Tree

Tree

- A nonlinear data structure
- Contain a distinguished node R, called the root of tree and a set of sub trees.
- Two nodes n1 and n2 are called siblings if they have the same parent node.

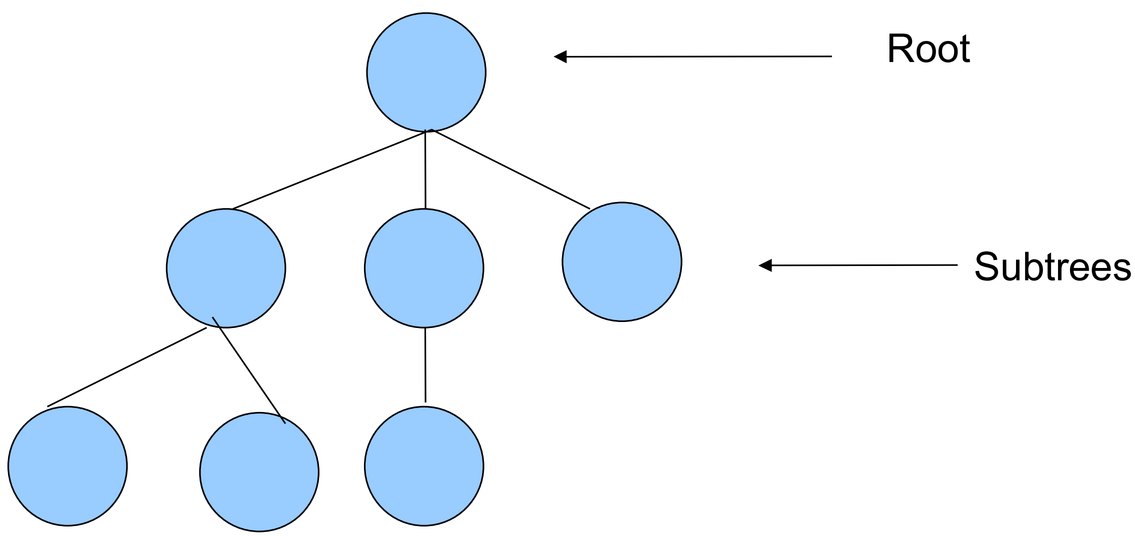


Figure: Tree

Binary Tree

- A binary tree T is defined as a finite set of elements, called nodes such that:
- T contains a distinguished node R, called the root of T and the remaining nodes of T form an ordered pair of disjoint binary trees T1 and T2. T1 and T2 are called the left and right subtrees of R.
- Any node N in a binary tree T has either 0, 1 or 2 successors.
- Nodes with no successors are called terminal nodes or leaf nodes.
- Example:

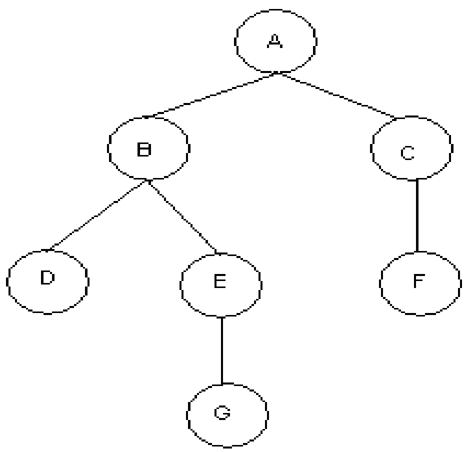


Figure: Binary Tree T

Binary Tree: T
Root: A
Nodes with 2 Successors: A, B
Nodes with 1 Successors: C, E
Terminal Nodes: D, F, G

Some Basic Terms

- **Edge:** A line from a node N of T to a successor is called an edge.
- **Path:** A sequence of consecutive edges is called a path.
- **Branch:** A path from root node to a leaf node is called branch.
- **Level of Binary Tree:** Each node in a binary tree T is assigned a level number. The root R of T has level number 0 and every other node has level number which is one more than the level number of its parent.
- **Depth of Binary Tree:** Maximum number of nodes in a branch of T is the depth of T.

