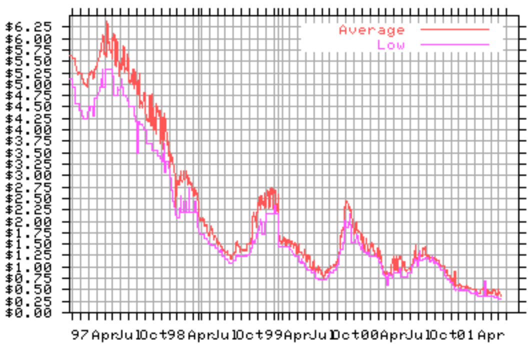
Last Course Topic: Graphs & Trees

- Motivation for Graphs
- Definition
  - Directed and undirected graphs
- Representing Graphs
- Paths, Circuits, and Trees
- Famous Graph Problems

- Covered in Chapters 9 and 10 in the text (we will cover mainly 9.1 and 10.1; you can browse the other sections)

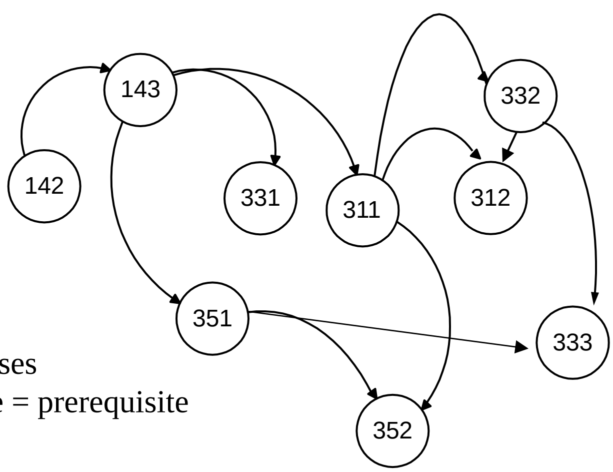
What are graphs? (Take 1)

- Yes, this is a graph….



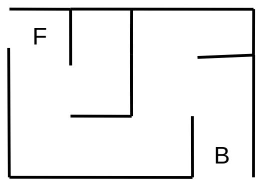
- But we are interested in a different kind of "graph"

Course Prerequisites for CSE at UW

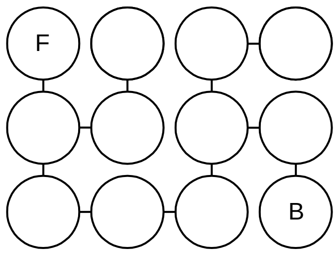


Nodes = courses
Directed edge = prerequisite

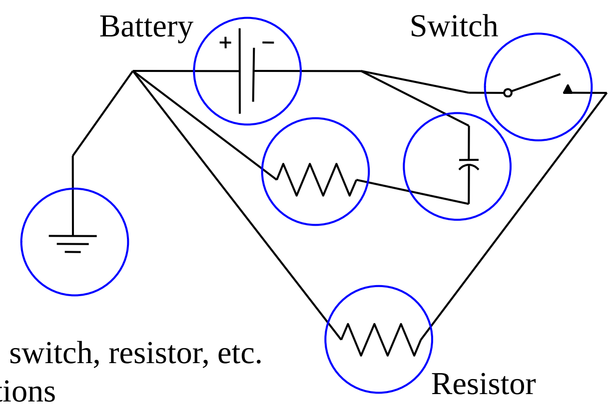
Representing a Maze or Floor Plan of a House



Nodes = rooms
Edge = door or passage



Representing Electrical Circuits

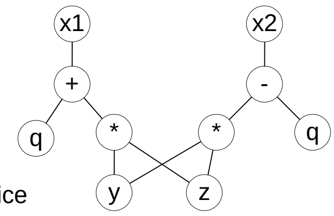


Nodes = battery, switch, resistor, etc.
Edges = connections

Representing Expressions in Compilers

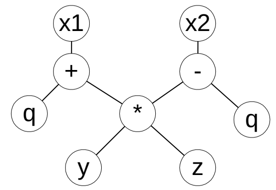
```
x1=q+y*z
x2=y*z-q
```

Naive:



y*z calculated twice

common
subexpression
eliminated:



Nodes = symbols/operators
Edges = relationships

Dependency structure of statements

S_1 $a := 0$

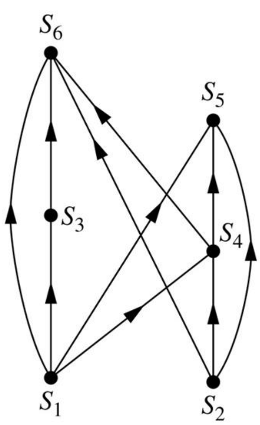
S_2 $b := 1$

S_3 $c := a + 1$

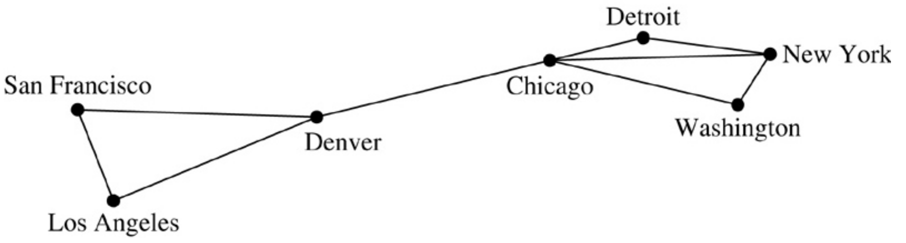
S_4 $d := b + a$

S_5 $e := d + 1$

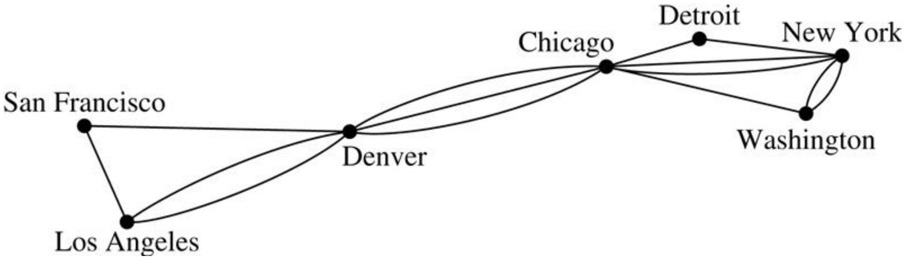
S_6 $e := c + d$



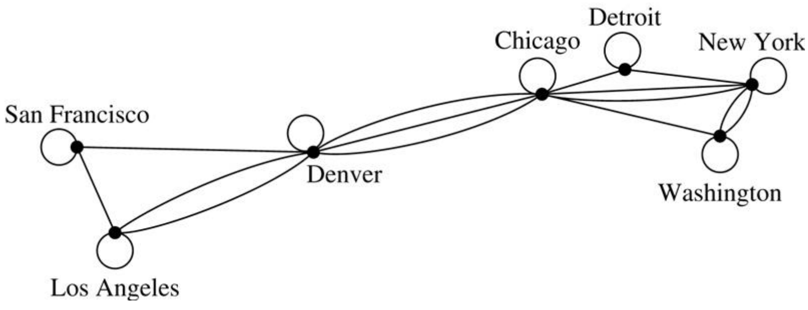
Data Centers and Connections



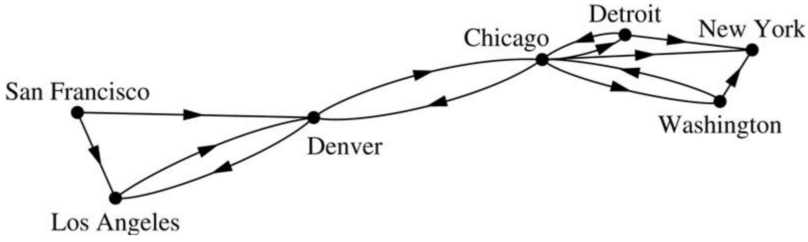
Data Centers with Multiple Connections



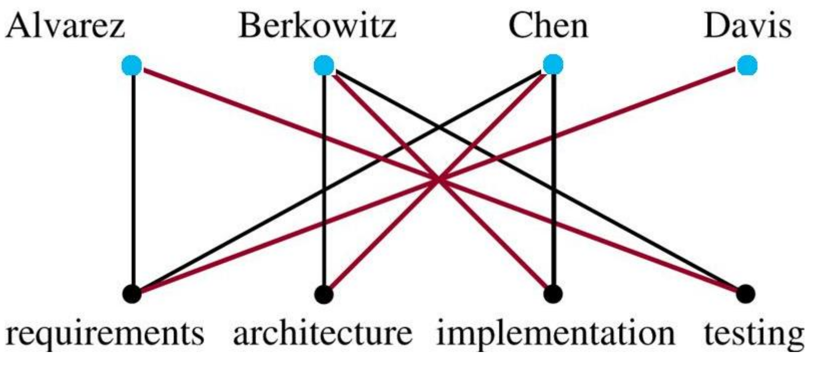
Data Centers with Diagnostic Connections



