Trees
(some material from Rosen, 7th edition)

- Trees were introduced by Arthur Cayley in 1857 to represent saturated hydrocarbons (isomers of chemical formula: $C_n H_{2n+2}$)

- Applications of trees in computer science
  - Search trees
  - Game trees
  - Decision trees
  - Huffman codes

Butane / isobutane

- Butane / isobutane are isomers with the same chemical formula, C_4H_{10}, but different properties
- The English mathematician Arthur Cayley uses a tree graph to illustrate how they differ

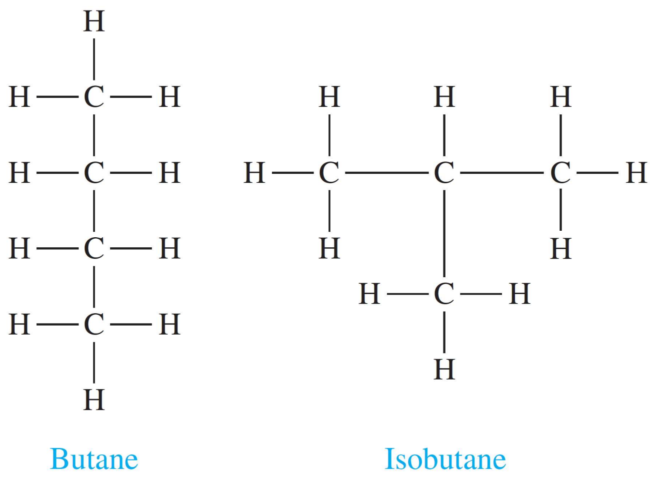


FIGURE 9 **The Two Isomers of Butane.**

A family tree

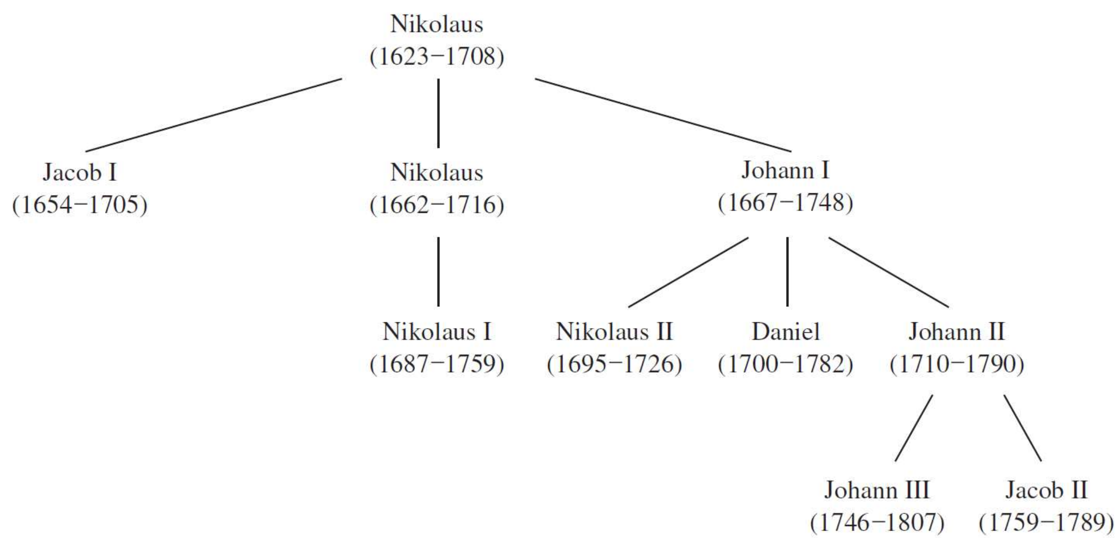


FIGURE 1 The Bernoulli Family of Mathematicians.