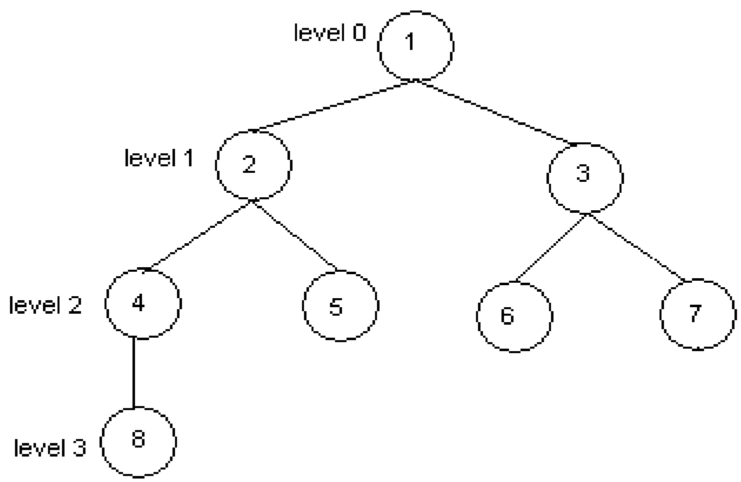
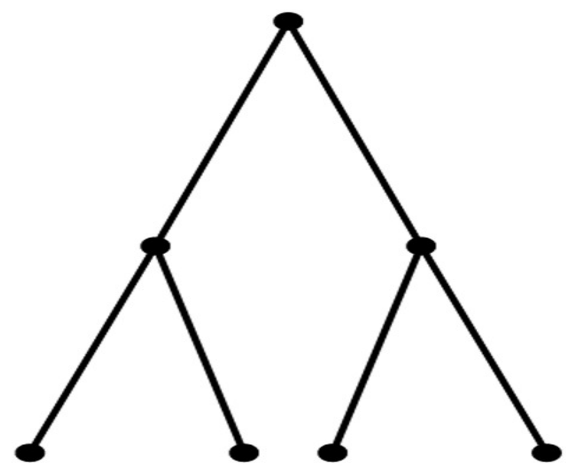


Figure: Binary Tree T.

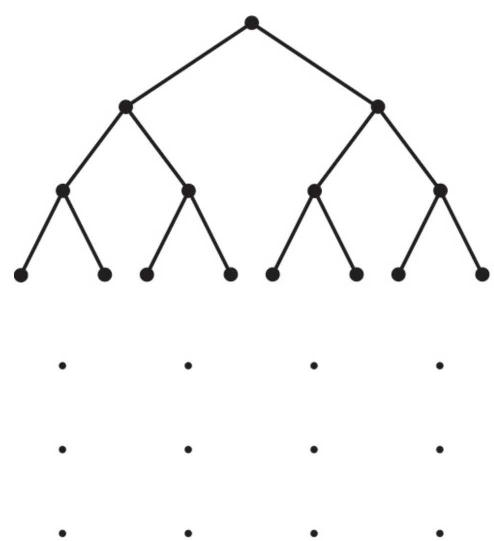
Binary Tree: T
Edge: (1, 2), (3, 6) ....
Path: (1, 2, 4), (1, 3, 6)
Branch: (1, 2, 4, 8), (1, 2, 5), (1, 3, 6), (1, 3, 7)
Depth: 4

Full Binary Trees

- Full binary tree
  - All nodes that are at a level less than h have two children, where h is the height
- Each node has left and right subtrees of the same height