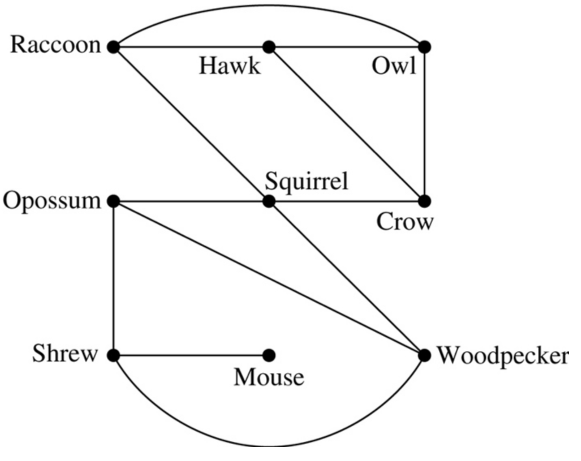
Network with One-Way Links



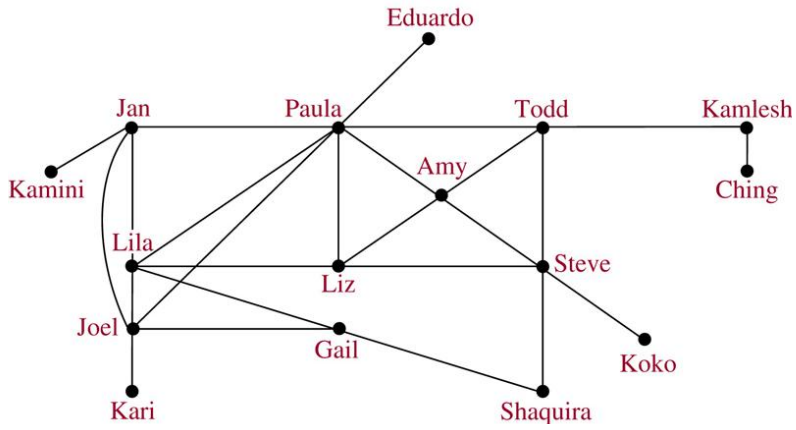
People and Tasks



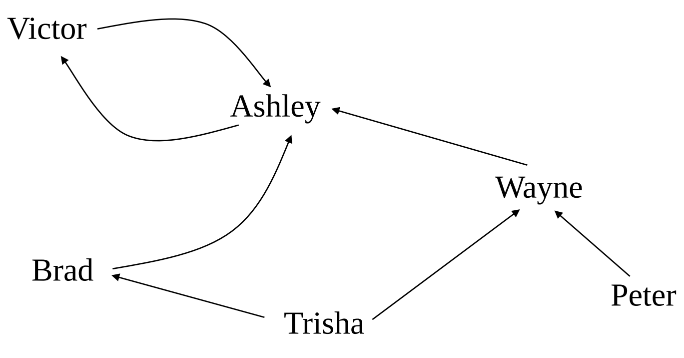
Competition between Species



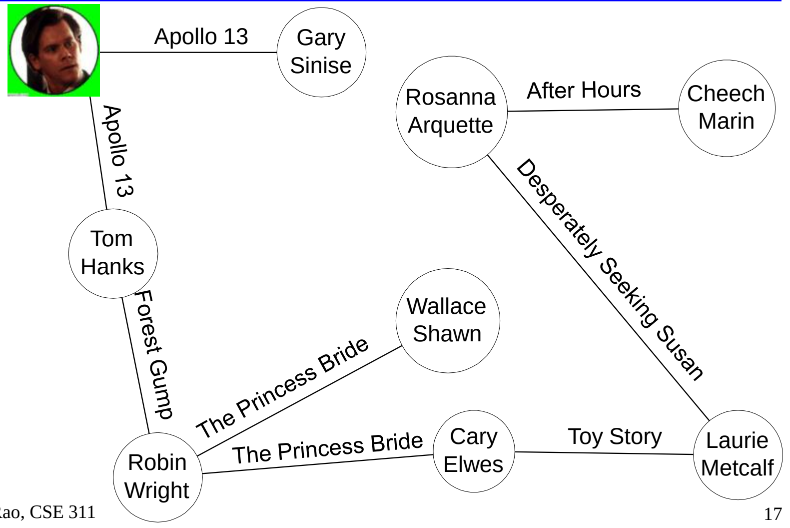
Facebook Friends



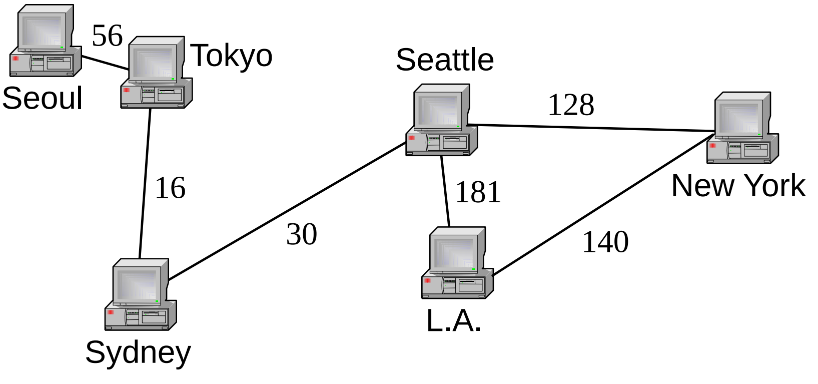
Soap Opera Relationships



Six Degrees of Separation from Kevin Bacon



Information Transmission in a Computer Network

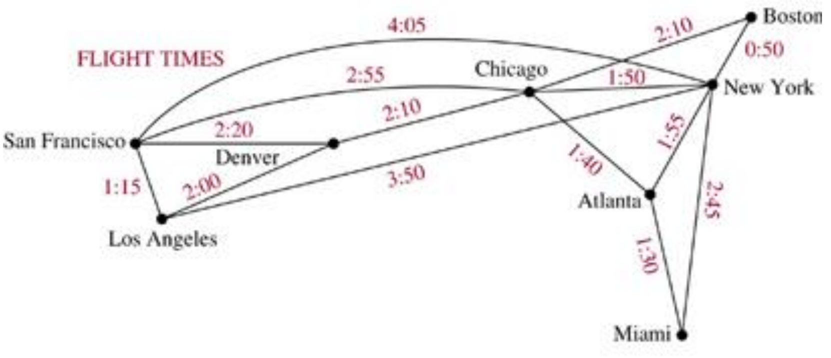


