COP 3502: Computer Science I Spring 2004

– Note Set 23 – Introduction to Graphs

Instructor : Mark Llewellyn
markl@cs.ucf.edu
CC1 211, 823-2790
http://www.cs.ucf.edu/courses/cop3502/spr04

School of Electrical Engineering and Computer Science
University of Central Florida

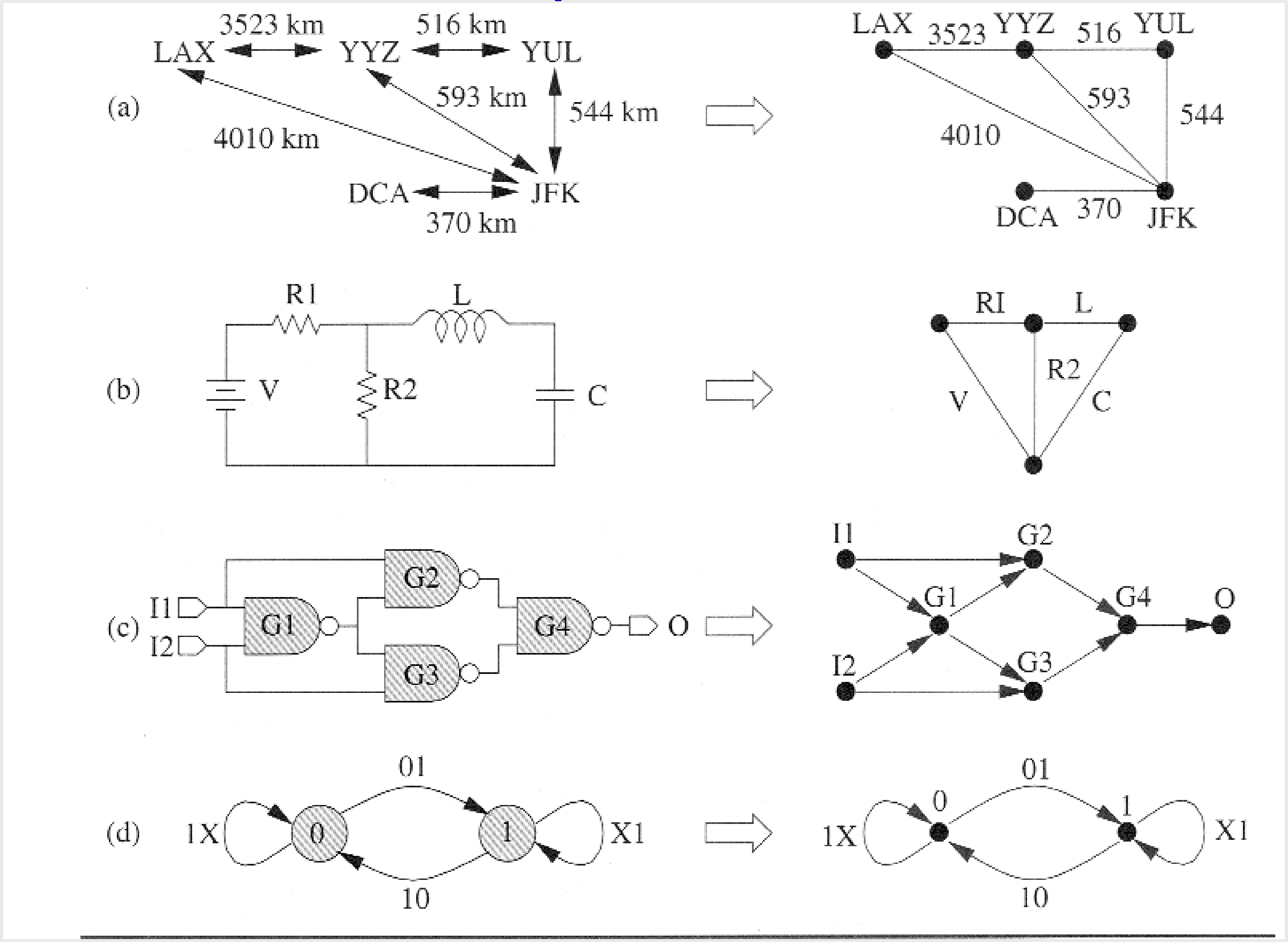
Graphs

- In spite of the flexibility of trees and the many different tree applications, trees, by their very nature, have one limitation, namely, they can only represent relationships of a hierarchical type, such as the relation between a parent and child.
- Other relationships can only be represented indirectly, such as the relationship of being a sibling. In a tree there are no links between children of the same parent, thus the sibling relationship is determined only through the parent node.
- A *graph*, which is a generalization of a tree, does not have this limitation.
- Intuitively, a graph is a collection of vertices (nodes) and the connections (edges or arcs) between them.
- Generally, there are no restrictions imposed on the number of vertices in a graph nor on the number of connections one vertex can have to other vertices.

Graphs (cont.)

- Graphs are versatile data structures that can represent a large number of different situations and events from a rather diverse group of applications.
- Graph theory has grown into a sophisticated area of mathematics and computer science in the last two hundred years since it was first studied.
- We will look only very briefly at this data structure and restrict our focus to areas of interest to computer science.
- As you will see on the next few pages, there are an enormous number of application areas to which graphs have been applied.