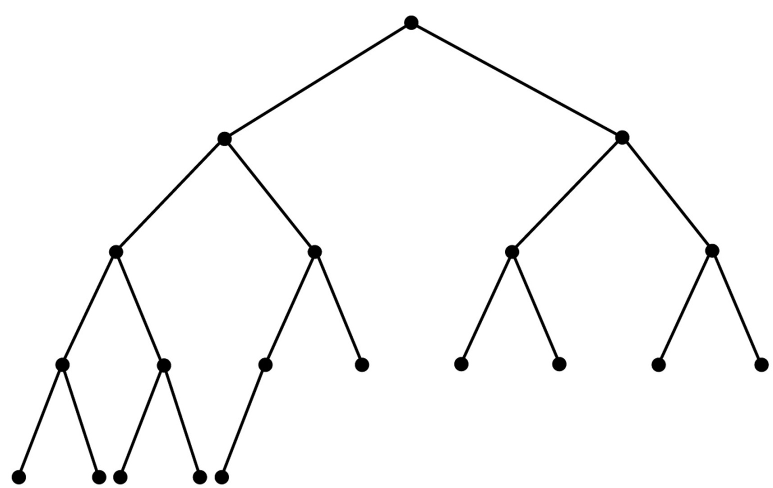
Full Binary Tree



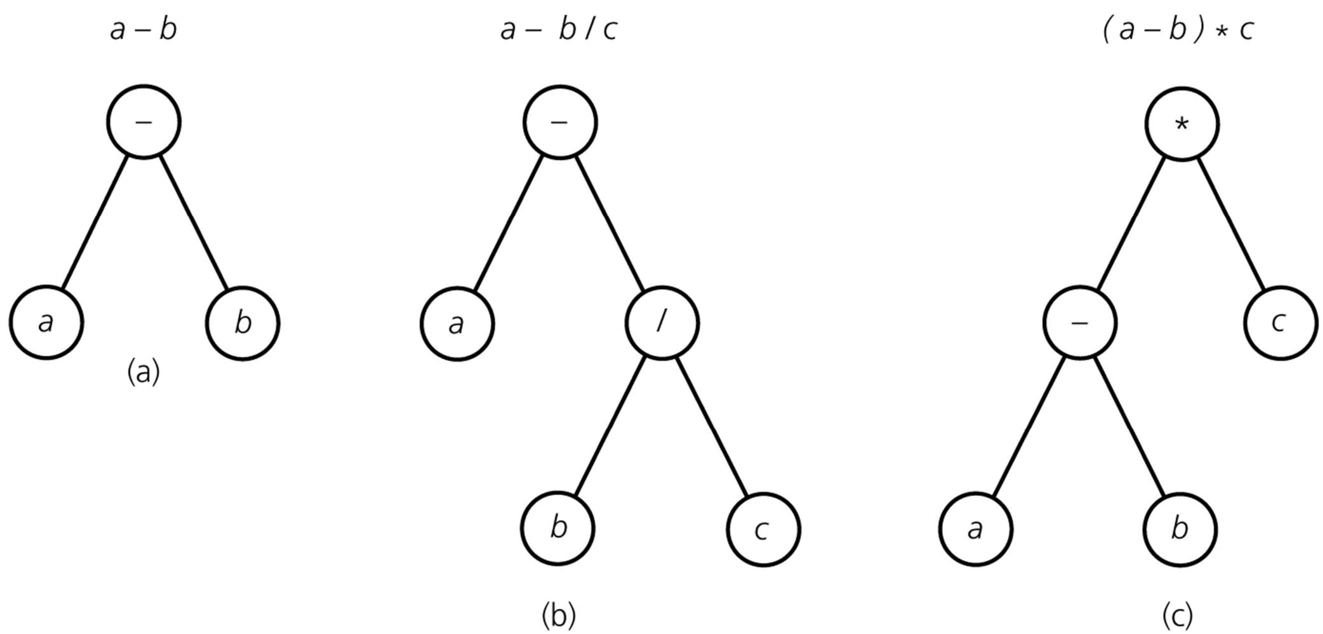
Level	Number of nodes at this level	Number of nodes at this and previous levels
1	$1 = 2^0$	$1 = 2^1 - 1$
2	$2 = 2^1$	$3 = 2^2 - 1$
3	$4 = 2^2$	$7 = 2^3 - 1$
4	$8 = 2^3$	$15 = 2^4 - 1$
•	•	•
•	•	•
•	•	•
h	2^{h-1}	$2^h - 1$

Complete Binary Trees

- Complete binary tree
  - A binary tree full down to level h-1, with level h filled in from left to right



Represent Algebraic Expressions using Binary Tree



Representation of Algebraic Expression Using Binary Tree

Expression E = ((a + b) * r + w / t) * x

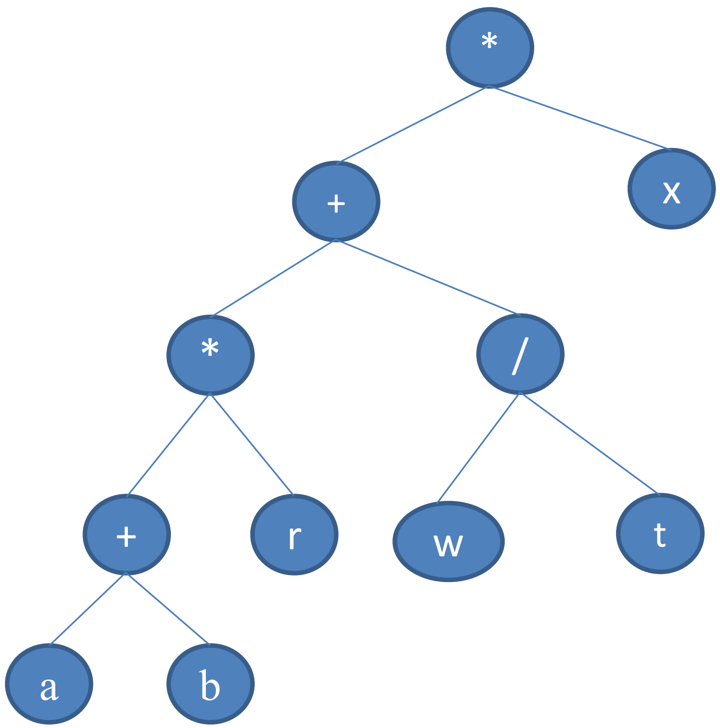


Figure: Binary Tree T Expressing the Algebraic Expression E.