Nodes = computers
Weights on edges = transmission rates

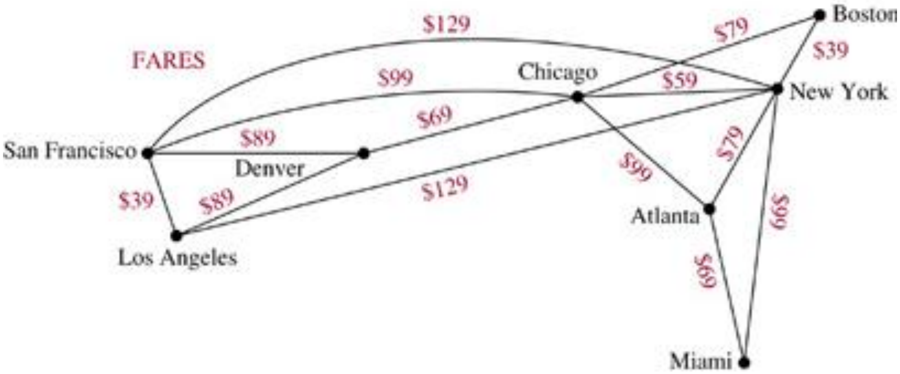
Traffic Flow on Highways

Nodes = cities
Weights on edges = # vehicles on connecting highway

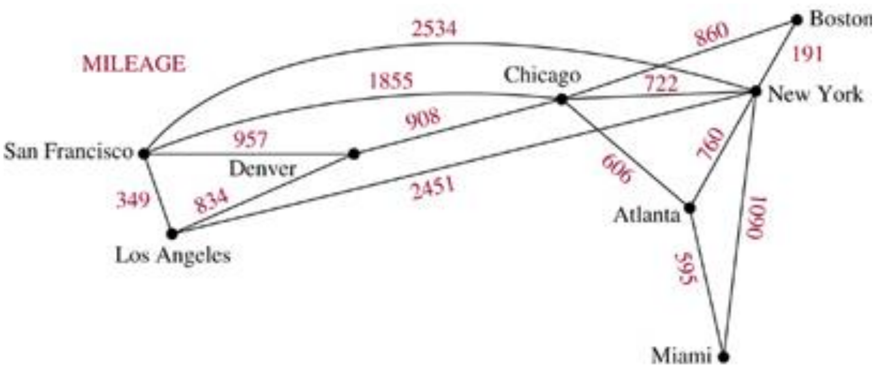
Flight times between cities



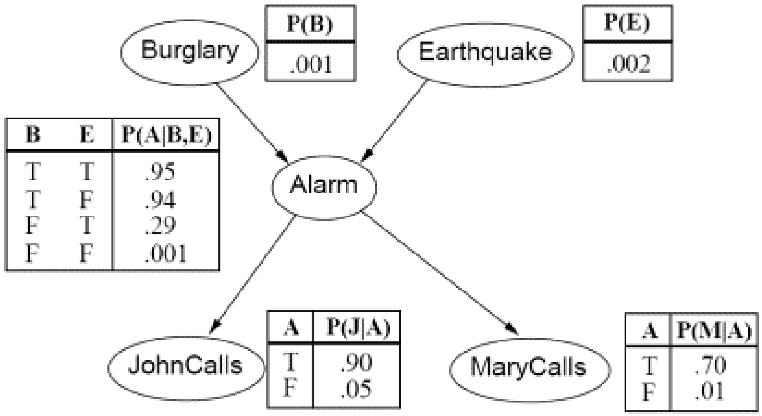
Fares between cities



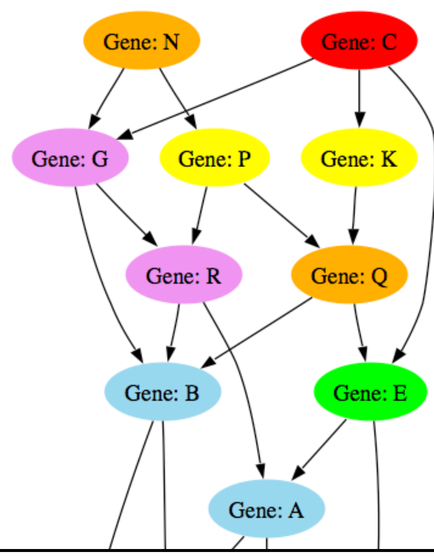
Mileage between cities



Bayesian Networks (Nodes + Edges + Probabilities)