Graphs (cont.)


(a)
LAX
YYZ
YUL
JFK
DCA
3523 km
516 km
593 km
544 km
4010 km
370 km
3523
516
593
544
4010
370
(b)
R1
L
V
R2
C
RI
(c)
I1
I2
G1
G2
G3
G4
O
(d)
0
1
01
10
1X
X1

Graphs (cont.)

- The picture on page 4 illustrates just four areas in which graphs and graph theory have applications ranging from simple graph traversal techniques to what can be very complex abstract machines known as FSA (finite state automata, also called FSM; finite state machines – see…there's that discrete stuff again!).
- In the this picture the portion labeled (a) shows how a graph can be used to determine the shortest distance between the airports in two different cities (so you can calculate your frequent flyer miles!).
- The diagram labeled (b) illustrates a graph modeling an electrical circuit where the vertices in the graph denote where the components are connected together with the edges representing the components themselves (e.g., resistors and capacitors).
  - Using a graph you can answer questions such as "What are the mesh equations which describe the circuit's behavior?"

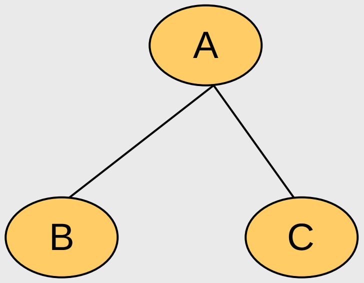
Graphs (cont.)

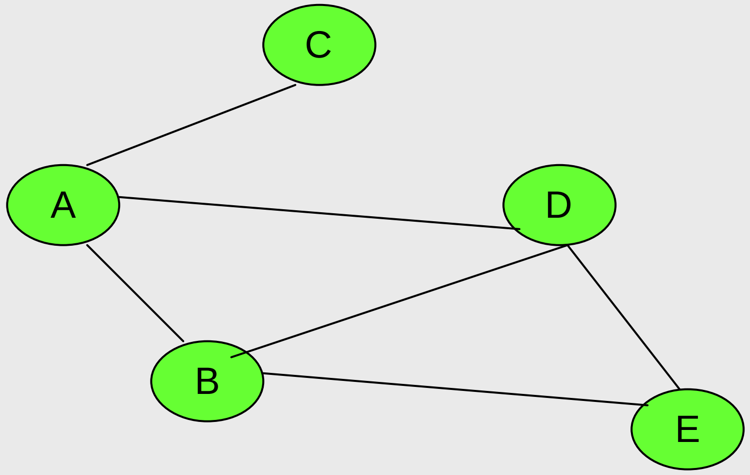
- The diagram component labeled (c) shows how a logic circuit can be reduced to a graph. In this case the logic gates are represented by the vertices, and the arrows represent the signal propagation from gate outputs to gate inputs.
  - Using a graph such as this you can answer questions of the form: “How long does it take for the signals to propagate from the inputs to the outputs?” or “Which gates are on the critical path?”
- Finally, the portion of the diagram labeled (d) represents a FSA with the vertices representing the states of the machine and the labeled arrows representing the allowable state transitions.
  - Given such a graph representation of the FSA questions such as: “Are all the states reachable?” or “Can the FSA deadlock?”

Some Graph Definitions

- A simple graph G = (V, E) consists of a non-empty set V of vertices and a possibly empty set E of edges, each edge being a set of two vertices from V. The number of vertices and edges is typically denoted |V| and |E|, respectively.

Some simple graph examples:

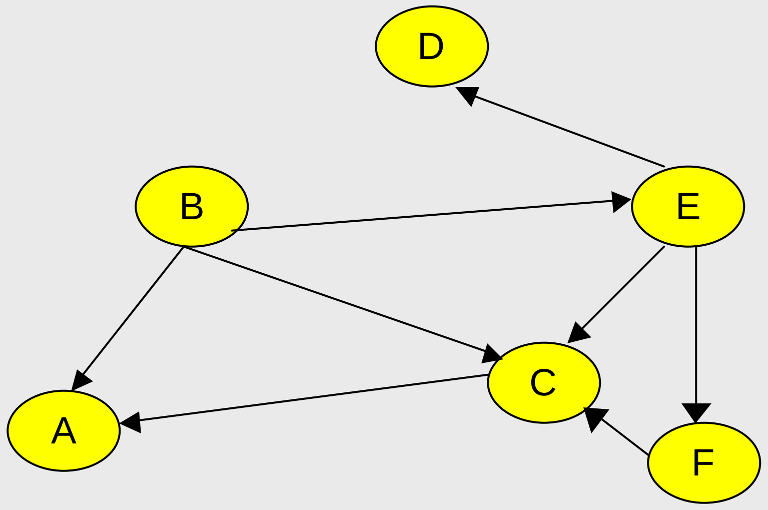


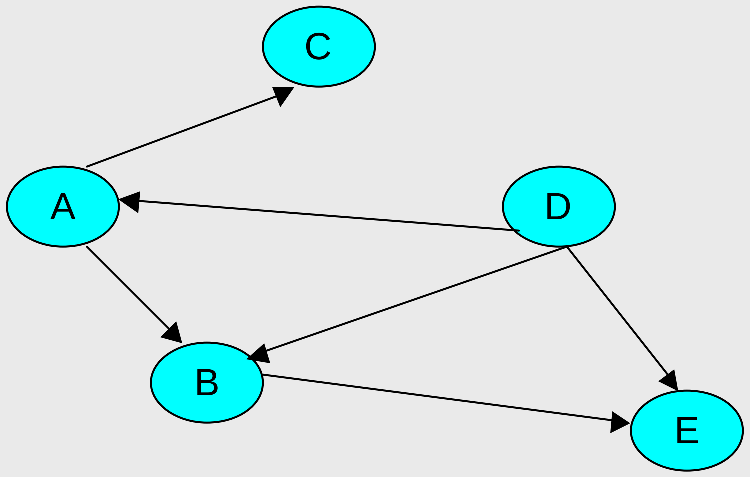


Some Graph Definitions (cont.)