745

Computer file system

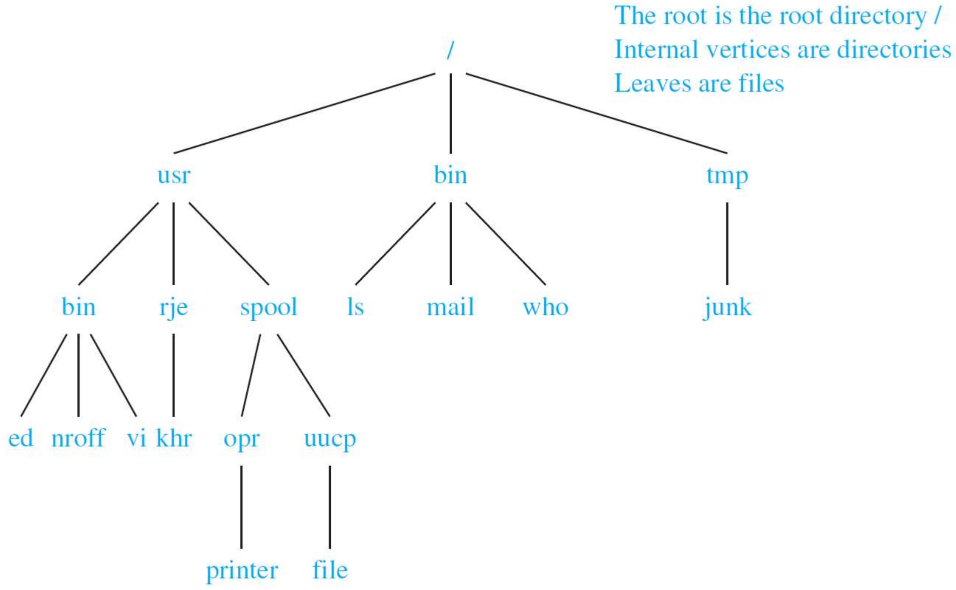


FIGURE 11 **A Computer File System.**

Representation of organizations

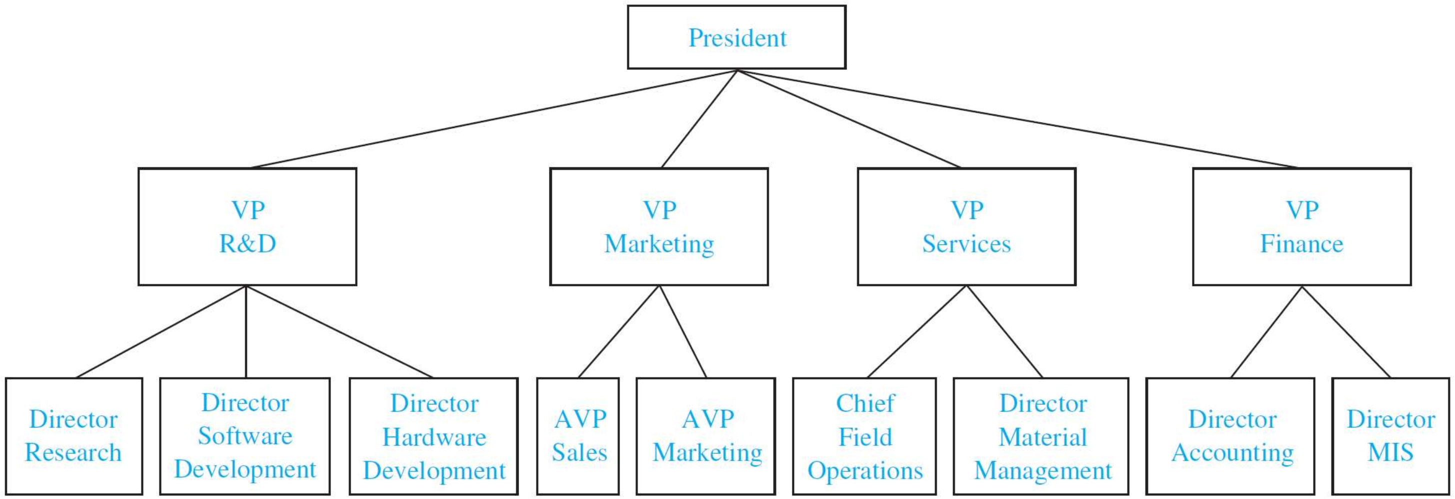
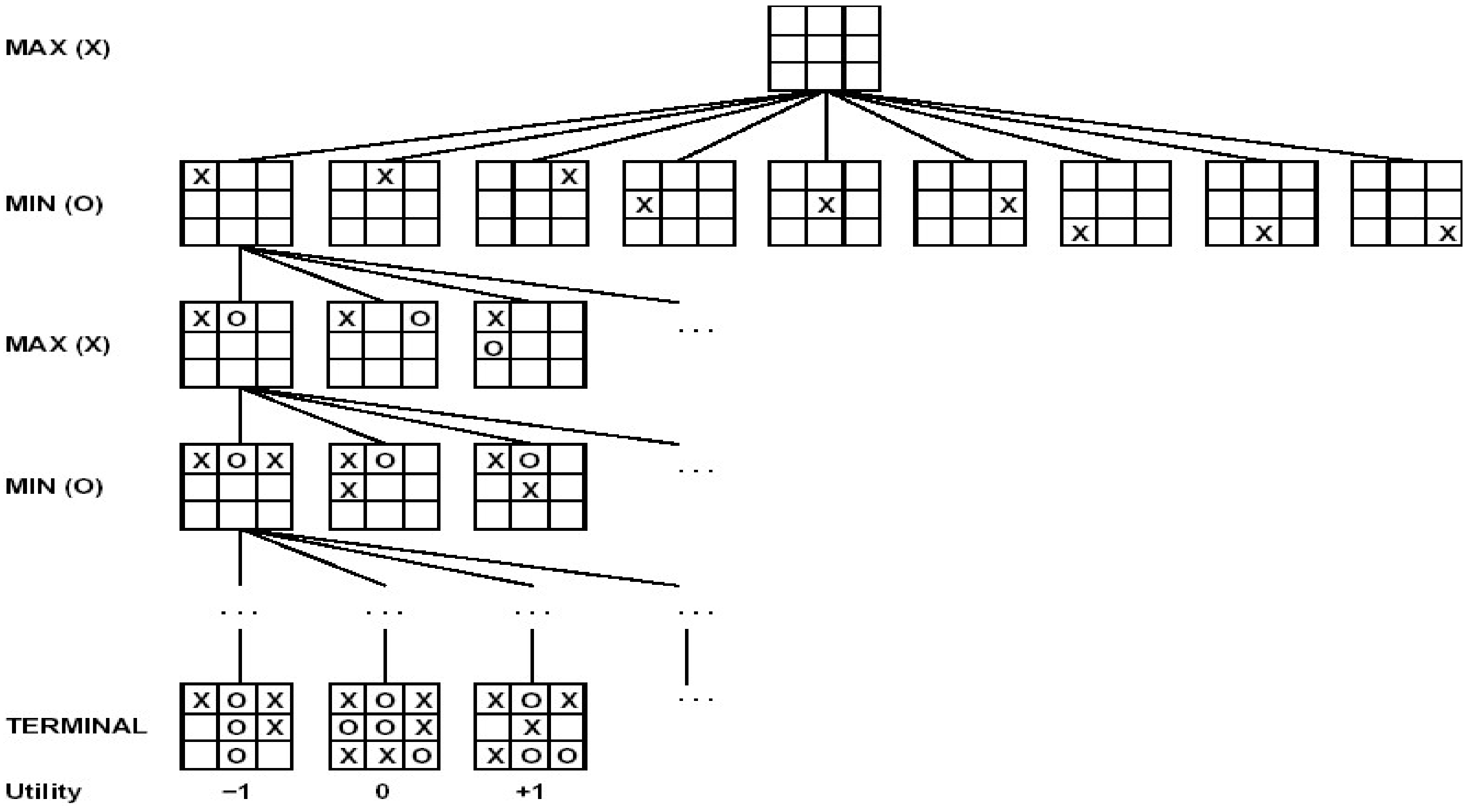


