Tree

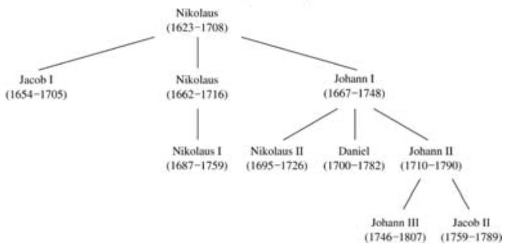
Section 10.1

A Tree

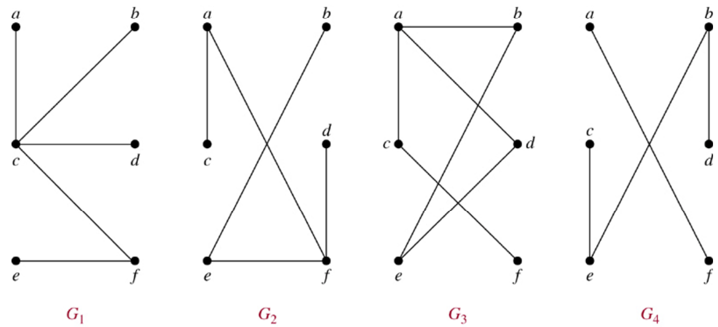
- Definition
  – A **tree** is a connected undirected graph with no simple circuit

Example

- Family Tree

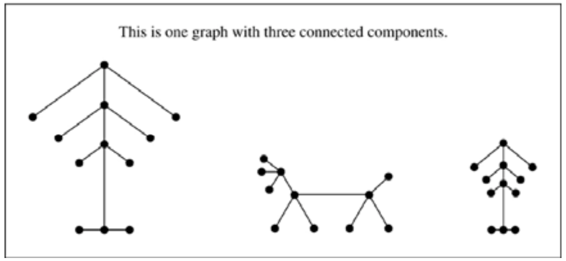


More Example



A Forest

- An unconnected graph such that each connected component is a tree



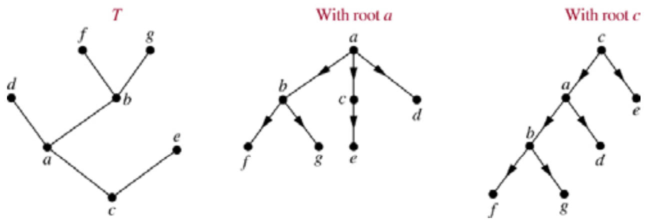
Theorem 1

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

Rooted Tree

- Pick any vertex and declare it as a **root**
  - We can assign a direction to the tree as the direction from the root to every vertex
  - A directed graph result from such direction is the **rooted tree**
  - Different root yields different **rooted tree**

Example



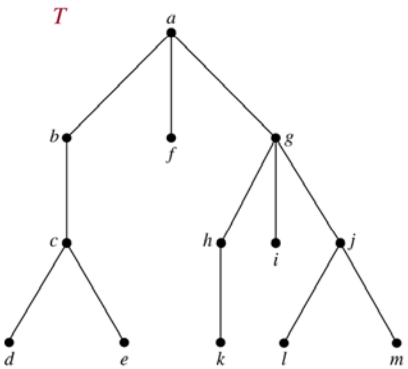
Terminology for a rooted tree

- Parent
  - ***u*** is a parent of ***v*** when ***u*** there is a directed edge from u to ***v***
    - ***v*** is the child of ***u***
- Sibling
  - Vertices with the same parent
- Ancestors of ***u***
  - Any vertex in the path from the root to ***u***
  - i.e., the parent, the parent of the parent, the parent of the parent of the parent,…
- Descendant of ***u***
  - Any vertex having the vertex as its ancestor

Terminology for a rooted tree

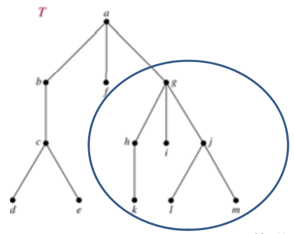
- Definition
  - **Leaf**
    - Any vertex having no children
  - **Internal vertex**
    - Any vertex not being a leaf

