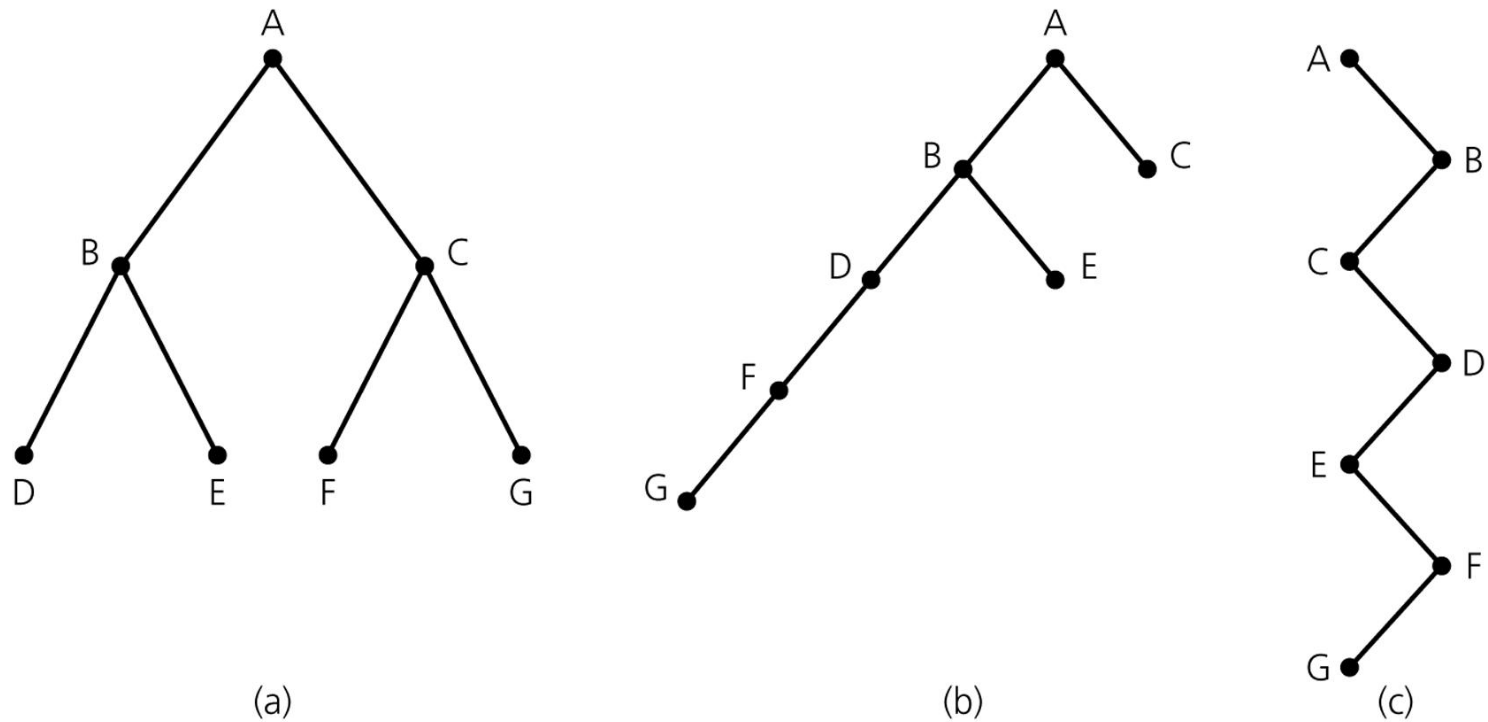
Trees

CSc106

Terminology

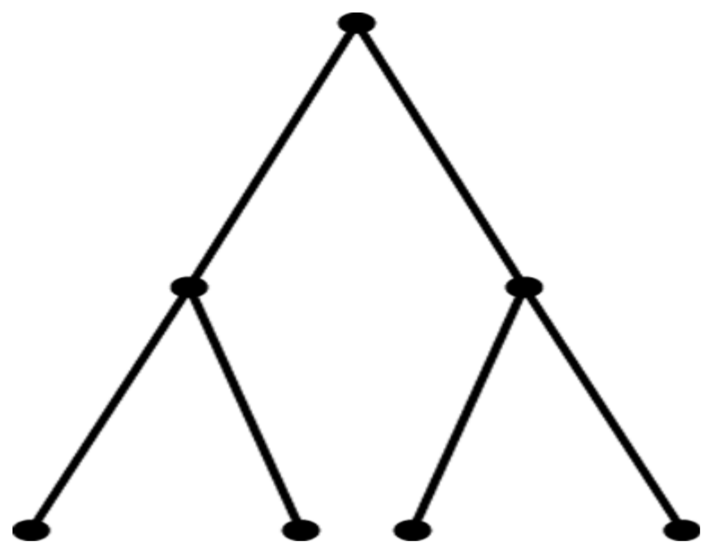
- The height of trees
  - *Level of a node* n in a tree T
    - If n is the root of T, it is at level 1
    - If n is not the root of T, its level is 1 greater than the level of its parent
  - *Height of a tree* T defined in terms of the levels of its nodes
    - If T is empty, its height is 0
    - If T is not empty, its height is equal to the maximum level of its nodes