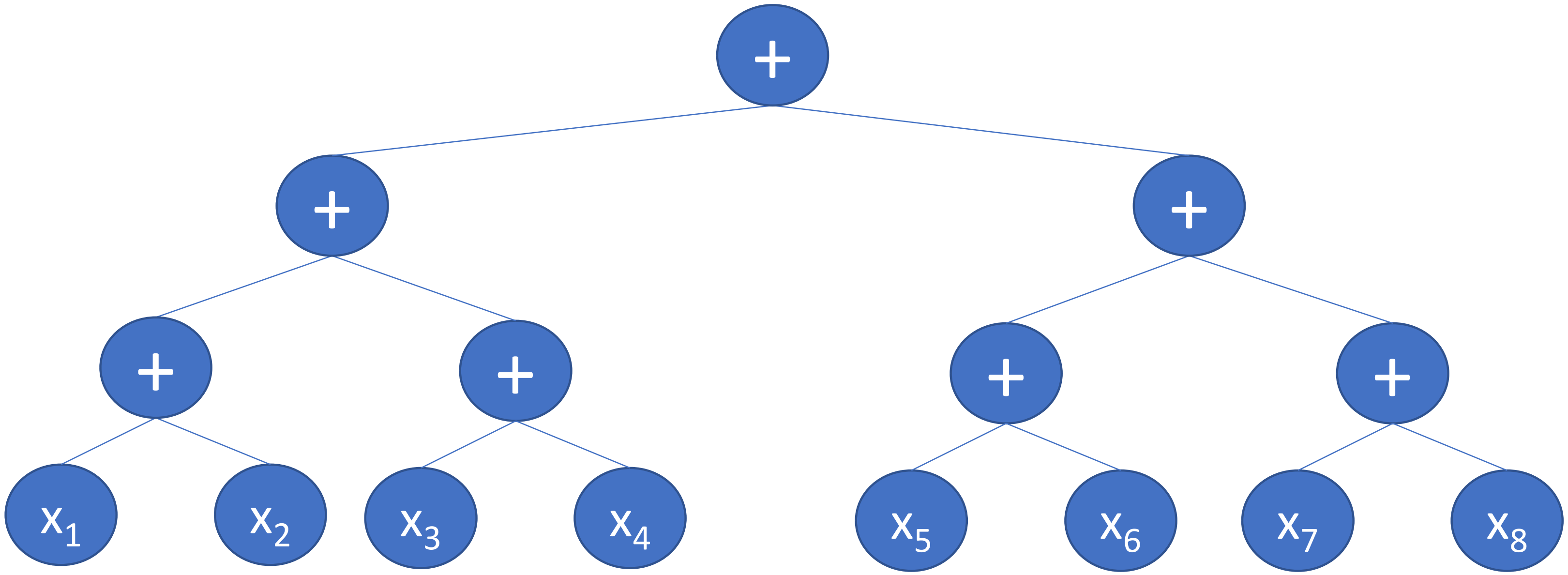
FIGURE 10 **An Organizational Tree for a Computer Company.**

Game trees: Tic-tac-toe

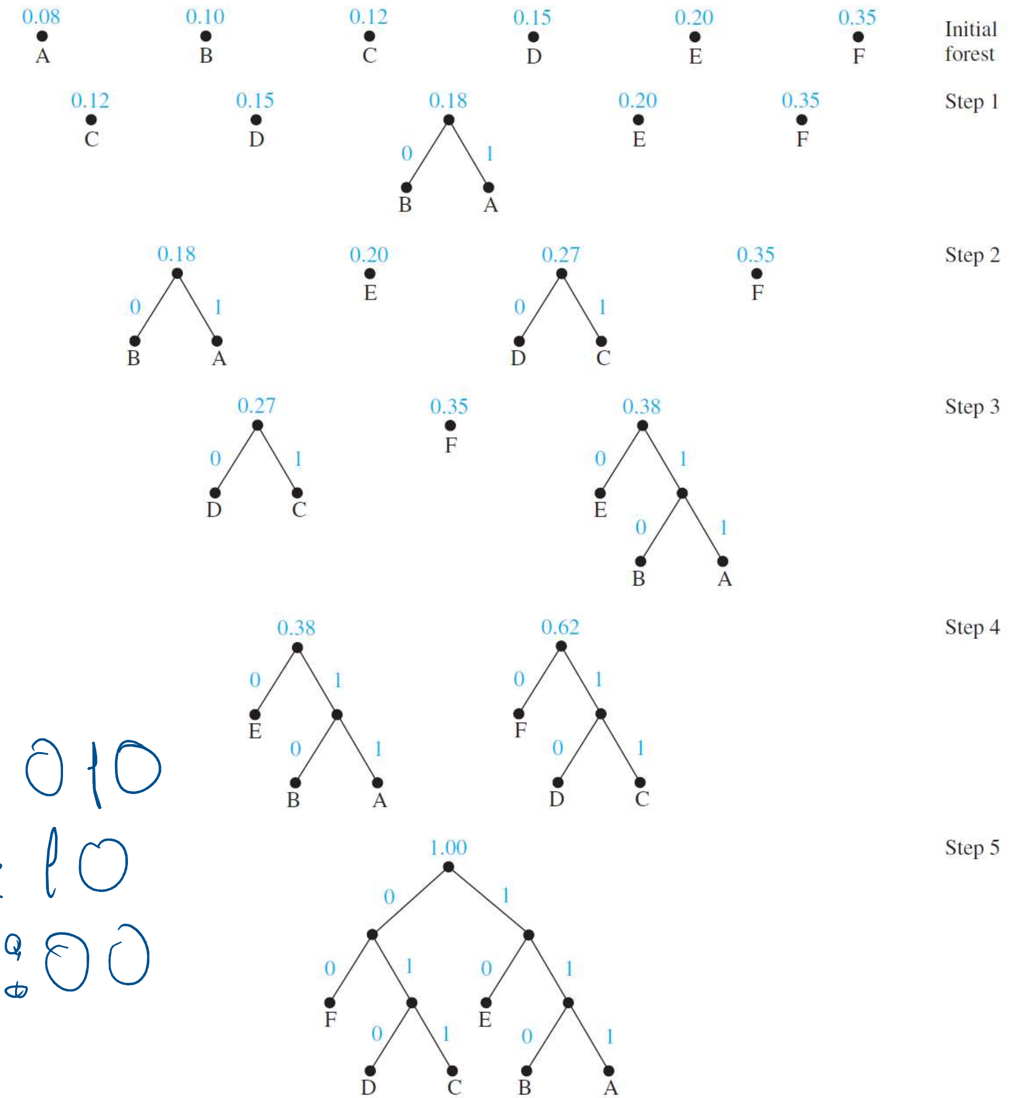


Parallel processing

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8$$
$$= \big((x_1 + x_2) + (x_3 + x_4)\big) + \big((x_5 + x_6) + (x_7 + x_8)\big)$$



Huffman coding



A: 111 D: 010
B: 110 E: 10
C: 011 F: 00

Tree: Definition

A tree is a connected undirected graph with no simple circuits.

