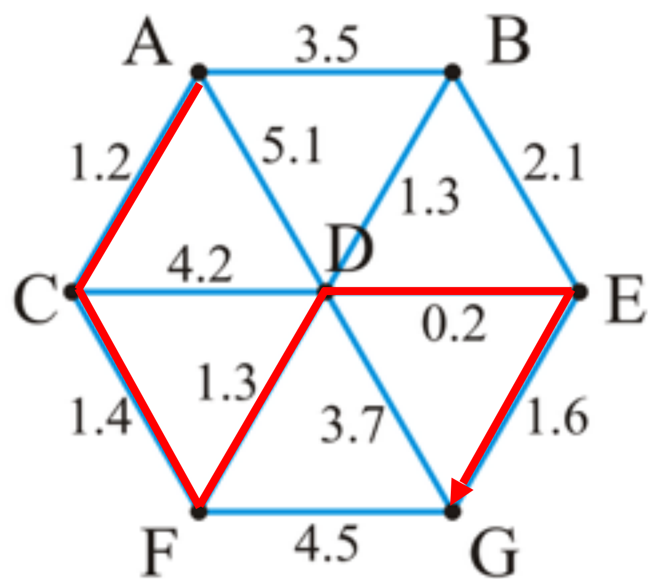
- Another path is (A, C, F, G) with length $1.2 + 1.4 + 4.5 = 7.1$



Weighted graphs

Problem: find the shortest path between two vertices

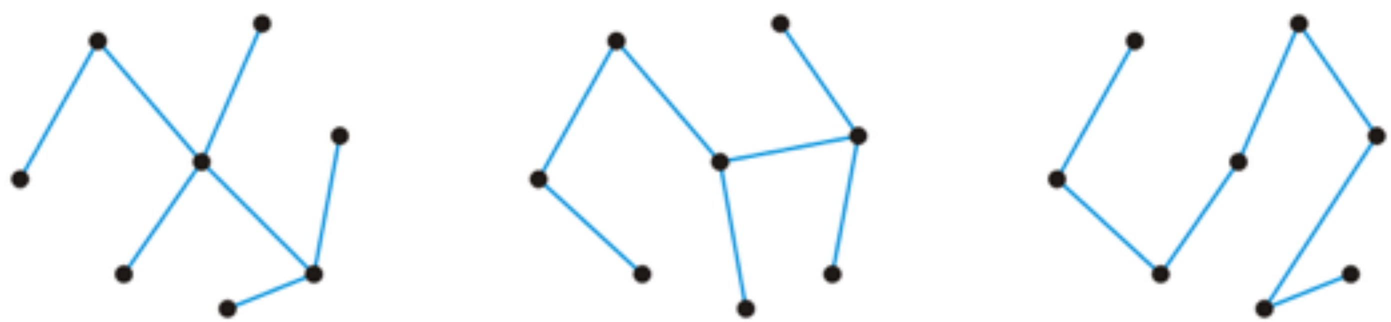
– Here, the shortest path from A to H is (A, C, F, D, E, G) with length 5.7



Trees

A graph is a ***tree*** if it is connected and there is a unique path between any two vertices

– Three trees on the same eight vertices



Consequences:

– The number of edges is $|E| = |V| - 1$

– The graph is ***acyclic***, that is, it does not contain any cycles

– Adding one more edge must create a cycle

– Removing any one edge creates two disjoint non-empty sub-graphs

Directed graphs

In a *directed graph*, the edges on a graph are be associated with a direction

- Edges are ordered pairs (v_j, v_k) denoting a connection from v_j to v_k
- The edge (v_j, v_k) is different from the edge (v_k, v_j)

Streets are directed graphs:

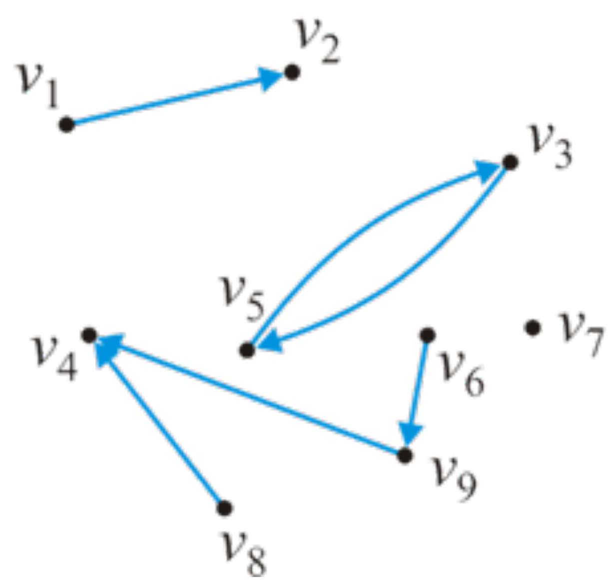
- In most cases, you can go two ways unless it is a one-way street

Directed graphs

Given our graph of nine vertices $V = \{v_1, v_2, \ldots v_9\}$

– These six pairs (v_j, v_k) are *directed edges*

$$E = \{(v_1, v_2), (v_3, v_5), (v_5, v_3), (v_6, v_9), (v_8, v_4), (v_9, v_4)\}$$



Directed graphs

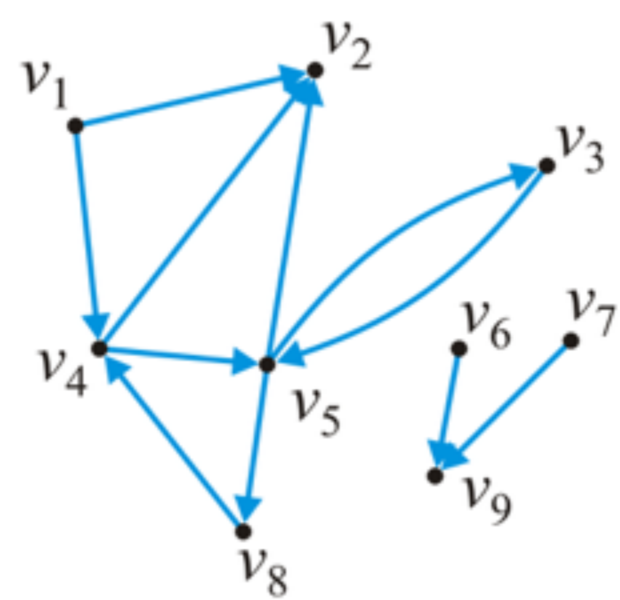
The maximum number of directed edges in a directed graph is

$$\left|E\right| \leq 2\binom{\left|V\right|}{2} = 2\frac{\left|V\right|\left(\left|V\right|-1\right)}{2} = \left|V\right|\left(\left|V\right|-1\right) = \mathrm{O}\left(\left|V\right|^2\right)$$

In and out degrees

The degree of a vertex must be modified to consider both cases:

- The ***out-degree*** of a vertex is the number of vertices which are adjacent to the given vertex
- The ***in-degree*** of a vertex is the number of vertices which this vertex is adjacent to



In this graph:

$\text{in\_degree}(v_1) = 0 \qquad \text{out\_degree}(v_1) = 2$

$\text{in\_degree}(v_5) = 2 \qquad \text{out\_degree}(v_5) = 3$

Paths

A path in a directed graph is an ordered sequence of vertices

$$(v_0, v_1, v_2, ..., v_k)$$

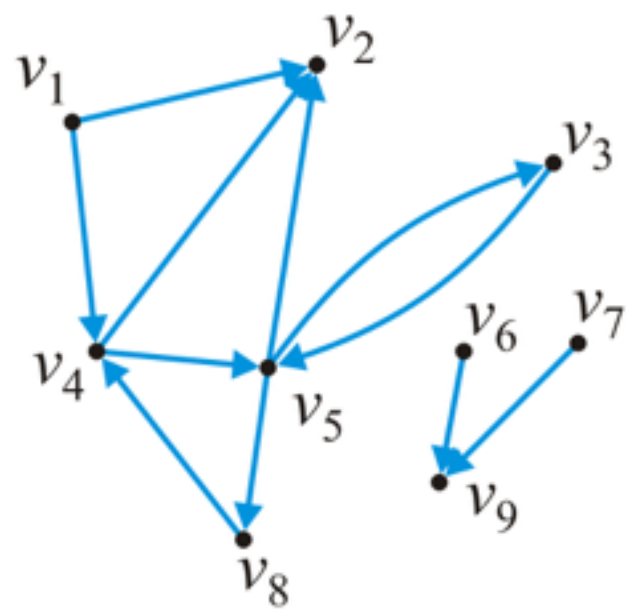
where (v_{j-1}, v_j) is an edge for $j = 1, ..., k$

A path of length 5 in this graph is

$$(v_1, v_4, v_5, v_3, v_5, v_2)$$

A simple cycle of length 3 is

$$(v_8, v_4, v_5, v_8)$$



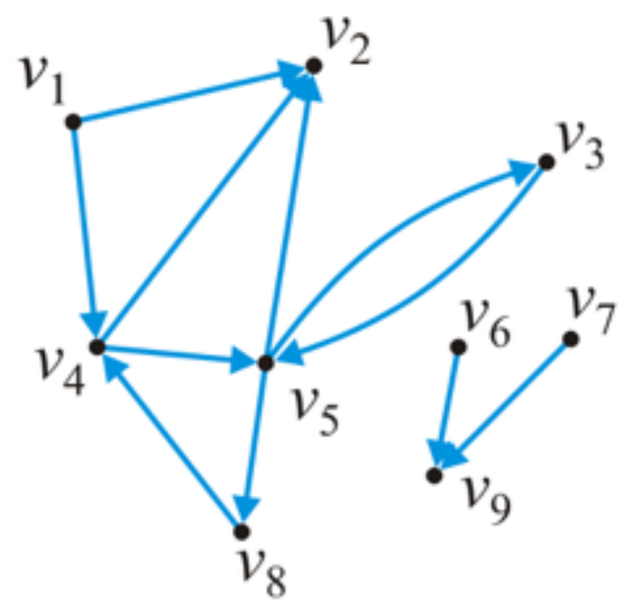
Connectedness

Two vertices v_j, v_k are said to be *connected* if there exists a path from v_j to v_k

- A graph is ***strongly connected*** if there exists a directed path between any two vertices
- A graph is ***weakly connected*** there exists a path between any two vertices that ignores the direction

In this graph:

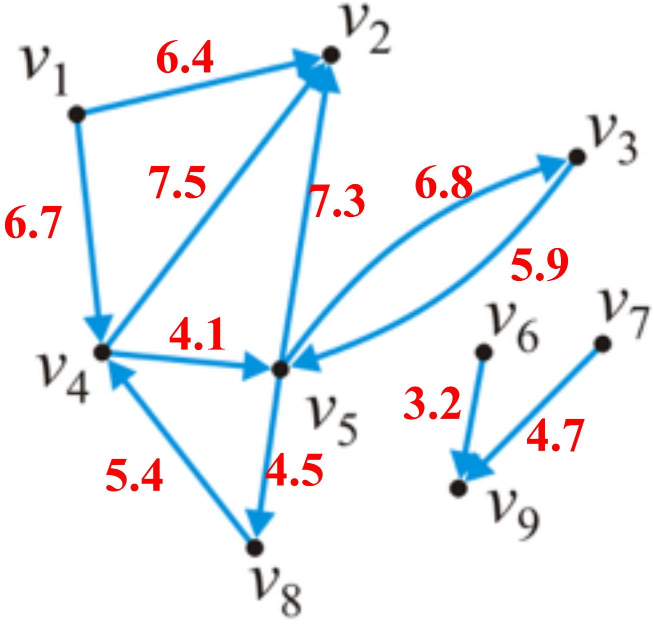
- The sub-graph $\{v_3, v_4, v_5, v_8\}$ is strongly connected
- The sub-graph $\{v_1, v_2, v_3, v_4, v_5, v_8\}$ is weakly connected