Bayesian Network for Gene Interactions



Bayesian Network for Medical Diagnosis

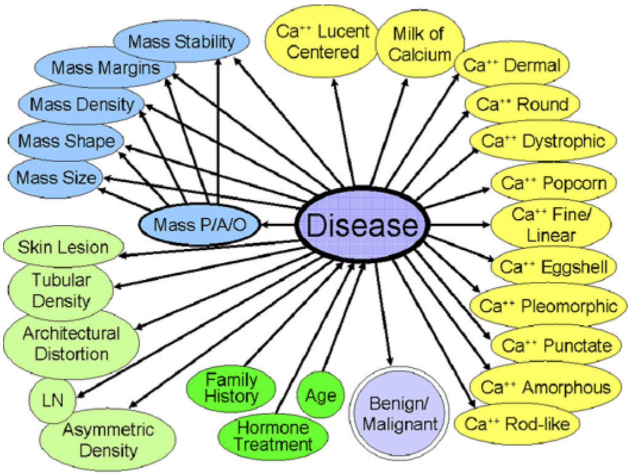
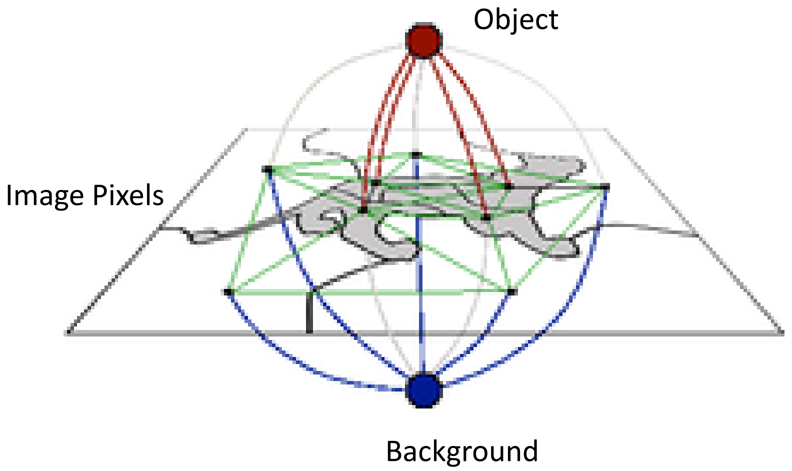


Image Analysis ("Markov Random Field")



Graphs: Definition

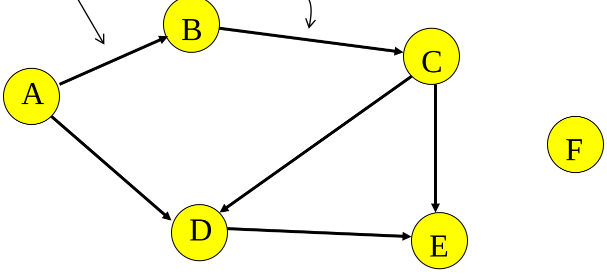
- A graph is simply a collection of nodes plus edges
  - Linked lists, trees, and heaps are all special cases of graphs
- The nodes are known as vertices (node = "vertex")
- Formal Definition: A graph $G = (V, E)$ where
  - V is a set of vertices or nodes
  - E is a set of edges that connect vertices

Graphs: An Example

- Here is a graph $G = (V, E)$
  - Each edge is a pair (v_1, v_2), where v_1, v_2 are vertices in V

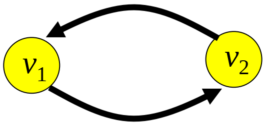
$V = \{A, B, C, D, E, F\}$

$E = \{(A,B), (A,D), (B,C), (C,D), (C,E), (D,E)\}$



Directed versus Undirected Graphs

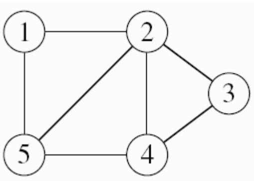
- If order of edge pairs (v_1, v_2) matters, graph is directed (also called a digraph): $(v_1, v_2) \neq (v_2, v_1)$