- A directed graph (digraph) G = (V, E) consists of a non-empty set V of vertices and a possibly empty set E of edges (called arcs), where each edge is a pair of vertices from V. The difference is that an arc denotes a direction so that the edge $(V_i, V_j) \in E$ implies that the edge may be traversed in the direction from vertex *i* to vertex *j*. Traversal from vertex *j* to vertex *i* can occur only if the edge $(V_j, V_i) \in E$.

Some digraph examples:

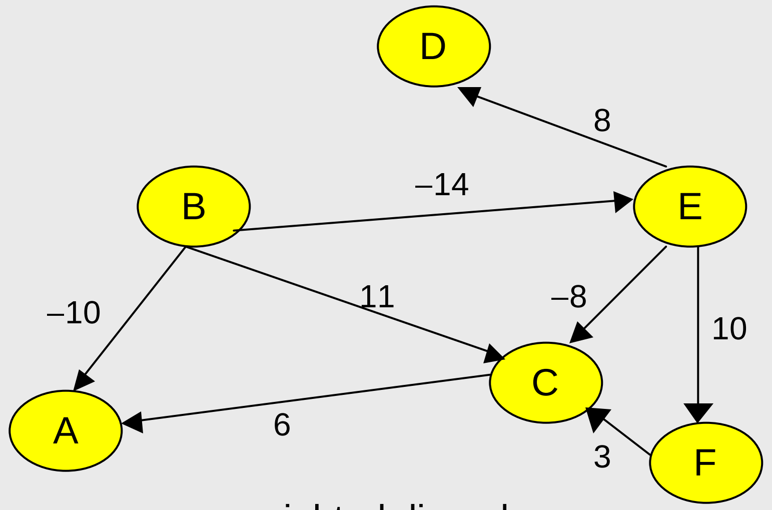




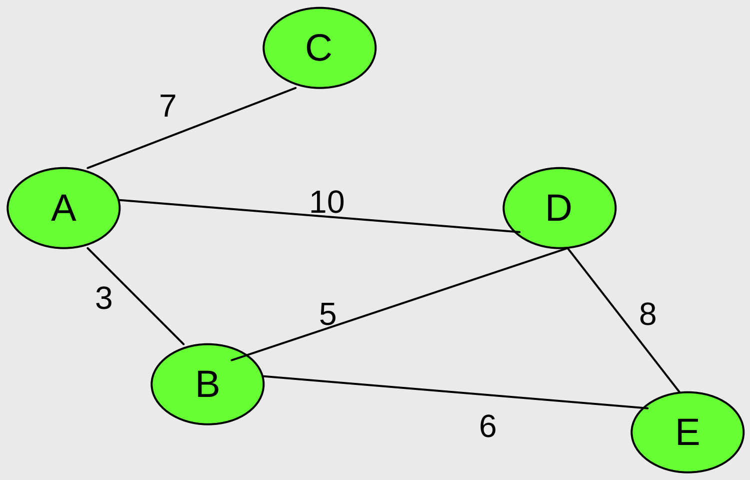
Some Graph Definitions (cont.)

- A weighted digraph is a digraph to which weights have been assigned to the edges. Weights may also be applied to the edges of an undirected graph. It is common to refer to a weighted digraph or weighted undirected graph as a network.

Examples of weighted graphs:



weighted digraph

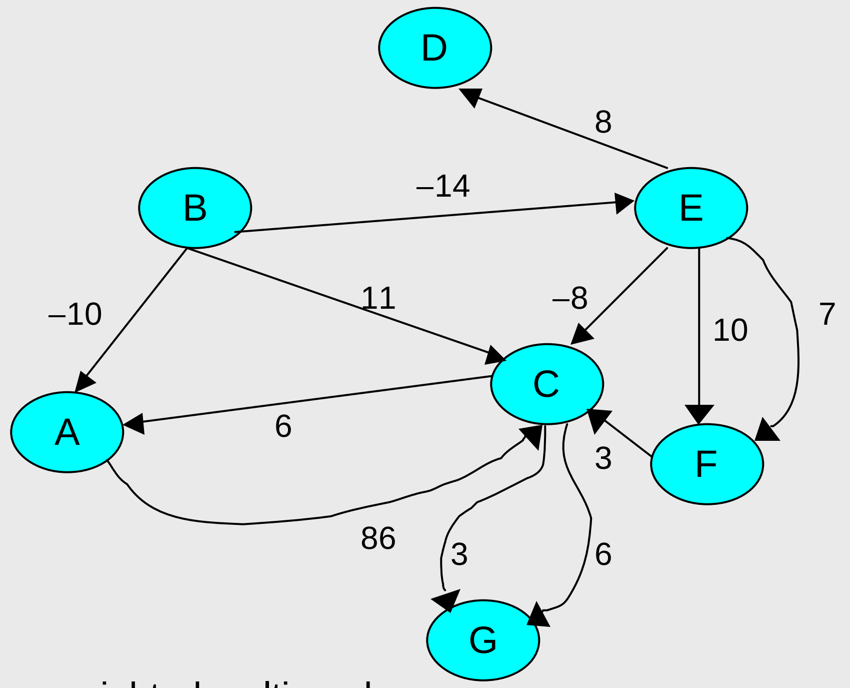


weighted graph

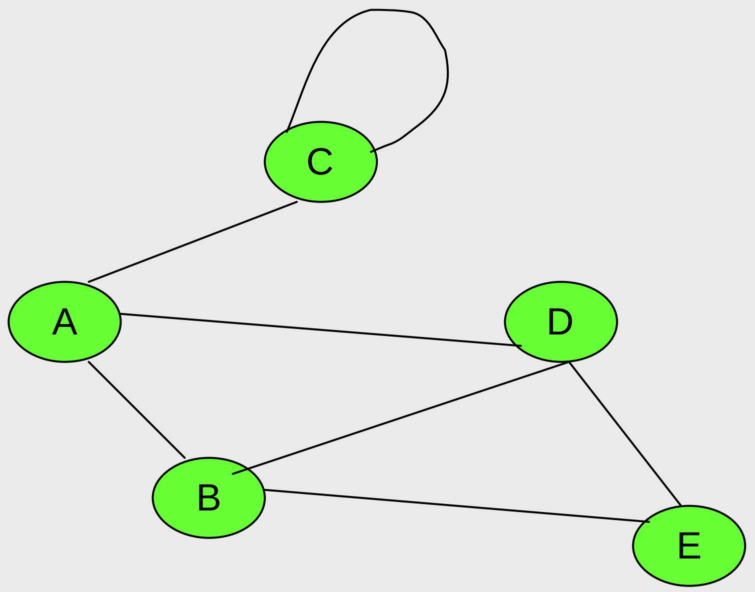
Some Graph Definitions (cont.)

- The definitions for the directed graph and the weighted digraph are too restrictive in that they do not allow for two vertices to have more than one edge between them.
  - This is analogous to saying that you can get from Orlando to Tampa but there is only one way to do it.
- More general definitions are required to ease this restriction.
- A multigraph is a graph in which two vertices can be joined by multiple edges. More formally, a multigraph G = (V, E, f) is composed of a set of vertices V, edges E, and a function f: $E \rightarrow \{\{V_i, V_j\}: V_i, V_j \in V \ \& \ V_i \neq V_j\}$.
- A pseudograph is a multigraph which does not have the restriction that an edge cannot begin and end on the same vertex. This allows cycles to be introduced into the graph which involve only a single node.