Weighted directed graphs

In a weighted directed graphs, each edge is associated with a value

Unlike weighted undirected graphs, if both (v_j, v_k) and (v_j, v_k) are edges, it is not required that they have the same weight

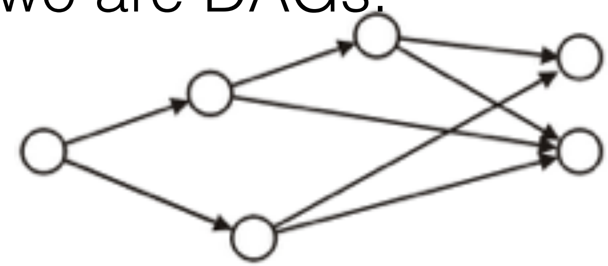


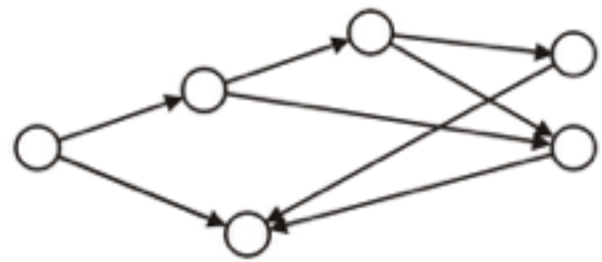
Directed acyclic graphs

A *directed acyclic graph* is a directed graph which has no cycles

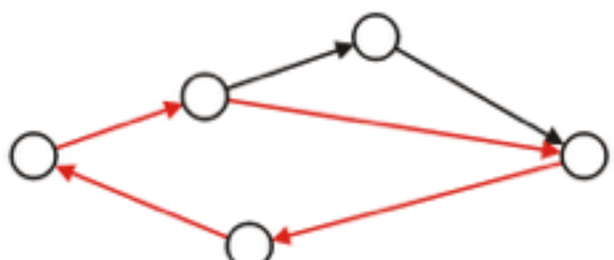
- These are commonly referred to as DAGs
- They are graphical representations of partial orders on a finite number of elements

These two are DAGs:





This directed graph is not acyclic:



Directed acyclic graphs

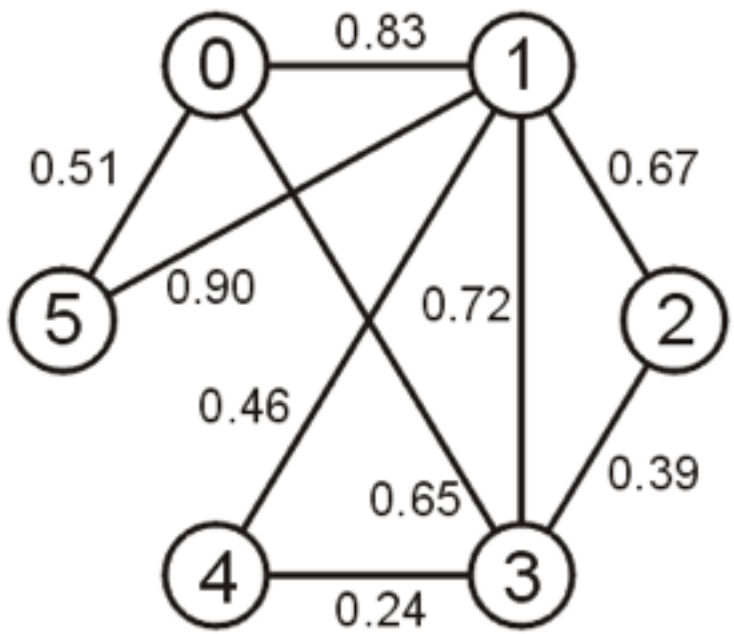
Applications of directed acyclic graphs include:

- The parse tree constructed by a compiler
- Dependency graphs between classes formed by inheritance relationships in object-oriented programming languages
- Information categorization systems, such as folders in a computer
- Directed acyclic word graph data structure to memory-efficiently store a set of strings (words)

Implementation

How do we store the graph?

- Adjacency matrix
- Adjacency list

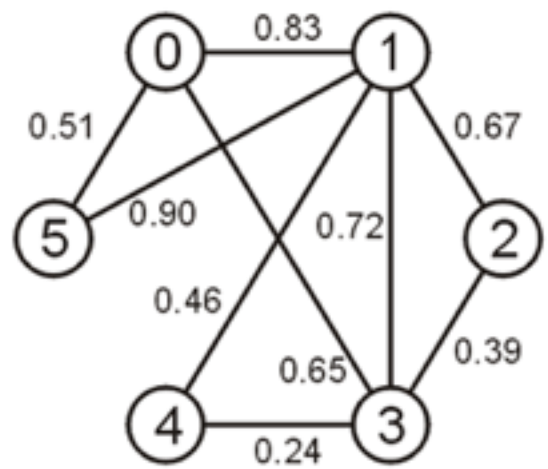


Adjacency Matrix

Define an $n \times n$ matrix $\mathbf{A} = (a_{ij})$ and if the vertices v_i and v_j are connected with weight w, then set $a_{ij} = w$ and $a_{ji} = w$

That is, the matrix is symmetric, *e.g.*,

	0	1	2	3	4	5
0		0.83		0.65		0.51
1	0.83		0.67	0.72	0.46	0.90
2		0.67		0.39		
3	0.65	0.72	0.39		0.24	
4		0.46		0.24		
5	0.51	0.90				



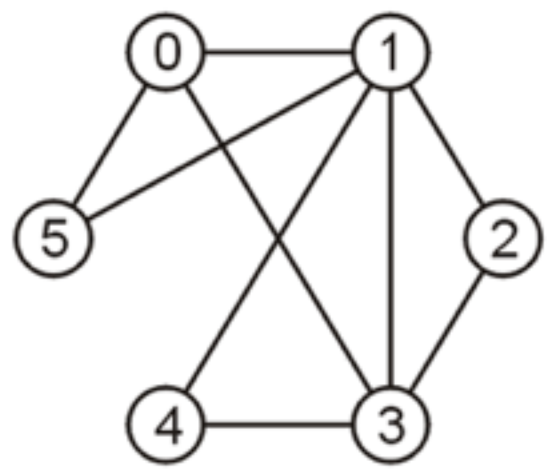
Adjacency Matrix

An unweighted graph may be saved as an array of Boolean values

– vertices v_i and v_j are connected then set

$$a_{ij} = a_{ji} = true$$

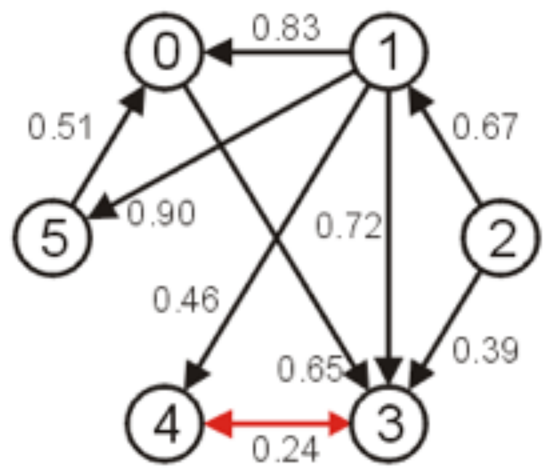
	0	1	2	3	4	5
0		T	F	T	F	T
1	T		T	T	T	T
2	F	T		T	F	F
3	T	T	T		T	F
4	F	T	F	T		F
5	T	T	F	F	F	