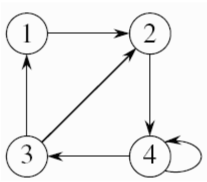


- If order of edge pairs (v_1, v_2) does not matter, graph is called an undirected graph: in this case, $(v_1, v_2) = (v_2, v_1)$ so the edge = $\{v_1, v_2\}$

Degree, In-Degree, Out-Degree

- Degree of a vertex in an undirected graph = number of edges incident on the vertex
- In-Degree/Out-degree in a digraph = number of edges entering/exiting a vertex





Deg(1) = 2
Deg(4) = 3

In-Deg(2) = 2
Out-Deg(2) = 1
In-Deg(4)=Out-Deg(4)=2

Graph Representations

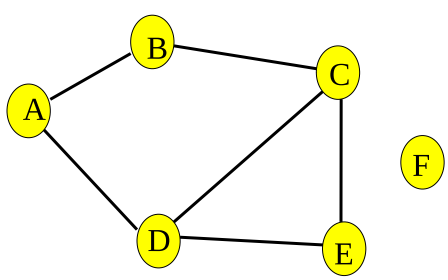
There are two ways of representing graphs:

- The *adjacency matrix* representation
- The *adjacency list* representation

Graph Representation: Adjacency Matrix

The *adjacency matrix* representation:

$$M(v, w) = \begin{cases} 1 & \text{if } (v, w) \text{ is in E} \\ 0 & \text{otherwise} \end{cases}$$

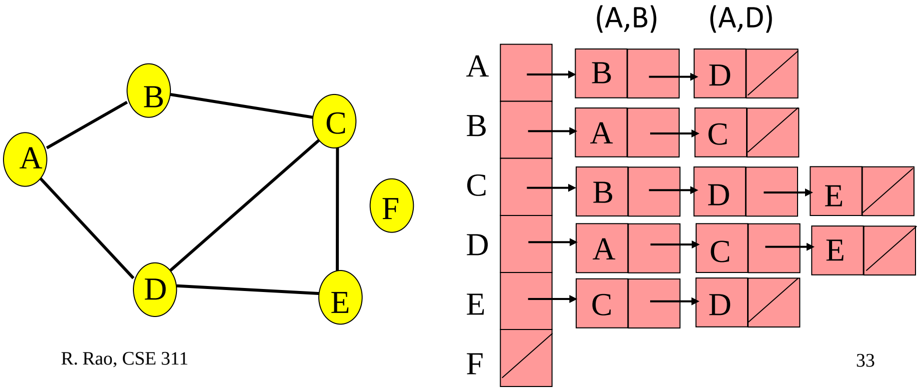


$$\begin{array}{c c} & \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \end{array} & \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

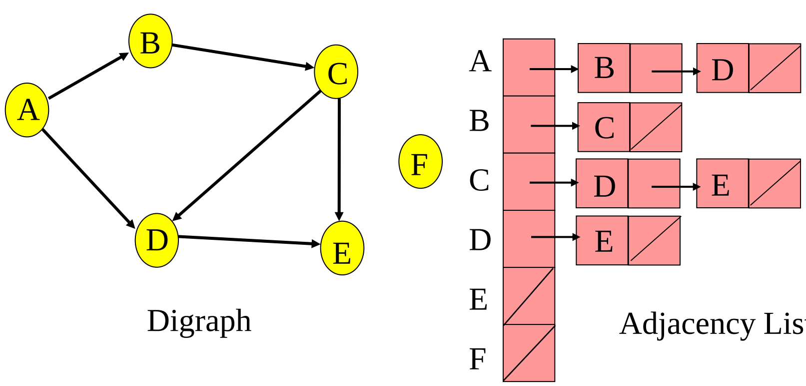
Graph Representation: Adjacency List

The *adjacency list* representation: For each v in V,

$L(v)$ = list of w such that (v, w) is in E



Adjacency List for a Digraph

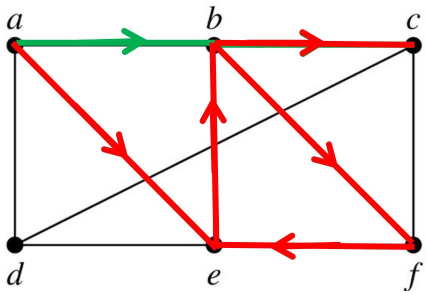


Digraph

Adjacency List

Paths in Graphs

- *Path* of length k from vertex u to vertex u' in G = (V, E) = sequence of vertices $<v_0, v_1, ..., v_k>$ where $v_0 = u$, $v_k = u'$, and $(v_{i-1}, v_i) \in E$ for i = 1, 2, ..., k.

