



# Trees

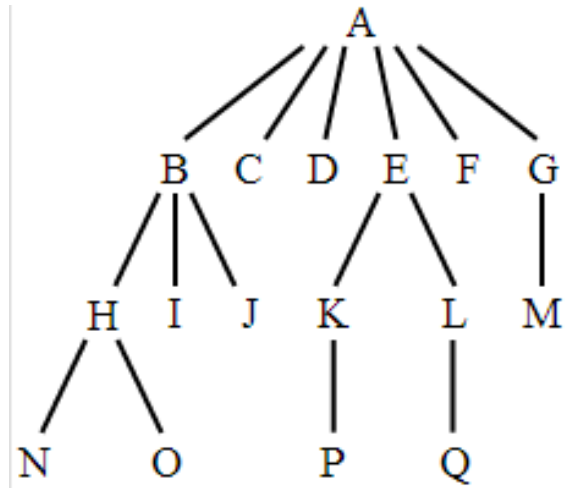
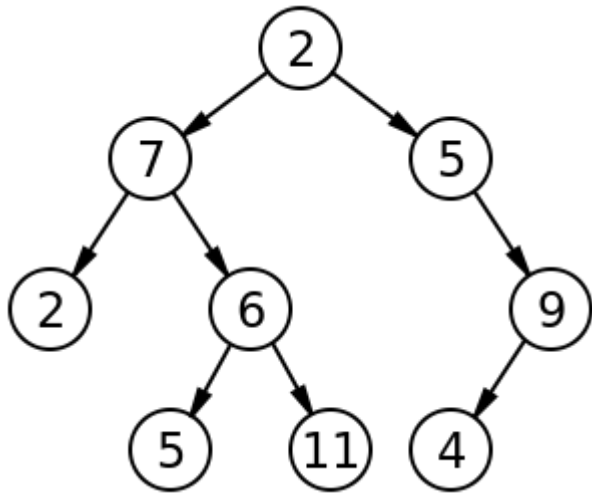
Until now, we have dealt with linear data structures such as:

- arrays
- linked lists
- stacks
- queues

A tree is:

- a nonlinear data structure where members may have multiple successors
- a data structure made up of nodes.

# Trees



# Tree Terminology

root – unique starting node

parent – predecessor of a node

child – successor of a node

leaf – a node with no children

siblings – two nodes with the same parent

ancestors – let  $A$  be an arbitrary node of a tree. If  $A$  is the root node, then  $A$  has no ancestors; otherwise, the parent of  $A$  and all ancestors of  $A$ 's parent are ancestors of  $A$

What kind of definition is ancestor?

# Tree Terminology

descendants – let B be an arbitrary node of a tree. If B is a leaf node, then B has no descendants; otherwise, each child of B and all descendants of each child of B are descendants of B.

subtree – an arbitrary node in the tree and all descendants of that node

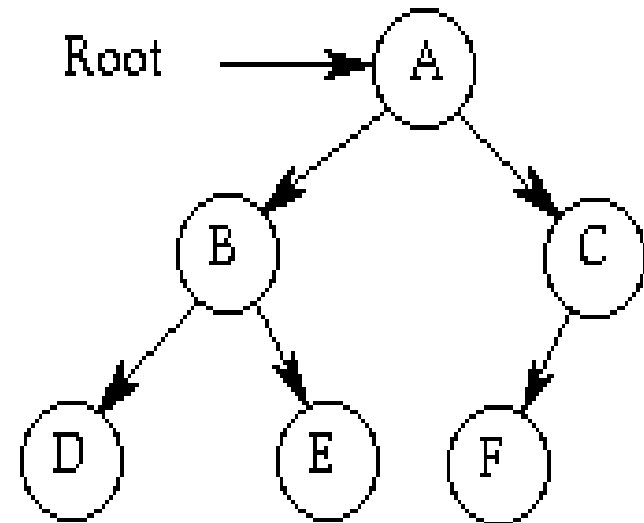
level – the root node is level 1 and every other node in the tree is at level n where n is the number of nodes in the path from the root node to the node in question

depth (or height) – maximum level of any node in the tree

# Identify Tree Attributes

For the given tree, identify:

- a) root
- b) parent of E
- c) children of A
- d) leaf nodes
- e) any two siblings
- f) ancestors of B
- g) descendants of F
- h) level of D
- i) depth of the tree



# Binary Tree

Characteristics of a binary tree:

a) Each parent can have at most two children

b) A binary tree can be empty

c) If a binary tree has two children, the child on the left is the "left child" and the one on the right is the "right child"

Note: The left child is the root of the left subtree and the right child is the root of the right subtree

# Some Binary Tree Operations

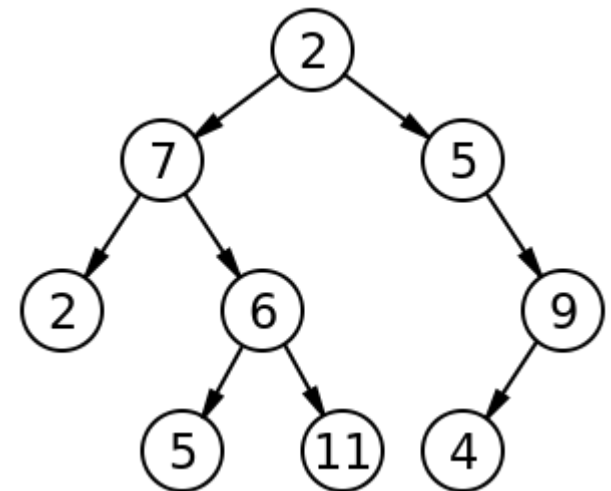
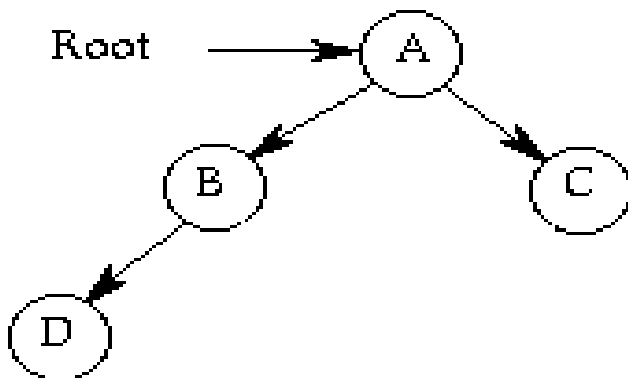
Before defining the Binary Tree ADT, let's work a few problems.

- 1) Write the appropriate data structure definitions for a binary tree.
- 2) We can define three traversal methods for a binary tree:
  - a) inorder: Left, Visit, Right
  - b) preorder: Visit, Left, Right
  - c) postorder: Left, Right, Visit

# Identify

For the following binary tree, identify the inorder, preorder, and postorder traversals.

Write a C function that performs an inorder traversal of a binary tree.





# Binary Search Tree (BST)

## Definition

Consider an arbitrary node in a tree called A.

All values in the left subtree are less than the value in A.

All values in the right subtree are greater than the value in A.

# Create BST

Create a BST for the following strings (note: apr < jan):

jan, feb, mar, apr, may, jun, jul, aug, sep, oct  
nov, dec

# Traversals

If visiting a node means printing the contents of the node, show each of the following traversals of the newly created BST.

- preorder
- inorder
- postorder

# BST Functions

- Write an algorithm for `bstInsert`.
- What is the worst case computing complexity of your algorithm? Why?
- Write the C function `bstInsert`.
- Write a C function `bstFindLevel` that returns the level of a node in a BST.
- Write a C function `btFindLevel` that returns the level of a node in a binary tree.