- Example: Which of the following graphs are trees?

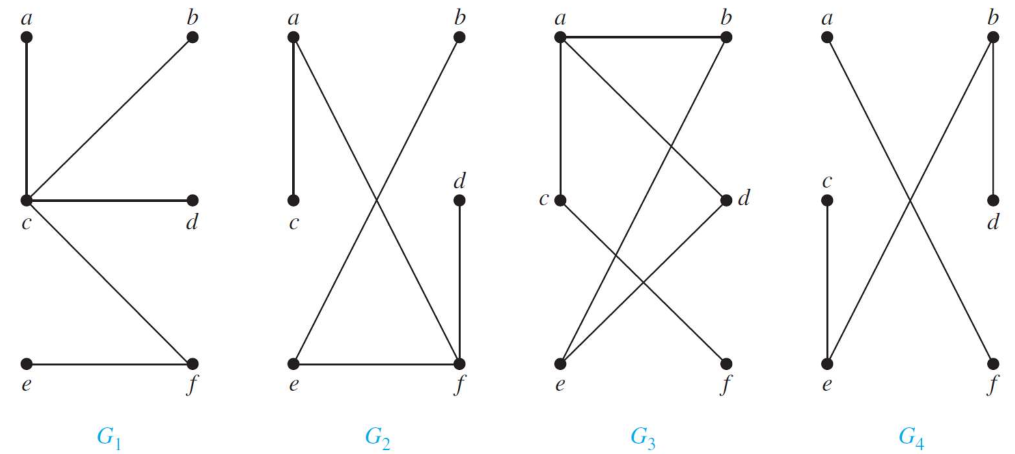


FIGURE 2 **Examples of Trees and Graphs That Are Not Trees.**

Forest: Definition

A forest is a disconnected undirected graph with no simple circuits.

- Each of the strongly connected components of a forest is a tree

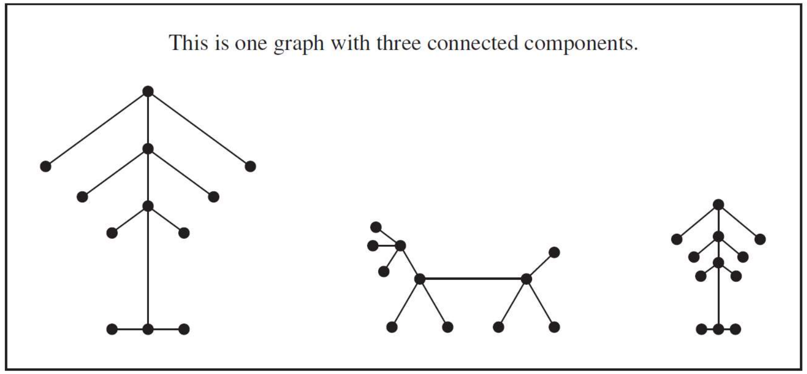


FIGURE 3 **Example of a Forest.**

THEOREM 1 An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices.

- Proof [assume that T is a tree]
  - A tree is a connected graph. Therefore there must exist a trajectory between any pair of vertices.
  - The path has to be unique. If there were two distinct paths between two vertices, the paths could be joined to form a simple circuit.
- Proof [assume that there is a unique trajectory between any two vertices]:
  - T is connected, because there is a path between any two of its vertices.
  - T cannot have simple circuits. If T had a simple circuit that contained the vertices x and y, then there would be two simple paths between x and y: A simple path from x to y and a second, distinct, simple path from y to x.

  Hence, a graph with a unique simple path between any two vertices is a tree.

THEOREM 2 A tree with n vertices has $n - 1$ edges.

- Proof [by induction]
  - [base case] A tree with n = 1 vertex has 0 = n-1 edges.
  - [inductive case]

    Assume that all possible trees with n vertices have (n-1) edges.
    - Suppose that a tree T has n + 1 vertices
    - Let v be a leaf of T (which must exist because the tree is finite)
    - Let w be the parent of v.
    - Removing from T the vertex v and the edge connecting w to v produces a graph T’ with n vertices that is connected and has no simple circuits. Therefore, T’ is a tree.
    - By the inductive hypothesis, T’, which has n vertices, must have n − 1 edges.
    - It follows that T, which has n+1 vertices, has n edges because it has one more edge than T’: the edge connecting v and w.

Trees: properties

- A connected graph with n vertices and (n-1) edges is a tree.
- Any graph without any circuits, n vertices and (n-1) edges is a tree.

Rooted tree: Definition

A rooted tree is a tree in which one vertex has been designated as the root and every edge is directed away from the root.