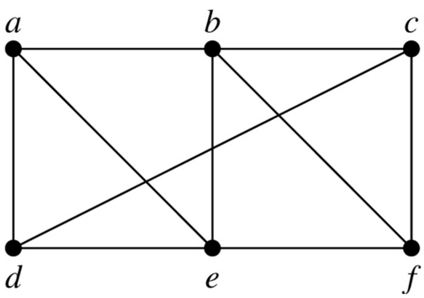


A path from a to c
<a, b, c>

Another path:
<a, e, b, f, e, b, c>

Simple Paths and Circuits

- ***Simple Path***: Path that does not repeat an edge
- ***Circuit***: Path that begins and ends at the same vertex

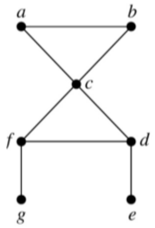


A simple path from a to c
<a, b, c>
Not a simple path:
<a, e, b, f, e, b, c>

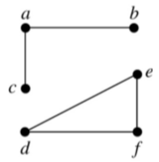
Circuit:
<a,b,c, f, b, a>
Simple circuit:
<a,b,c,f,e,a>
Simple circuit visiting all vertices:
<a,b,c,f,e,d,a>

Connected Graphs

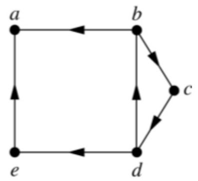
- An undirected graph is connected iff there is a path between every pair of vertices
- A directed graph is (weakly) connected iff the underlying undirected graph is connected



Connected



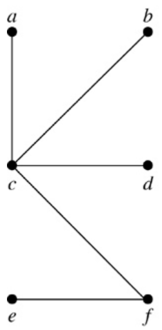
Not connected



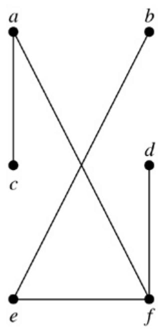
Connected

Trees

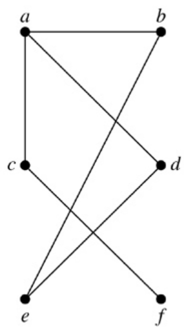
- A tree is a connected graph with no circuits



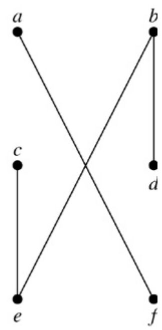
Tree



Tree

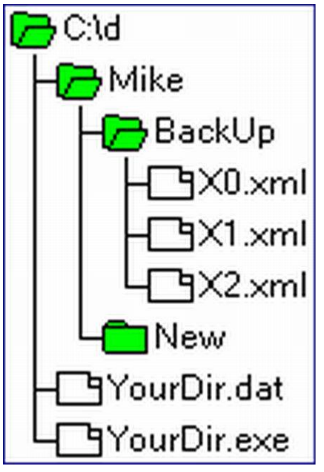


Not a tree

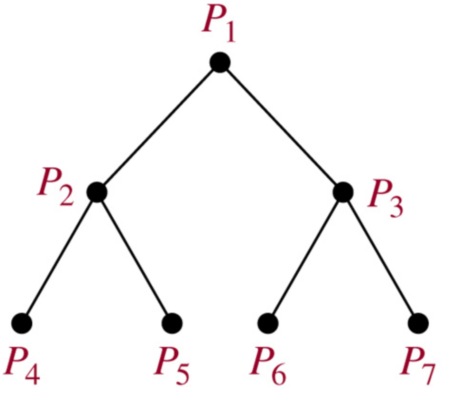


Not a tree

Examples of Trees: Folders and file system

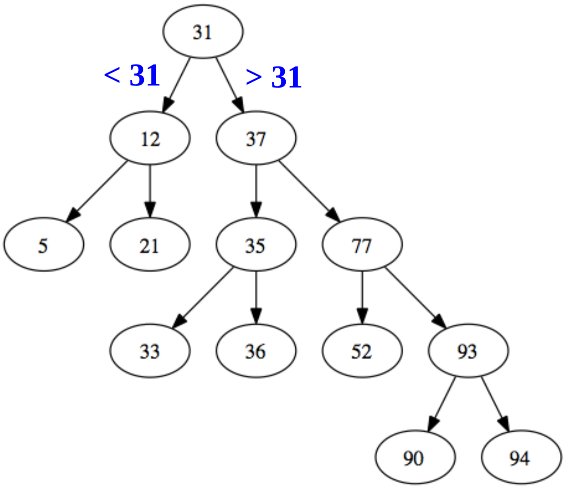


Example: Connecting Multi-Processors



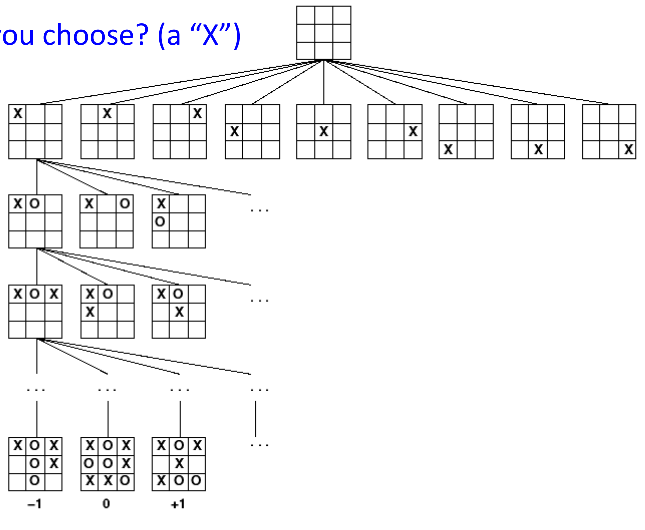
E.g., Multiplying 8 large numbers in 3 steps