Adjacency Matrix

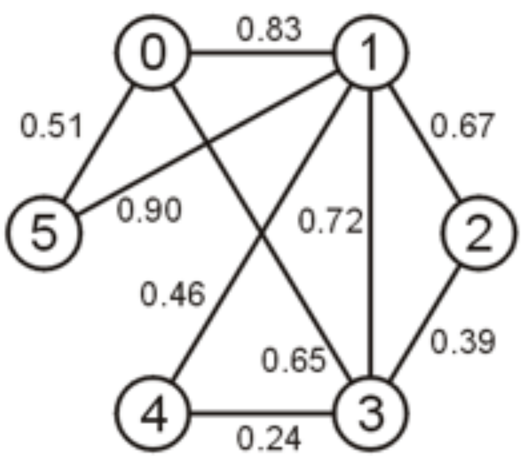
If the graph was directed, then the matrix would not necessarily be symmetric

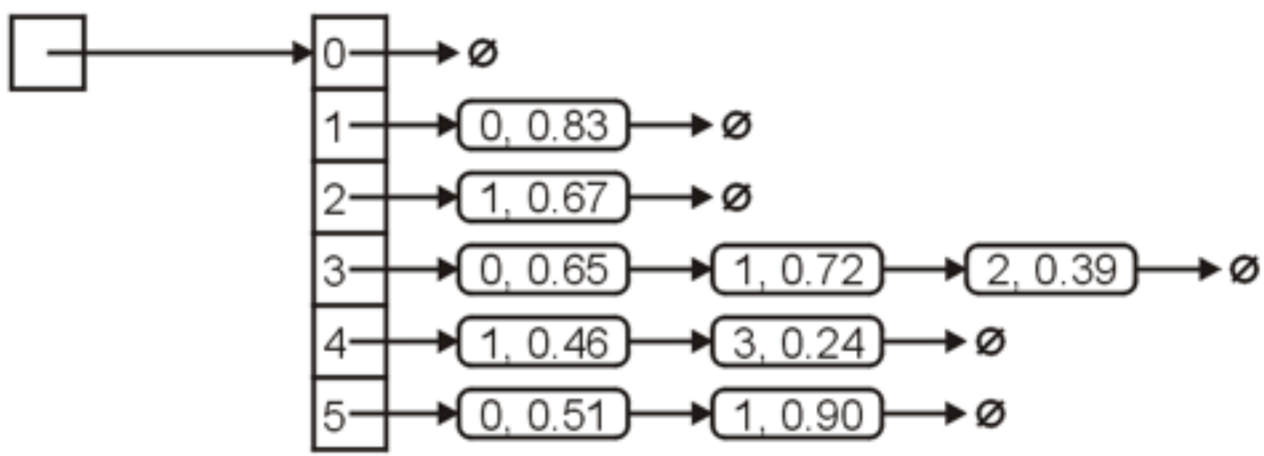
	0	1	2	3	4	5
0				0.65		
1	0.83			0.72	0.46	0.90
2		0.67		0.39		
3					0.24	
4				0.24		
5	0.51					



Adjacency List

- A space efficient implementation:
  - use an array of linked lists to store edges
- Note, however, that each node in a linked list must store two items of information:
  - the connecting vertex and the weight





References

Wikipedia, http://en.wikipedia.org/wiki/Topological_sorting

[1] Cormen, Leiserson, and Rivest, *Introduction to Algorithms*, McGraw Hill, 1990, ch11.

[2] Weiss, Data Structures and Algorithm Analysis in Java, 4rd Ed., Addison Wesley, ch 9.