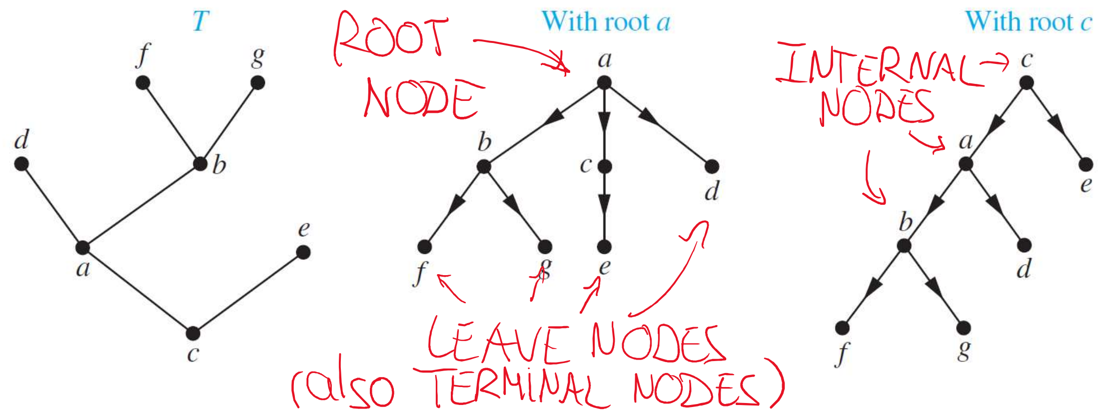
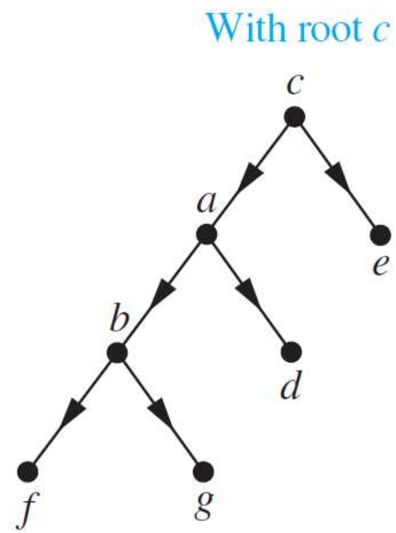


FIGURE 4 **A Tree and Rooted Trees Formed by Designating Two Different Roots.**

Rooted tree: Concepts

Some books use height instead of depth of a tree

- The level of node n, $l(n)$, is the number of intermediate edges in the simple path from the root node of the tree to n. The **depth of a tree** is the maximum level of a node in the tree.
- The nodes in the path [root → n] are called **ancestors** of n. n is a **successor** of such nodes.
- **Parent /child node**: Node $\pi(n)$ is the parent of node n / n is a child node of $\pi(n)$, if they are connected by edge $(\pi(n), n)$ and $\pi(n)$ is closer to the root node, so that $l(n) = l\big(\pi(n)\big) + 1$.
- Nodes who have the same parent are **sibling nodes**.
- A **root node has no parent.**
- A **leaf node has no child nodes**.
- An **internal node** is characterized by having **child nodes**.
- A **subtree of tree rooted at a particular node** is the tree composed of that node as a root and its successors.
- In **ordered rooted trees**, the children of a node are ordered.



m-ary rooted tree: Definition

DEFINITION 3

A rooted tree is called an *m-ary tree* if every internal vertex has no more than m children. The tree is called a *full m-ary tree* if every internal vertex has exactly m children. An m-ary tree with $m = 2$ is called a *binary tree*.

Links

EXAMPLE 3 Are the rooted trees in Figure 7 full m-ary trees for some positive integer m?

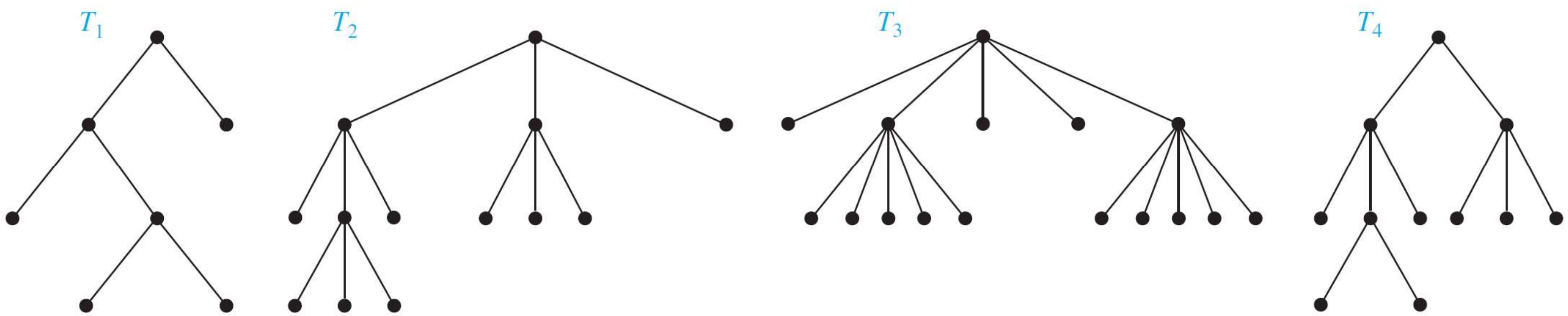


FIGURE 7 Four Rooted Trees.

Binary trees: properties

In a full binary tree, n, the number of vertices is odd and the number of leaves is $\frac{n+1}{2}$.

- Proof [by induction]
  - [base case]: A graph with $n = 1$ nodes is a tree with $1 = \frac{n+1}{2}$
  - [inductive step]: Starting from a tree T with $n = 2k - 1$ vertices and k leaf nodes we build a larger complete binary tree T' by fully expanding a leave node:
    - The selected leaf node becomes an internal node (number of leaves: -1).
    - The new internal node is the parent of two child nodes (number of nodes +2), which are leave nodes (number of leaves: +2).
    - Therefore, the tree T' has $n' = 2k + 1$ vertices and $k' = k + 1 = \frac{n'+1}{2}$ leaf nodes

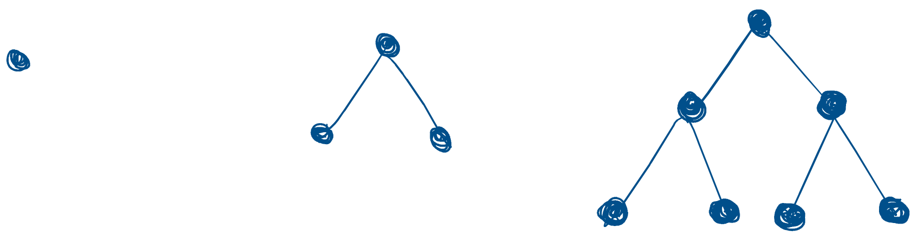
Binary trees: properties

Balanced means that all the leaves are at depth d or depth $d-1$

The minimum depth of a binary tree of n vertices is $\lceil \log_2(n+1) \rceil$

- Proof: Minimum depth trees are well-balanced trees

depth = 0   depth = 1   depth 2



$n = 1 = 2^1 - 1$ $n = 2^2 - 1$ $n = 2^3 - 1$

depth $d \rightarrow n = 2^{d+1} - 1$

If not fully developed

$$n \leq 2^{d_{min}+1} - 1$$

$$\Rightarrow d_{min} \geq \log_2(n+1) - 1$$

$$d_{min} \geq \lceil \log_2(n+1) - 1 \rceil$$

m-ary trees: properties

THEOREM 3 A full m-ary tree with i internal vertices contains $n = mi + 1$ vertices.