Binary Search Trees

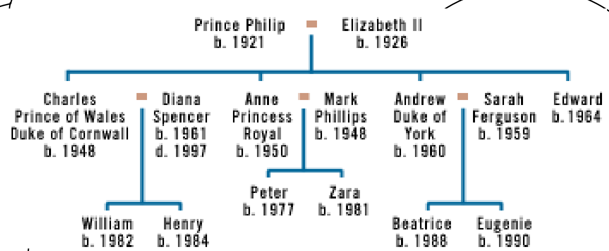


Game Trees

Which move do you choose? (a "X")

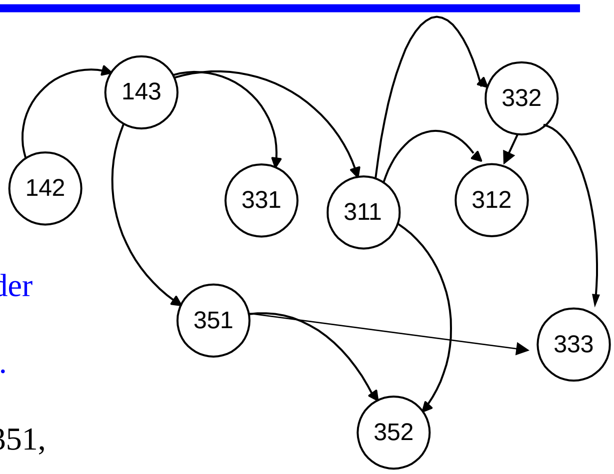


Pray tell, how does a prince represent his royal family tree?



Famous Graph Problems: Topological Sort

Graph of course prerequisites



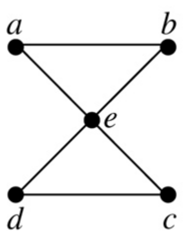
Problem: Find an order in which all these courses can be taken.

Example: 142, 143, 331, 311, 312, 332, 351, 352, 333

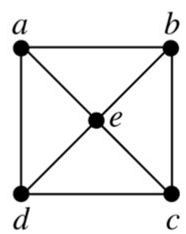
To take a course, all its prerequisites must be taken first

Famous Graph Problems: Euler Circuits

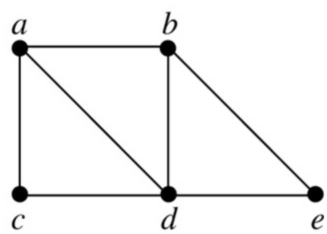
- Find a circuit going through *every edge* exactly once.



abedcea



No Euler circuit



No Euler circuit

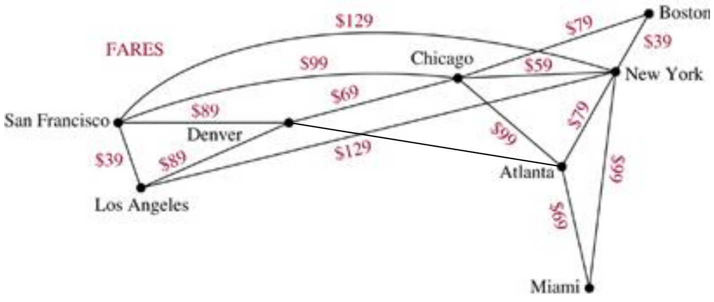
(An Euler path exists though)

Theorem: For Euler circuit to exist, every vertex must have even degree (Why? If entering vertex, must exit)

⇒ ***Fast algorithm for checking if Euler circuit exists***

Hamiltonian Circuit Problem

✦ Find a circuit passing through *every vertex* exactly once.



San Francisco, Chicago, Boston, New York, Miami, Atlanta, Denver, LA, San Francisco

Hamiltonian Circuit Problem

✦ Find a circuit in G = (V,E) passing through *every vertex* exactly once.

✦ Naïve algorithm:
- ➪ Try all permutations of the vertices
- ➪ Check to see if any permutation is a valid Hamiltonian circuit in the graph.

✦ There are |V|! permutations ⇒ running time is > exponential in size of input.

Can show this is an "NP-Complete" Problem:
Fast algorithm unlikely to exist!!
(More on this in CSE 312)

Next Class: Final Review!