Sequential Representation of Binary Tree

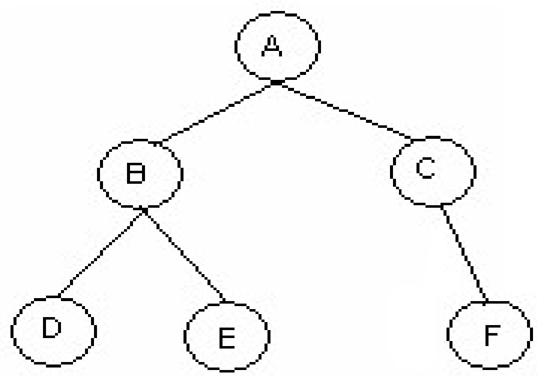
- Use only a single liner array Tree.

(a)Tree[1] represents the Root of T.

(b)If node N is in Tree[K], then its left child is in Tree[2K] and right child is in Tree[2K+1].

(c)Tree[1] = NULL indicates T is empty.

- Example:



Tree =

1	2	3	4	5	6	7	8	9	10
A	B	C	D	E		F			

Figure: Binary Tree T and its sequential representation.

Traversing a Binary Tree

Traversing Binary Tree

There are 3 ways of traversing a binary tree T having root R.

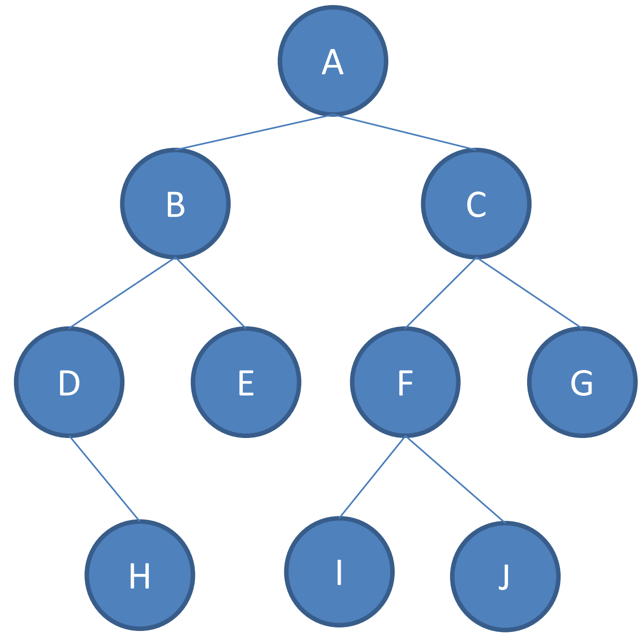
1. Pre-order Traversing

Steps:

(a) Process the root R

(b) Traverse the left subtree of R in preorder.

(c) Traverse the right subtree of R in preorder.



Preorder Traversal of T

A, B, D, H, E, C, F, I, J, G

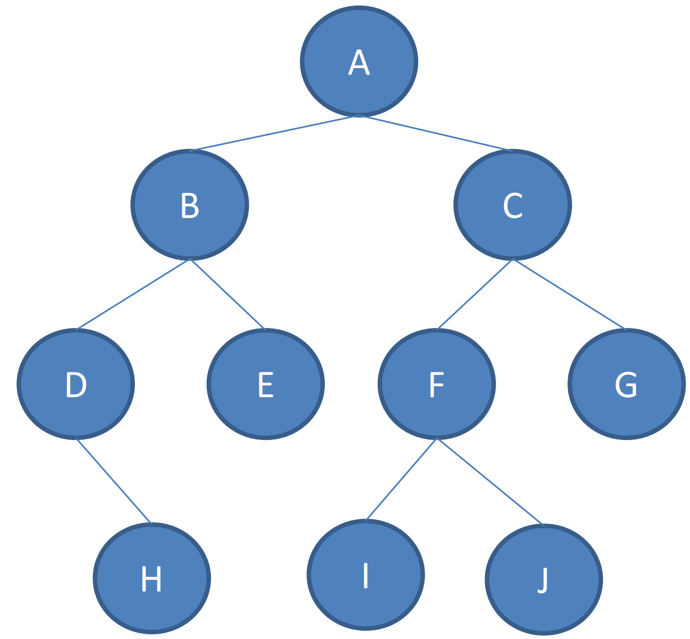
2. In-order Traversing

Steps:

(a) Traverse the left subtree of R in inorder.

(b) Traverse the root R.

(c) Traverse the right subtree of R in inorder.