Terminology



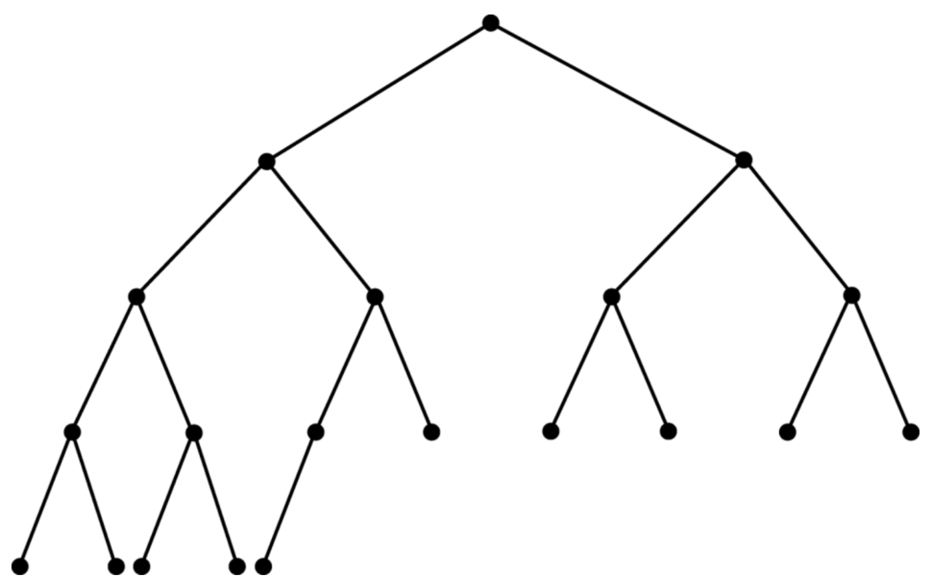
Terminology

- Full, complete, and balanced binary trees
  - Recursive definition of a full binary tree
    - If T is empty, T is a full binary tree of height 0
    - If T is not empty and has height h > 0, T is a full binary tree if its root's subtrees are both full binary trees of height h – 1



Terminology

- Complete binary trees
  - A binary tree T of height h is complete if
    - All nodes at level h – 2 and above have two children each, and
    - When a node at level h – 1 has children, all nodes to its left at the same level have two children each, and
    - When a node at level h – 1 has one child, it is a left child



Terminology

- Balanced binary trees
  - A binary tree is balanced if the height of any node's right subtree differs from the height of the node's left subtree by no more than 1
- Full binary trees are complete
- Complete binary trees are balanced

Number of Nodes (n) *vs* Height (h)

For *Full Binary Trees:*

- $h = \log_2(n+1)$ or $n = 2^h - 1$

For *Complete Binary Trees*:

- $h \leq \log_2(n+1)$ or $h = \lceil \log_2(n+1) \rceil$

For *Balanced Binary Trees*:

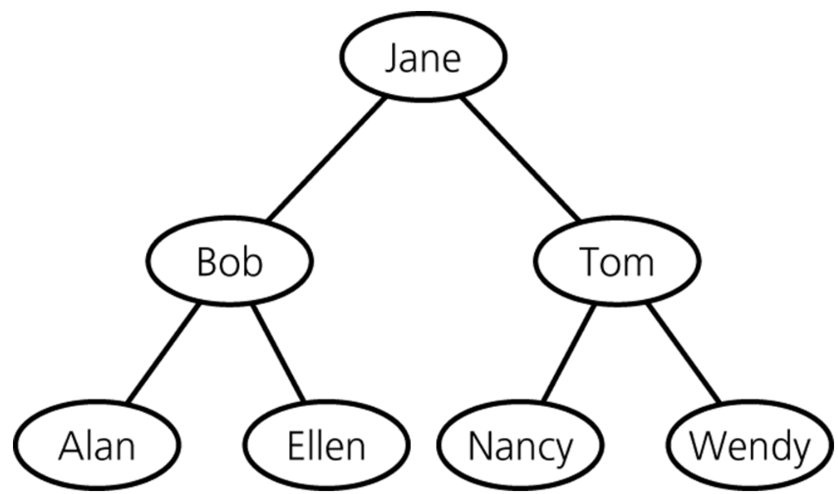
- $h \leq \log_2(n+1)$

Binary Search Tree

A binary tree that has the following properties for each node n

- n's value is greater than all values in its left subtree T_L
- n's value is less than all values in its right subtree T_R
- Both T_L and T_R are binary search trees

Construct BST for:
60,20,70,10,50,30,40



Traversals of a Binary Tree

- A traversal algorithm for a binary tree visits each node in the tree
- Big O run-time for a Traversal?
- Recursive traversal algorithms (examples on next slide)
  - Preorder traversal
  - Inorder traversal
  - Postorder traversal

Binary Tree Traversal

DisplayInPreOrder(binTree)

- if (binTree not empty)
  - display data in root of binTree
  - DisplayInPreOrder(left subtree of binTree)
  - DisplayInPreOrder(right subtree of binTree)

DisplayInPostOrder(binTree)

- if (binTree not empty)
  - DisplayInPostOrder(left subtree of binTree)
  - DisplayInPostOrder(right subtree of binTree)
  - display data in root of binTree

DisplayInOrder(binTree)

- if (binTree not empty)
  - DisplayInOrder(left subtree of binTree)
  - display data in root of binTree
  - DisplayInOrder(right subtree of binTree)