Examples of Multigraphs and Pseudographs



weighted multigraph



undirected pseudograph

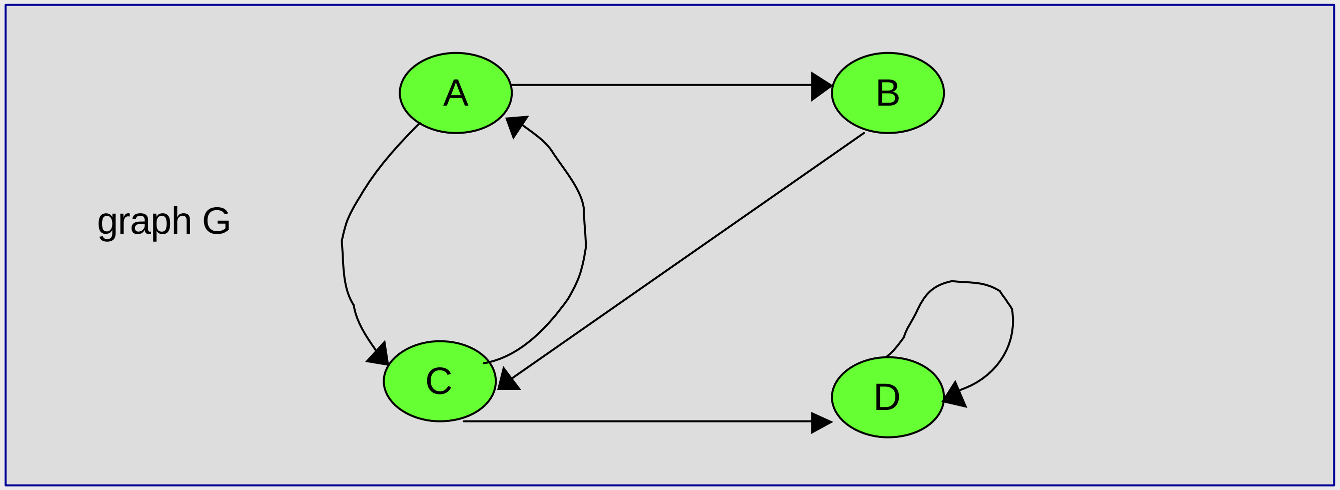
More Graph Definitions

- In an undirected graph two vertices V_1 and V_2 are adjacent if the edge $(V_1, V_2) \in E$. Such an edge is said to be incident on the vertices V_1 and V_2.
- In a directed graph the edge (V_1, V_2) is incident to vertex V_2 and incident from V_1. Being incident from is more commonly referred to as emanating from, thus the edge above emanates from V_1 and is incident on V_2.
- In a directed graph the out degree of a node is the number of edges which emanate from the node. The in degree of a node is the number of edges which are incident on the node.
- In an undirected graph the degree of a vertex V is the number of edges incident on V.
- A path in a digraph G = (V, E) is a non-empty sequence of vertices $P=\{V_1, V_2, \ldots, V_k\}$, where $V_i \in V$ for $1 \le i \le k$ such that $(V_i, V_{i+1}) \in E$ for $1 \le i \le k$. The path length of P is k-1.

More Graph Definitions (cont.)

- Given a path as defined above, vertex V_{i+1} is the successor of vertex V_i for $1 \le i \le k$. Every vertex V_i of path P (except the last vertex) has a successor.
- Given a path as defined above, vertex V_{i-1} is the predecessor of vertex V_i for $1 \le i \le k$. Every vertex V_i of path P (except the first vertex) has a predecessor.
- A path is called a simple path if and only if $V_i \neq V_j$ for all i and j such that $1 \le i < j \le k$. However, it is permissible for $V_1 = V_k$ in a simple path. If $V_1 = V_k$ the the path is a cycle (see below).
- A circuit is a path in which no edge is repeated.
- A cycle is a path of non-zero length in which the starting and ending vertex are the same. The length of the cycle is just the length of the path P. A graph containing a cycle has an infinite number of paths in the graph. A simple cycle is a path that is both a cycle and simple.
- A loop is a cycle of length 1; that is, it is a path of the form $\{V_i, V_i\}$.

More Graph Definitions – Example



Graph G is defined as: V = {A, B, C, D} and E = {(A,C), (A,B), (B,C), (C,A), (C,D), (D,D)}

The edge (A,C) is incident to C and emanates from A.

The out degree of node B is 1, the out degree of node C is 2.

The in degree of node C is 2, the in degree of node B is 1.

There is a path P = {A, B, C} in G. This is a simple path of length 2.

The path P is also a circuit.

There is a path P = {A, B, C, A} in G which is a simple cycle of length 3.

The cycle {A, C, A, C, A} has length 4, but is not a simple cycle.

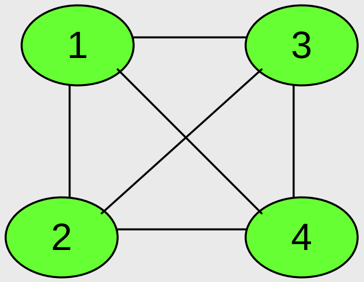
There is a loop in G consisting of (D, D).

The path {C, A, C, D} is not a simple path, C is repeated (but not at the end)

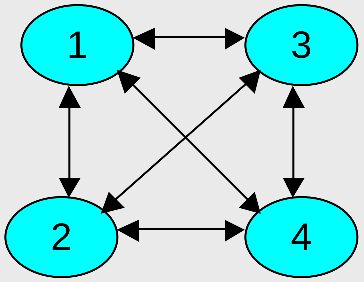
Even More Graph Definitions

- A graph of *n* vertices is complete, and is denoted K_n, if for each pair of distinct vertices there is exactly one edge connecting them. That is, each vertex can be connected to any other vertex. The number of edges is such a graph is $O(|V|)$.
- An undirected graph G = (V, E) is connected iff (if and only if) there is a path between every pair of vertices in G. A graph which is unconnected contains at least one vertex which is unreachable, i.e., the graph does not contain any paths which include the unreachable vertex.
  - Note that given a specific graph G = (V, E), it may be possible to remove edges from E while G remains connected. An edge which can be removed from E and yet G remains connected is said to be unnecessary.
- A connected undirected graph that contains no cycles is called a tree.
- A graph H is a subgraph of another graph G iff its vertex and edge sets are subsets of those of G.
- A subgraph of G that contains all of the vertices of G and is a tree is called a spanning tree of G.