Proof: Every vertex, except the root, is the child of an internal vertex. Because each of the i internal vertices has m children, there are mi vertices in the tree other than the root. Therefore, the tree contains $n = mi + 1$ vertices. ◁

m-ary trees: properties

THEOREM 4 A full m-ary tree with

(i) n vertices has $i = (n-1)/m$ internal vertices and $l = [(m-1)n+1]/m$ leaves,

(ii) i internal vertices has $n = mi + 1$ vertices and $l = (m-1)i + 1$ leaves,

(iii) l leaves has $n = (ml-1)/(m-1)$ vertices and $i = (l-1)/(m-1)$ internal vertices.

Proof: Let n represent the number of vertices, i the number of internal vertices, and l the number of leaves. The three parts of the theorem can all be proved using the equality given in Theorem 3, that is, $n = mi + 1$, together with the equality $n = l + i$, which is true because each vertex is either a leaf or an internal vertex. We will prove part (*i*) here. The proofs of parts (*ii*) and (*iii*) are left as exercises for the reader.

Solving for i in $n = mi + 1$ gives $i = (n-1)/m$. Then inserting this expression for i into the equation $n = l + i$ shows that $l = n - i = n - (n-1)/m = [(m-1)n+1]/m$. ◁

m-ary trees: properties

There are at most $\boldsymbol{m^d}$ leaves in an $\boldsymbol{m}$-ary tree of depth $\boldsymbol{d}$.

- Proof [by induction]
  - [base case]: A graph with $d = 0$ has $m^0 = 1$ leaves.
  - [inductive step]: Starting from a tree T of depth $d - 1$, which is assumed to have the maximum number of leaves m^{d-1}, we build a tree T' of depth d by fully expanding each of the leaf nodes.
    - From each of the leaves of T we generate m child nodes.
    - By this expansion, the old leaves of T become internal nodes of T'
    - The number of leaves of T' in this maximal expansion $m^{d-1} \times m = m^d$

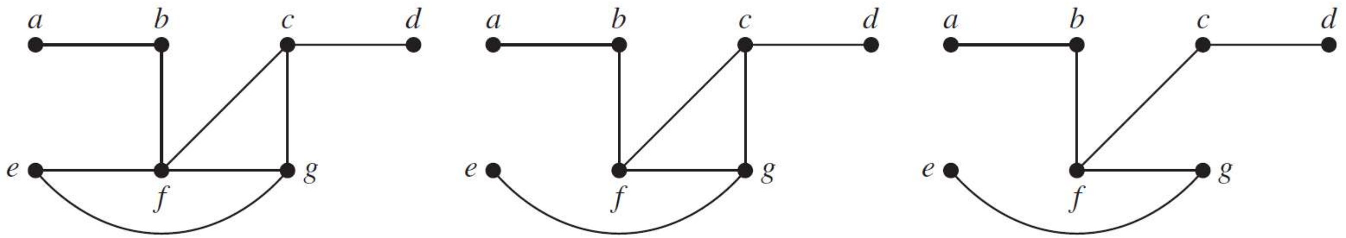
Corollary: For $\boldsymbol{m}$-ary tree tree with $\boldsymbol{l}$ leaves $\boldsymbol{d} \geq \lceil \log_{\boldsymbol{m}} \boldsymbol{l} \rceil$.

If $\boldsymbol{d} = \lceil \log_{\boldsymbol{m}} \boldsymbol{l} \rceil$ the tree is full and balanced.

Balanced means that all the leaves are at depth d or depth $d - 1$

Spanning tree

DEFINITION 1 Let G be a simple graph. A *spanning tree* of G is a subgraph of G that is a tree containing every vertex of G.



Edge removed: $\{a, e\}$ (a)

$\{e, f\}$ (b)

$\{c, g\}$ (c)

THEOREM 1 A simple graph is connected if and only if it has a spanning tree.

Proof:

- If G has a spanning tree, it must be connected
  - The spanning tree is connected. Therefore there is a trajectory between all pairs of vertices in the tree.
  - The spanning tree contains all the vertices in G.
  - Since the spanning tree is a subgraph of G, there is a trajectory in G between all pairs of vertices. Hence, G is connected.
- If G is a simple graph that is connected, it must have a spanning tree
  1. If G is a tree, then the spanning tree coincides with the graph.
  2. If it is not a tree, it must have a simple circuit.
  3. Generate a subgraph by eliminating one of the edges in the simple circuit. Such subgraph remains connected.
  4. If it is a tree, then this subgraph is the spanning tree.
  5. If it is not a tree, repeat steps 2-4 until the spanning tree is found.