Inorder Traversal of T

D, H, B, E, A, I, F, J, C, G

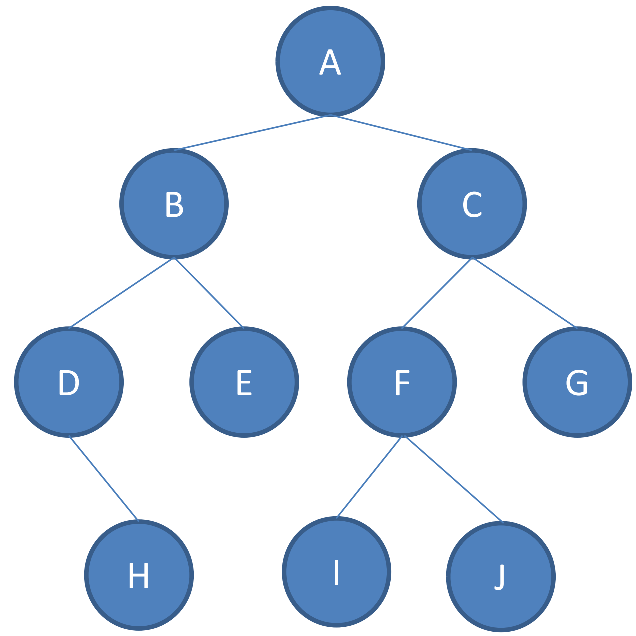
3. Post-order Traversing

Steps:

(a) Traverse the left subtree of R in postorder.

(b) Traverse the right subtree of R in postorder.

(c) Traverse the root R.



Postorder Traversal of T

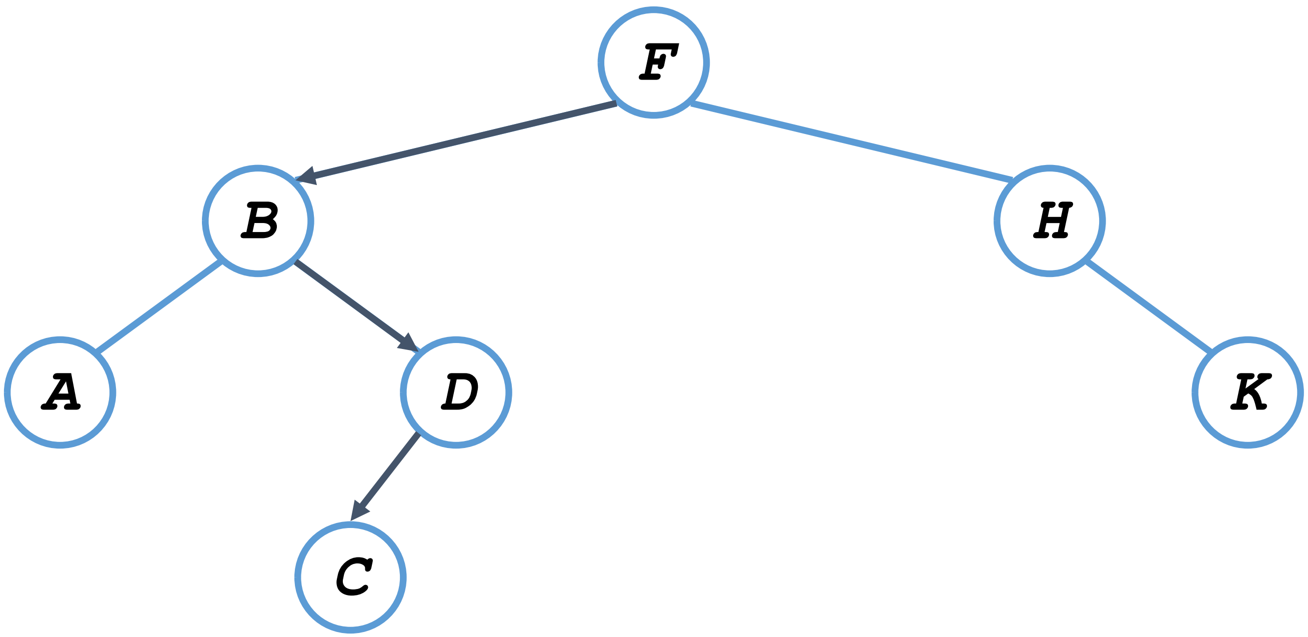
H, D, E, B, I, J, F, G, C, A

Operations of BSTs: Insert

- Adds an element x to the tree so that the binary search tree property continues to hold
- The basic algorithm
  - Like the search procedure above
  - Insert x in place of NULL
  - Use a “trailing pointer” to keep track of where you came from (like inserting into singly linked list)