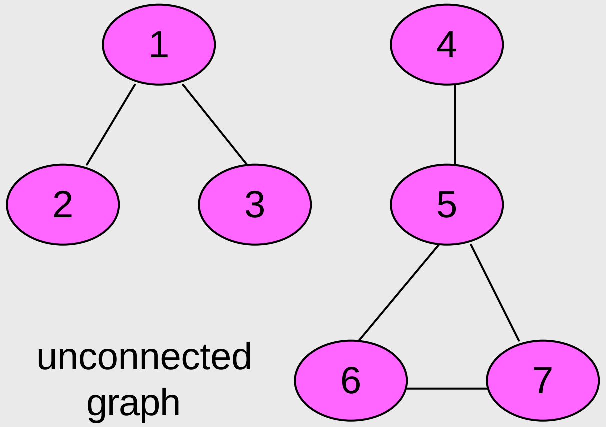
Even More Graph Definitions - Examples



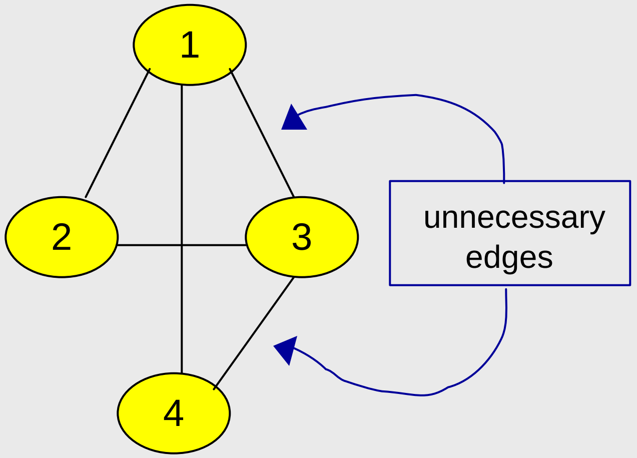
complete graph



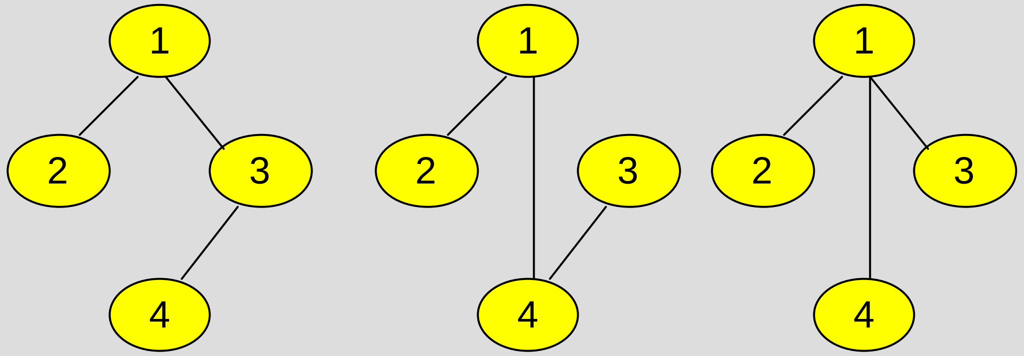
complete digraph



unconnected graph



connected graph

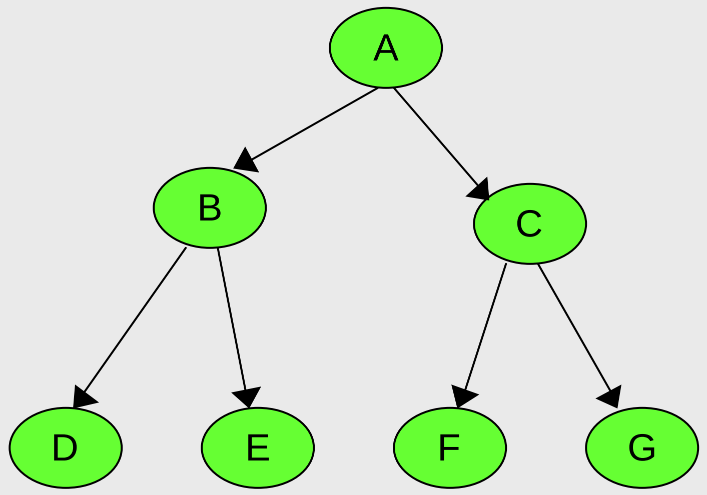


3 possible spanning trees for the connected graph shown on the right

Yet Even More Graph Definitions!

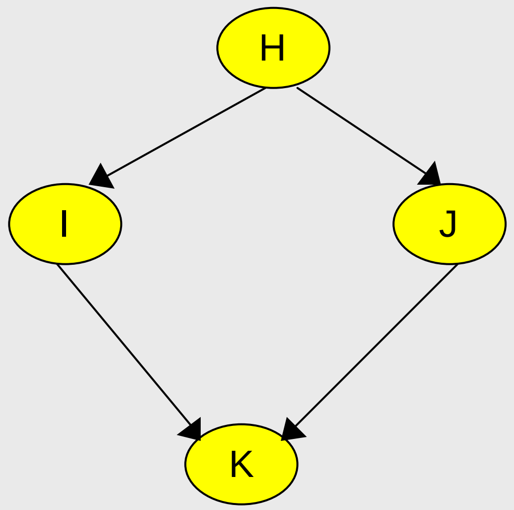
- A directed acyclic graph (DAG) is a directed graph which contains no cycles.
  - All trees are DAGs, however, not all DAGs are trees. In figure (a) on the next page we see a DAG which is clearly a tree (binary in fact), yet figure (b) represents a DAG which is clearly not a tree (node "K" has two parents and remember that all trees in Computer Science are from single parent homes).
- A directed graph which is connected is called a strongly connected graph. If a directed graph is not strongly connected but the underlying graph (its undirected counterpart) is connected, then the digraph is said to be weakly connected.