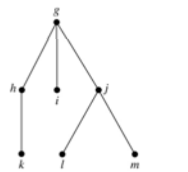
Example



Subtree

- Definition
  - A **subtree** with *v* as its root is a subgraph of a tree that consists of *v* and all its descendants and the corresponding edges

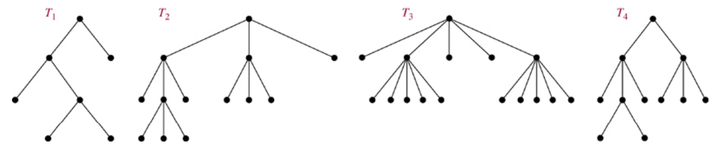




M-ary tree

- Definition
  - A tree is call **m-ary tree** if every internal vertex has no more than m children
  - A tree is called "**full m-ary tree**" when every internal vertex has exactly m children
  - And *m*-ary tree where $m = 2$ is called a binary tree

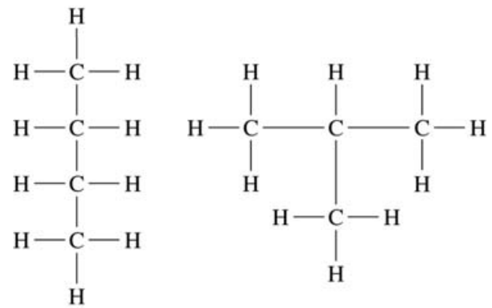
Example



An Ordered Tree

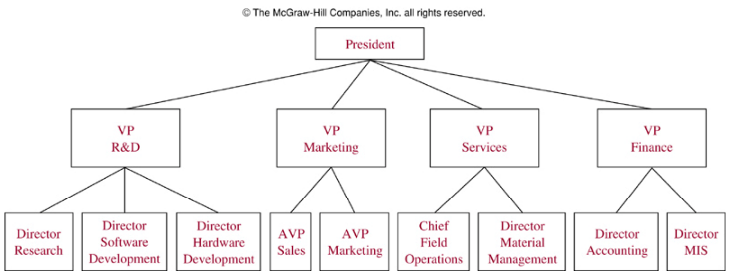
- A rooted tree such that we have "ordered" on the children of an internal node
  - When we draw an ordered tree, the children are drawn from left to right according to its order
- For binary tree
  - The first child is called the "**left**" child
    - The tree rooted at the left child is called the left subtree
  - The second child is called the "**right**" child
    - The tree rooted at the right child is called the right subtree

Tree as a model



Butane   Isobutane

Tree as a model



Properties of Trees

- Theorem 2
  - A Tree with n vertices has $n - 1$ edges
- Proof
  - Convert a tree into a rooted tree
  - we see that the arrow of the directed tree point out from root
  - For every directed edge, there is one vertex associated with the edge
    - Except the root
  - Since we have $n - 1$ non-root vertices
    - Then we have exactly $n - 1$ edge

Properties of Trees

- Theorem 3
  - A full *m*-ary tree with i internal vertices contains $n = mi + 1$ vertices
- Proof
  - Every vertex , except the root, in a tree is a child of an internal vertex
  - Since the tree is "full", every internal vertex has exactly m children
    - So we have mi vertices, exclude the root

Properties of Trees

- Theorem 4
  - A full m-ary tree with
    1) n vertices *has* $i = (n-1)/m$ internal vertices and $l = [(m-1)n + 1]/m$ leaves
    2) i internal vertices has $n = mi + 1$ vertices and $l = (m - 1)i + 1$
    3) l leaves has $n = (ml - 1)/(m-1)$ vertices and $i = (l - 1)/(m - 1)$ internal vertices

Proof of theorem 4

- Let define for a full m-ary tree
  - l = number of leaves
  - i = number of internal vertices
  - n = total number of vertices
- We use theorem 3 ***($n = mi + 1$)*** and a fact that ***$n = i + l$*** (because a vertex must be either a leave or an internal vertex
- Try 1)
  - Solve $n = mi + 1$ for i
    - $i = (n - 1)/m$
  - Put this back into $n = i + l$ yields
    - $l = n - i = n - (n - 1)/m$
    - $= mn/m - (n-1)/m = [nm - (n - 1)]/m = [(m - 1)n + 1]/m$