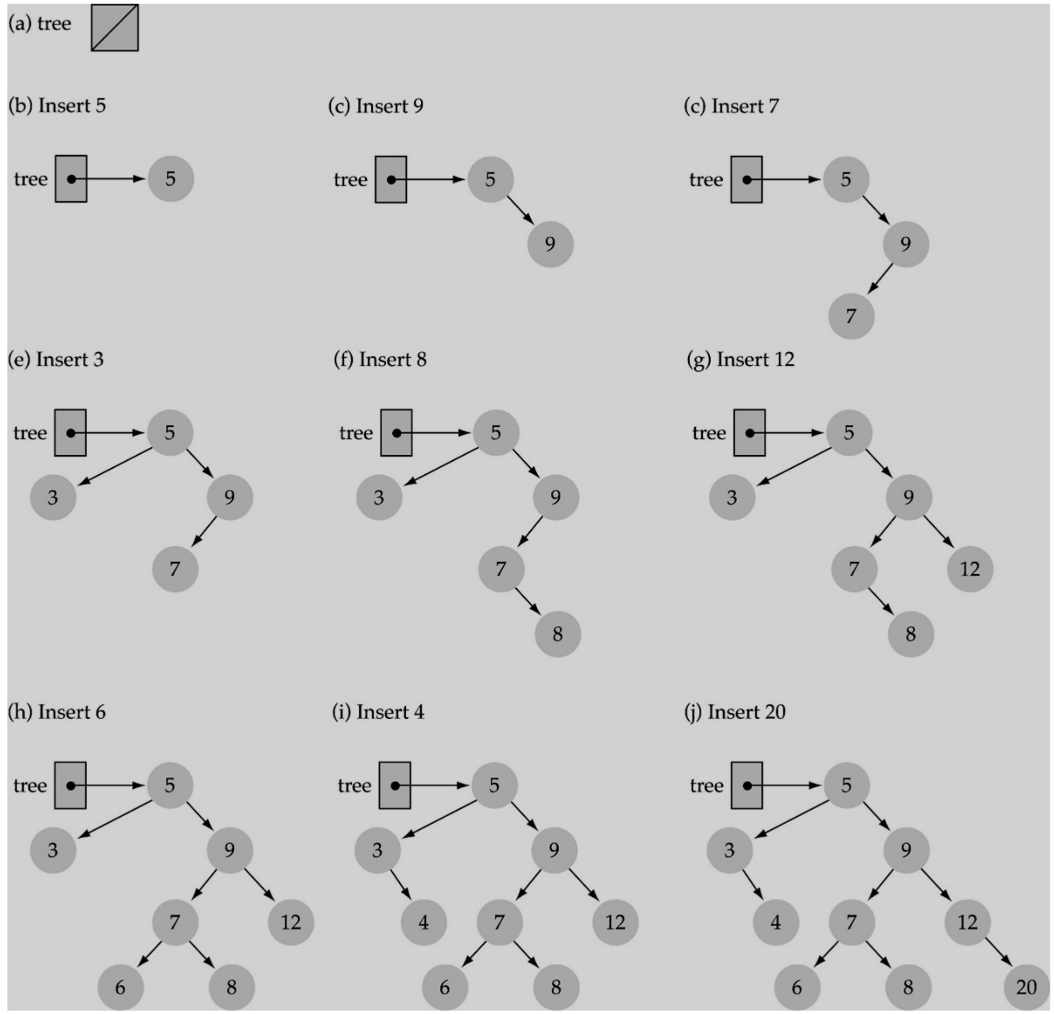
BST Insert: Example

- Example: Insert *C*



Function InsertItem

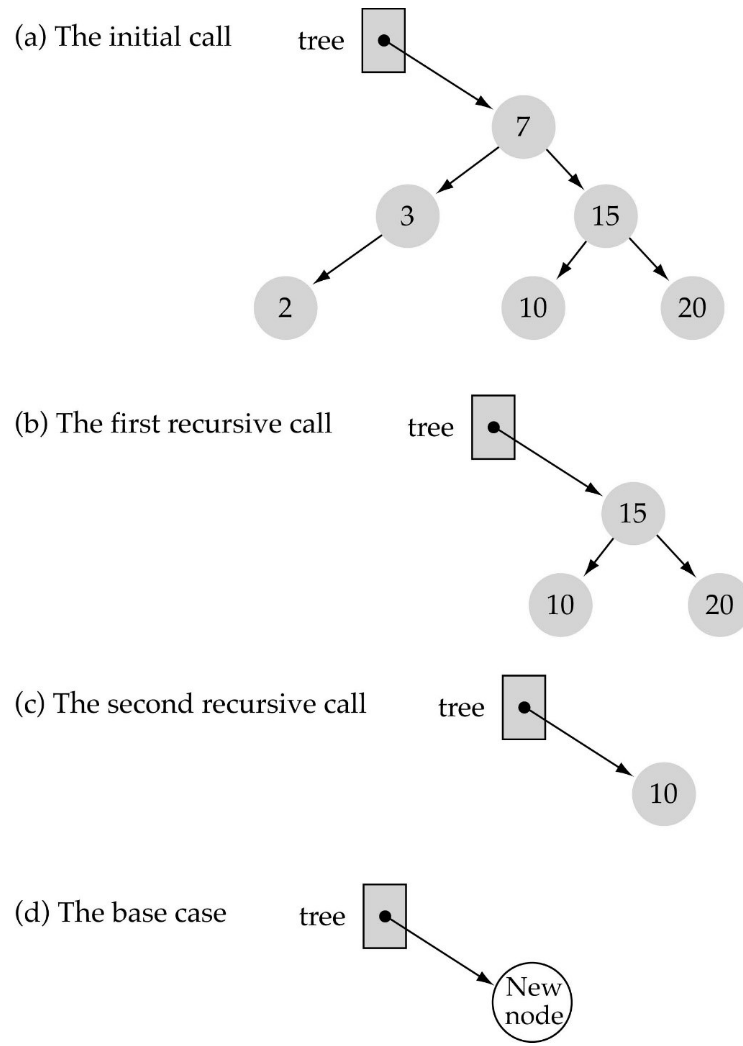
- Use the binary search tree property to insert the new item at the correct place