Yet Even More Graph Definitions – Examples



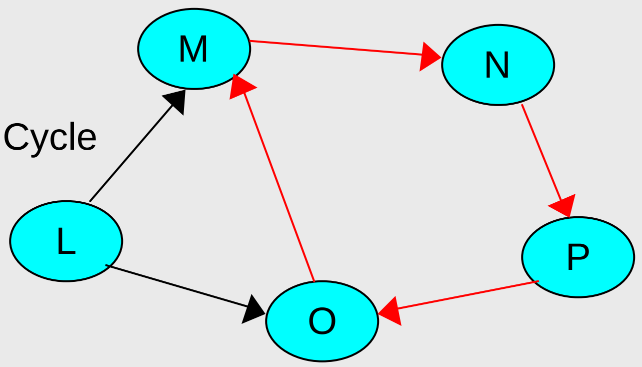
Directed Acyclic Graph (DAG)

also a binary tree



DAG but not a tree

Not a DAG. Cycle shown in red



Graph Representations

- There are a variety of ways to represent a graph and we will examine a couple of these techniques with a focus on the efficient implementation of the graph structure.
- Consider a directed graph G = (V, E). Since $E \subseteq V \times V$, graph G contains at most $|V|^2$ edges. There are many possible sets of edges for a given set of vertices V.
- Therefore, the main concern when designing a graph representation scheme is to find an efficient way to represent the set of edges.
- To do this properly depends upon whether the graph is a dense graph or a sparse graph. Informally, a graph with relatively few edges is a sparse graph while a graph with many edges is a dense graph. More formally we have:
- A sparse graph is a graph G = (V, E) in which $|E| = O(|V|)$.
  - For example, consider a graph G = (V,E) with *n* nodes. Suppose that the out-degree of each vertex in G is some fixed constant *k*. Graph G is a sparse graph because $|E| = k|V| = O(|V|)$.

Graph Representations (cont.)

- A dense graph is a graph G = (V, E) in which $|E| = \theta(|V|^2)$. For example, consider a graph G = (V,E) with *n* nodes. Suppose that the out-degree of each vertex in G is some fraction *f* of *n*, $0 < f \leq 1$. For example, if $n = 16$ and $f = 0.25$, the out-degree of each node is 4. Graph G is a dense graph because $|E| = f|V|^2 = \theta(|V|^2)$.

- For sparse graphs a simple representation technique is given by an adjacency list which specifies all vertices which are adjacent to each vertex in the graph. This list is can be implemented as a table (static implementation) in which case it is referred to as a star representation or as a linked list (the more common case). It can also be implemented as a matrix (a two-dimensional table) in which case it comes in two possible forms: an adjacency matrix or an incidence matrix.

Graph Representations (cont.)

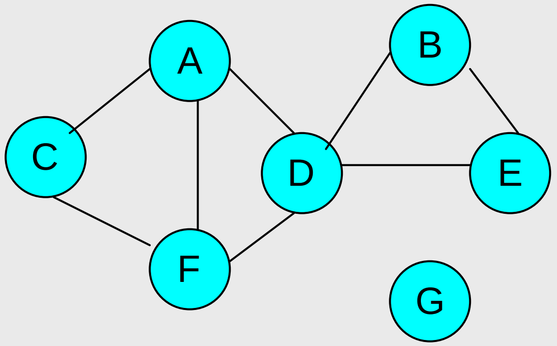
- An adjacency matrix, *A*, of graph G = (V, E) is a binary matrix: $|V| \times |V|$ such that each entry of the matrix is:

$$A_{ij} = \begin{cases} 1 & \text{if there exists an edge } (V_i, V_j) \\ 0 & \text{otherwise} \end{cases}$$

- An incidence matrix, *A*, of graph G = (V, E) is a binary matrix: $V| \times |E|$ such that each entry of the matrix is:

$$A_{ij} = \begin{cases} 1 & \text{if edge } E_j \text{ is incident with vertex } V_i \\ 0 & \text{otherwise} \end{cases}$$