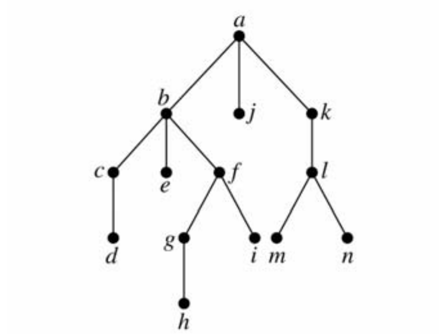
Balanced Tree

- It is a desirable property to have a tree that is "almost" full
  - i.e., we could say that we like balanced tree
  - Intuitively, a tree such that each subtree are of similar size

Level and Height

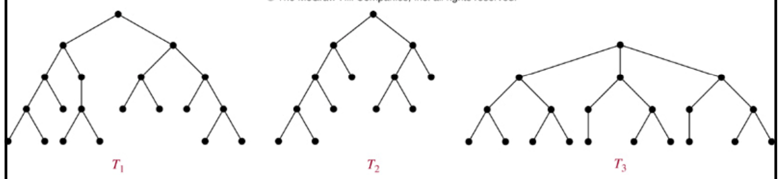
- Definition
  - The **level** of a vertex ***v*** is the length of the unique path from the root to ***v***
    - The level of the root is defined as 0
  - The **height** of a rooted tree is the maximum of the levels of its vertices

Example



Balanced Tree

- Definition
  - A balanced tree is a rooted m-ary tree of height h such that all leaves are at level h or $h - 1$



Theorem 5

- There are at most m^k leaves in an m-ary tree of height h
- Proof
  - By induction on the height of a tree

Proof: The basis step

- The base case
- Tree of height 1
  - The root is the only internal vertex
  - We can't have more than m children for the root
    - These children are leaves
    - So, the number of leaves is at most m^1
- Hence, the base case is correct

Proof: the Inductive Step

- Assume that the result is true for the height less than h, i.e., it is true for h-1, h-2, h-3, … but not h
- Consider a tree T of height h
  - Take out the root of T
  - Consider the remaining subtree of the root
    - These subtrees are of height at most $h-1$
    - Hence, we can say that each subtree has at most $m^{(h-1)}$ leaves

Proof: the Inductive step

- Now, the leaves of these subtrees are exactly the leaves of ***T***
- Since we have at most ***m*** subtrees
  - The number of leaves is at most $m * m^{(h-1)} = m^h$
- Hence, the theorem is correct by the principle of mathematical induction

Corollary 1

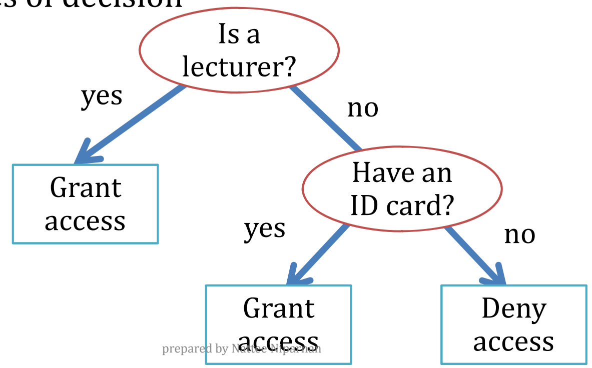
- If an m-ary tree of height ***h*** has ***l*** leaves, then $h \geq ceil(log_m l)$
- If an m-ary tree is full and balance, then $h = ceil(log_m l)$
  - (note that ceil is the ceiling function (round up))
- The proof is simple, try to read it from the book

Application of Tree

Section 10.2

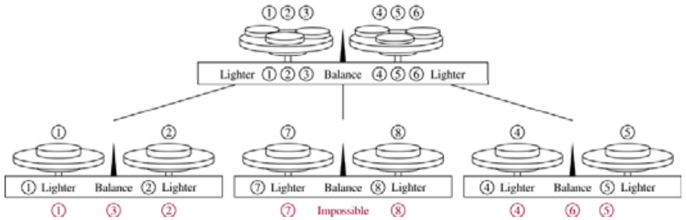
Decision Tree

- We use tree to describe a process involving series of decision



Eight coin problem

- We have 8 coins
  - One of them is a fake coin, which is lighter
  - We have a balance scale
  - How to determine a fake one?



Comparing three variables

- We have *a,b,c* all are distinct
  - We which to sort these variable