Function InsertItem (cont.)

-

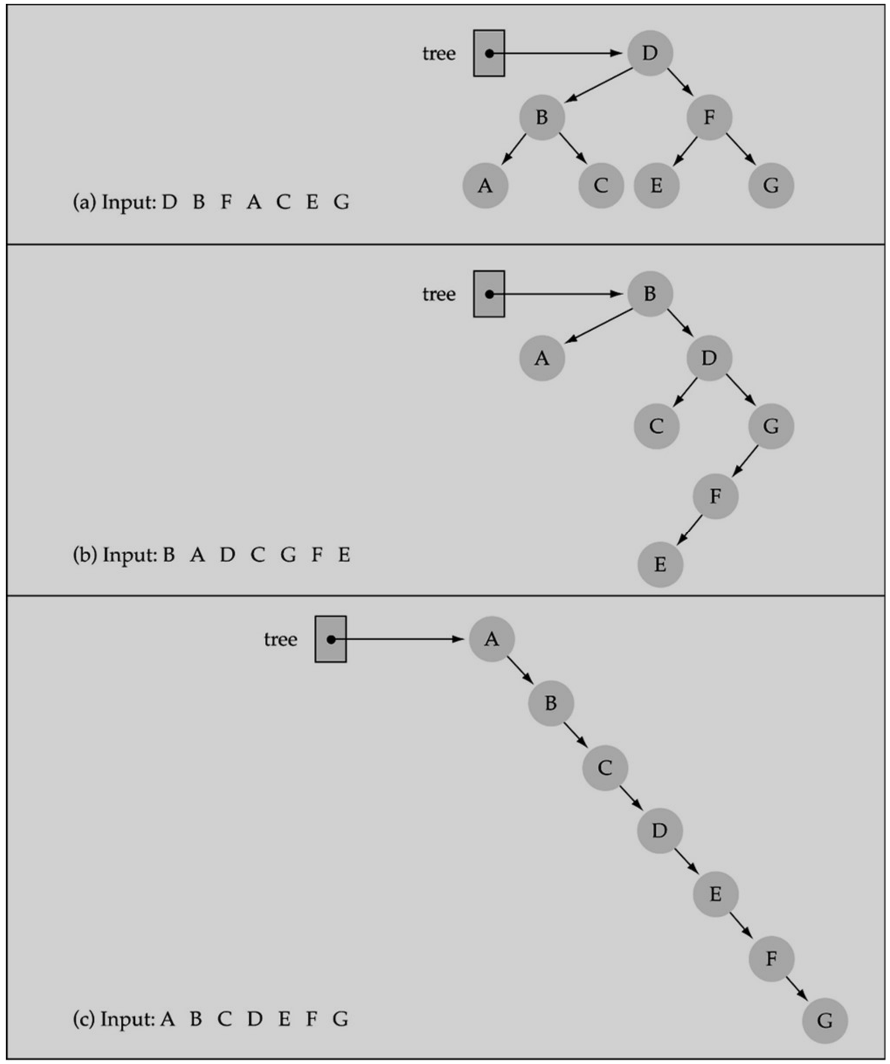
e.g., insert 11



Does the order of inserting elements into a tree matter?

- Yes, certain orders might produce very unbalanced trees!

Does the order of inserting elements into a tree matter? (cont.)



Does the order of inserting elements into a tree matter? (cont'd)

- Unbalanced trees are not desirable because search time increases!
- Advanced tree structures, such as red-black trees, guarantee balanced trees.