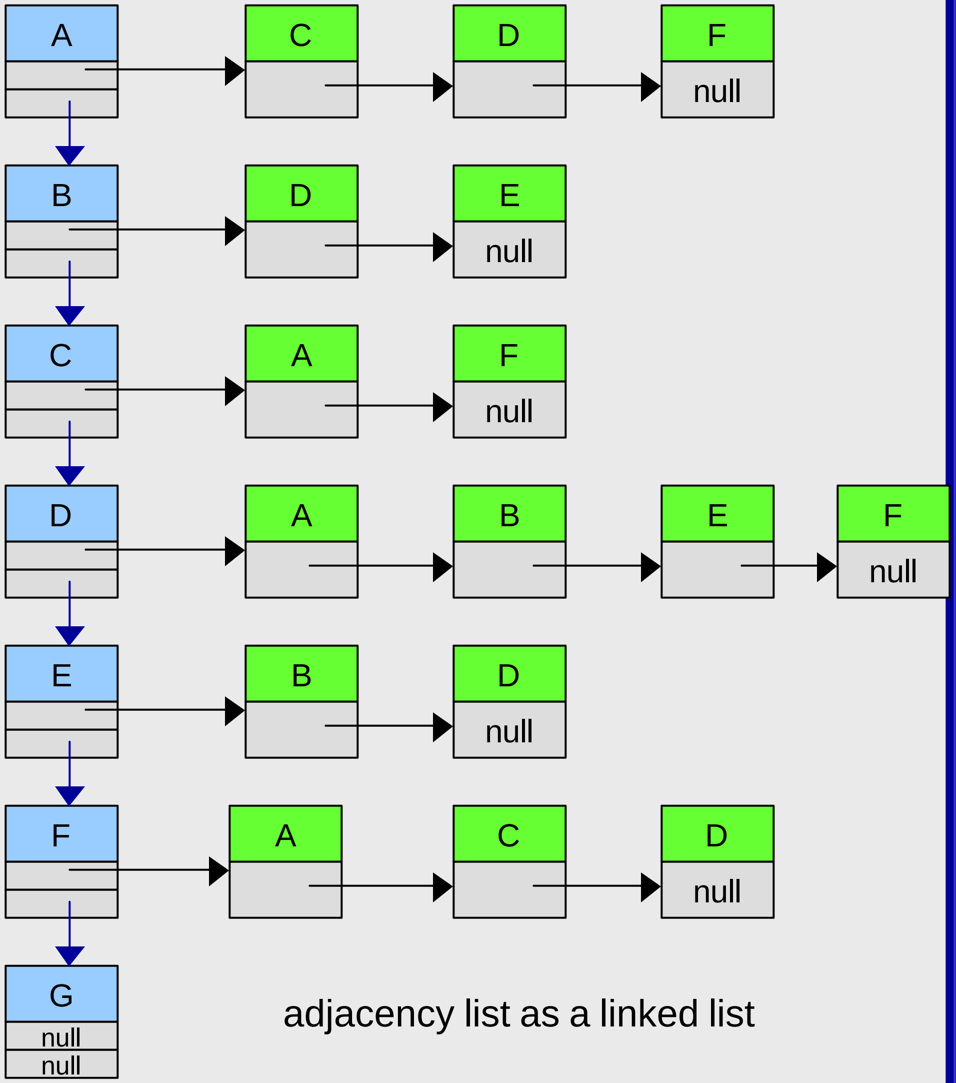
Adjacency Lists



a graph

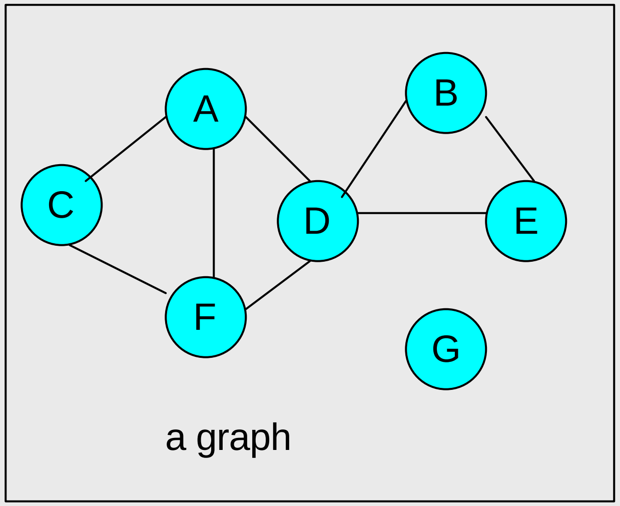
A	C	D	F	
B	D	E		
C	A	F		
D	A	B	E	F
E	B	D		
F	A	C	D	
G				

adjacency list as a table



adjacency list as a linked list

Adjacency Matrix



a graph

A value of 0 in a cell indicates that there is no edge between the row vertex and the column vertex.

1 column/vertex

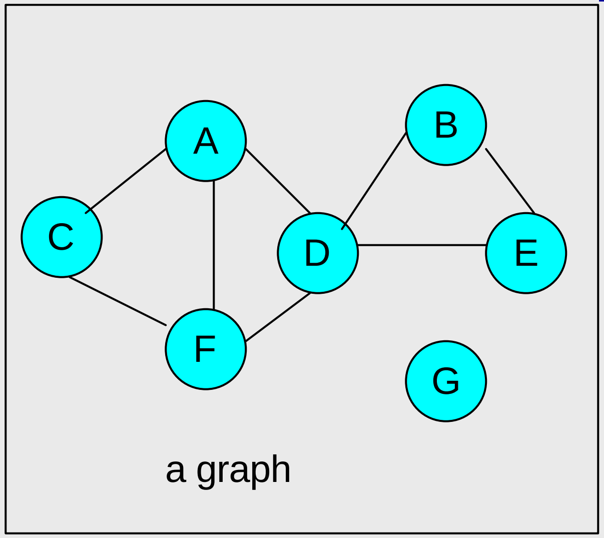
1 row/vertex

	A	B	C	D	E	F	G
A	0	0	1	1	0	1	0
B	0	0	0	1	1	0	0
C	1	0	0	0	0	1	0
D	1	1	0	0	1	1	0
E	0	1	0	1	0	0	0
F	1	0	1	1	0	0	0
G	0	0	0	0	0	0	0

adjacency matrix

A value of 1 in a cell indicates that there is an edge between the row vertex and the column vertex.

Incidence Matrix



a graph

A value of 0 in a cell indicates that the row vertex is not connected to the edge.

1 column/edge

1 row/vertex

	AC	AD	AF	BD	BE	CF	DE	DF
A	1	1	1	0	0	0	0	0
B	0	0	0	1	1	0	0	0
C	1	0	0	0	0	1	0	0
D	0	1	0	1	0	0	1	1
E	0	0	0	0	1	0	1	0
F	0	0	1	0	0	1	0	1
G	0	0	0	0	0	0	0	0

incident matrix

A value of 1 in a cell indicates that the row vertex is connected to the column edge.