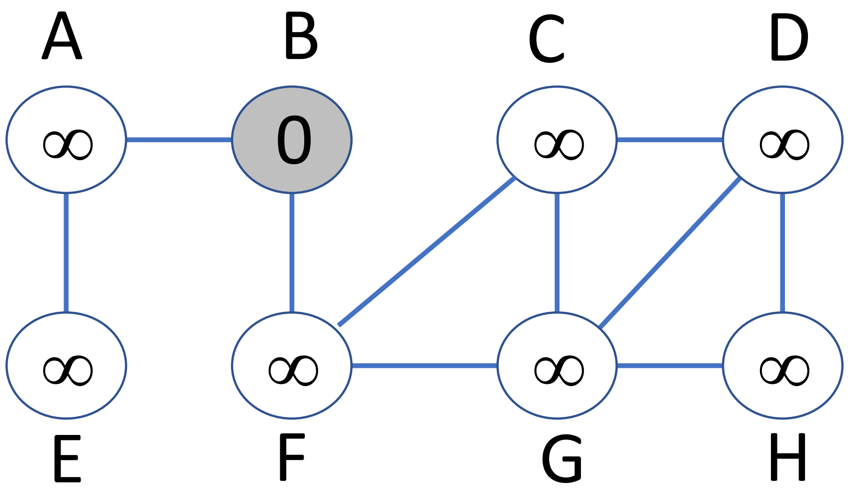
BREADTH-FIRST SEARCH (G, s)

Explore less deep nodes first

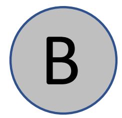
```
for each vertex u ∈ V [G]-s
        do color[u] ← WHITE
           distance[u] ← ∞
           predecessor[u] ← NIL
color[s] ← GRAY
distance[s] ← 0
predecessor[s] ← NIL
Q ← ∅
ENQUEUE (Q, s)
while Q ≠ ∅
        do u ← DEQUEUE (Q)
            for each v ∈ Adj[u]
                 do if color [v] = WHITE
                         then color [v] ← GRAY
                              distance[v] ← distance[u] + 1
                              predecessor[v] ← u
                              ENQUEUE (Q, v)
            color[u] ← BLACK
```

BREADTH-FIRST SEARCH (start)

Source s = B

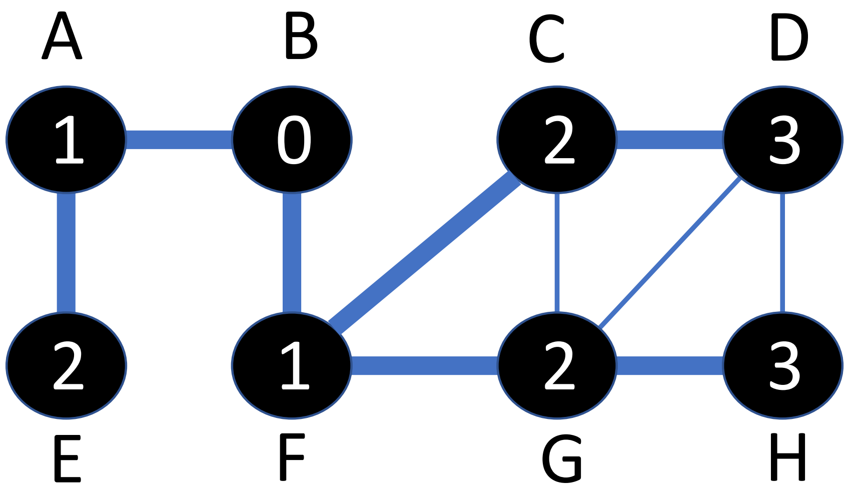


Breadth-first tree



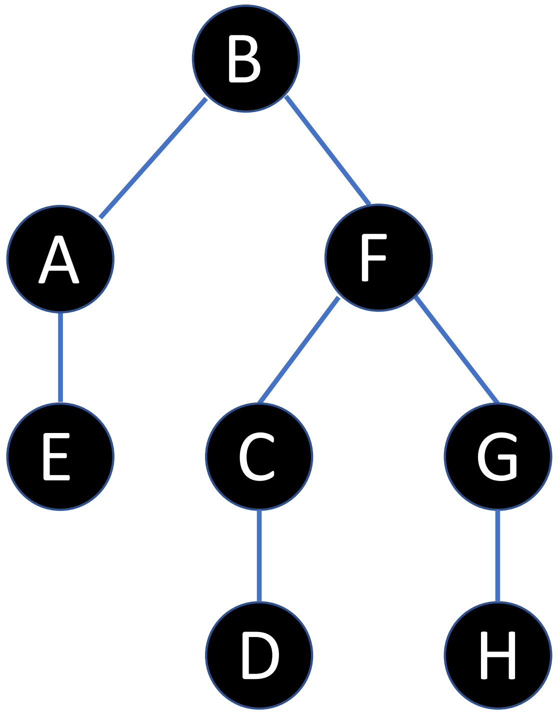
Q = {B_0}

BREADTH-FIRST SEARCH (end)



Q = {}

Breadth-first tree

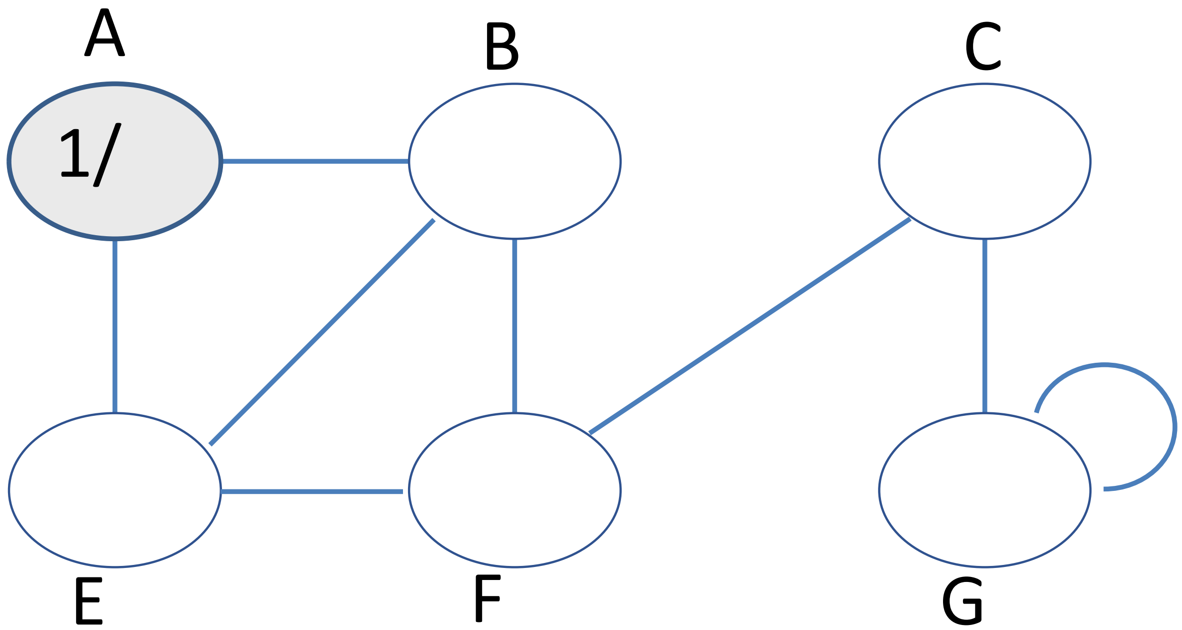


DEPTH-FIRST SEARCH:

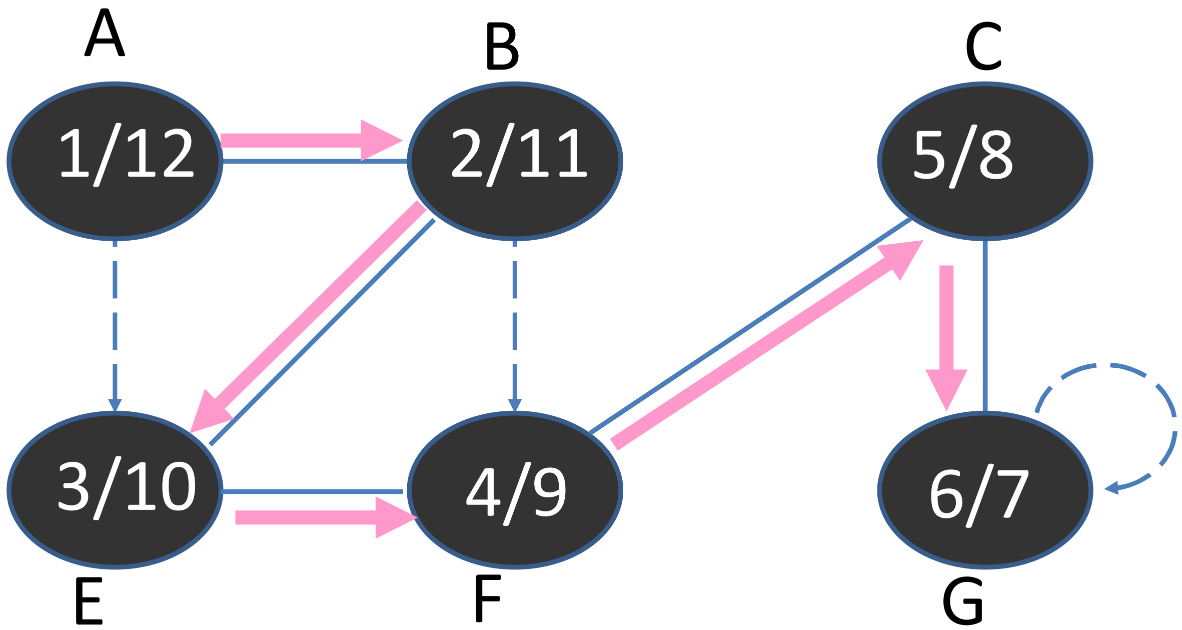
```
DFS (G)
for each vertex u ∈ V [G]
        do color[u] ← WHITE
           predecessor[u] ← NIL
time ← 0
for each vertex u ∈ V [G]
        do if color[u] =  WHITE
                then DFS_VISIT (u)

DFS_VISIT (u)
color [u] ← GRAY              {u from white to gray: vertex u has just been discovered}
time ← time + 1
discovery_time[u] ← time
for each v ∈ Adj[u]           {Explore edge (u,v) }
        do if color [v] = WHITE
                then predecessor[v] ← u
                     DFS_VISIT (v) {recursive step}
color [u] ← BLACK             {u from gray to black: vertex u has been fully explored}
finishing_time[u] ← time ← time + 1
```

DEPTH-FIRST SEARCH (start)

Source s = A

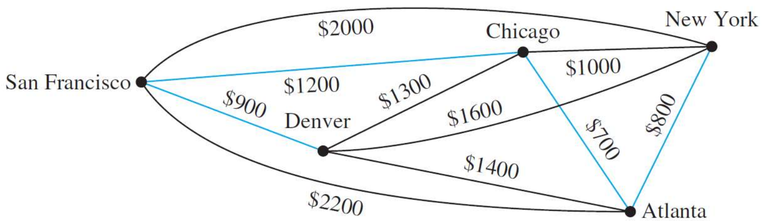


DEPTH-FIRST SEARCH (start)



Minimum spanning trees

DEFINITION 1 A *minimum spanning tree* in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.



Choice	Edge	Cost
1	{Chicago, Atlanta}	$ 700
2	{Atlanta, New York}	$ 800
3	{Chicago, San Francisco}	$1200
4	{San Francisco, Denver}	$ 900
	Total:	$3600

FIGURE 2 **A Minimum Spanning Tree for the Weighted Graph in Figure 1.**

Prim’s algorithm: minimum spanning tree

Include **minimum weight edge** that **does not form a circuit** and **maintains connectivity** **until all nodes are included.**

ALGORITHM 1 Prim’s Algorithm.

procedure *Prim*(G: weighted connected undirected graph with n vertices)
T := a minimum-weight edge
for $i := 1$ **to** $n - 2$
  e := an edge of minimum weight incident to a vertex in T and not forming a
    simple circuit in T if added to T
  $T := T$ with e added
return T {T is a minimum spanning tree of G}

Kruskal’s algorithm: minimum spanning tree

Include **minimum weight edge** that **does not form a circuit** **until all nodes are included.**

ALGORITHM 2 **Kruskal’s Algorithm.**

procedure *Kruskal*(G: weighted connected undirected graph with n vertices)
T := empty graph
for $i := 1$ **to** $n - 1$
  e := any edge in G with smallest weight that does not form a simple circuit
   when added to T
  $T := T$ with e added
return T $\{T$ is a minimum spanning tree of $G\}$