BST Delete Item

- First, find the item; then, delete it
- Binary search tree property must be preserved!!
- We need to consider three different cases:

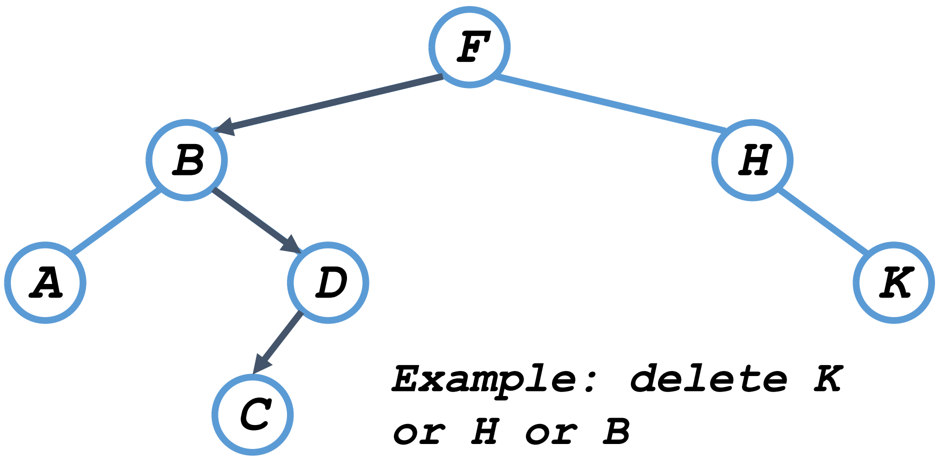
  (1) Deleting a leaf

  (2) Deleting a node with only one child

  (3) Deleting a node with two children

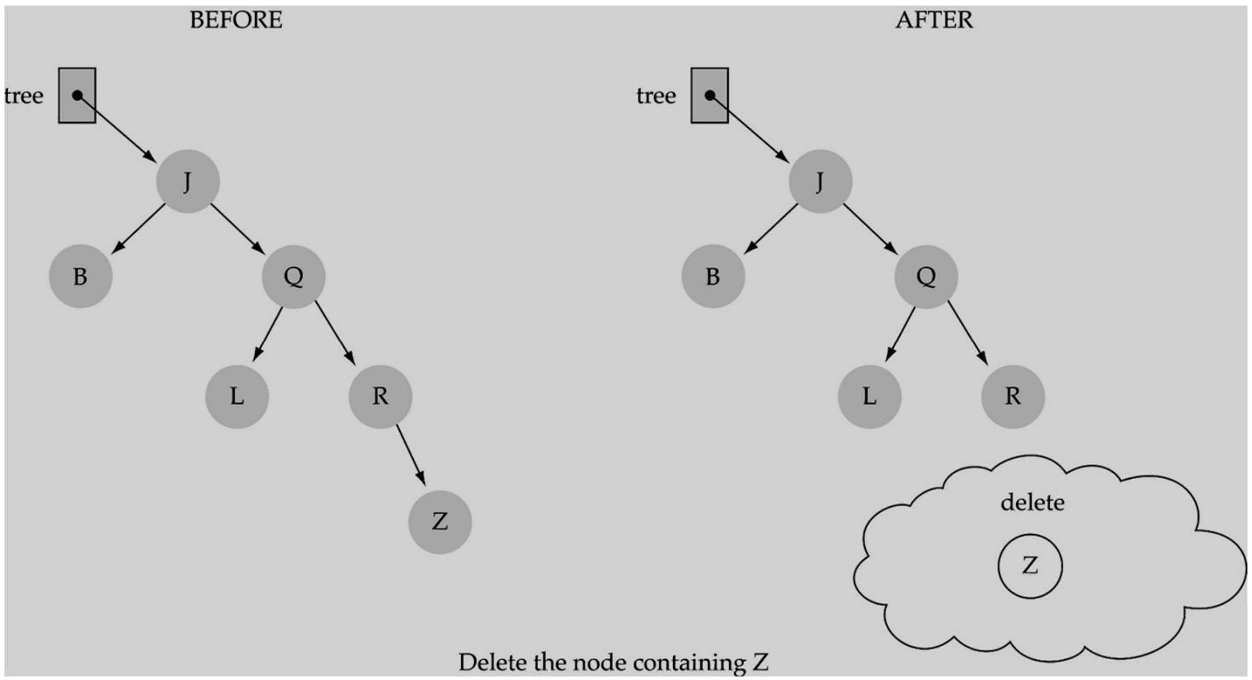
BST Operations: Delete

- Deletion is a bit tricky
- 3 cases:
  - x has no children:
    - Remove x
  - x has one child:
    - Splice out x
  - x has two children:
    - Swap x with successor
    - Perform case 1 or 2 to delete it

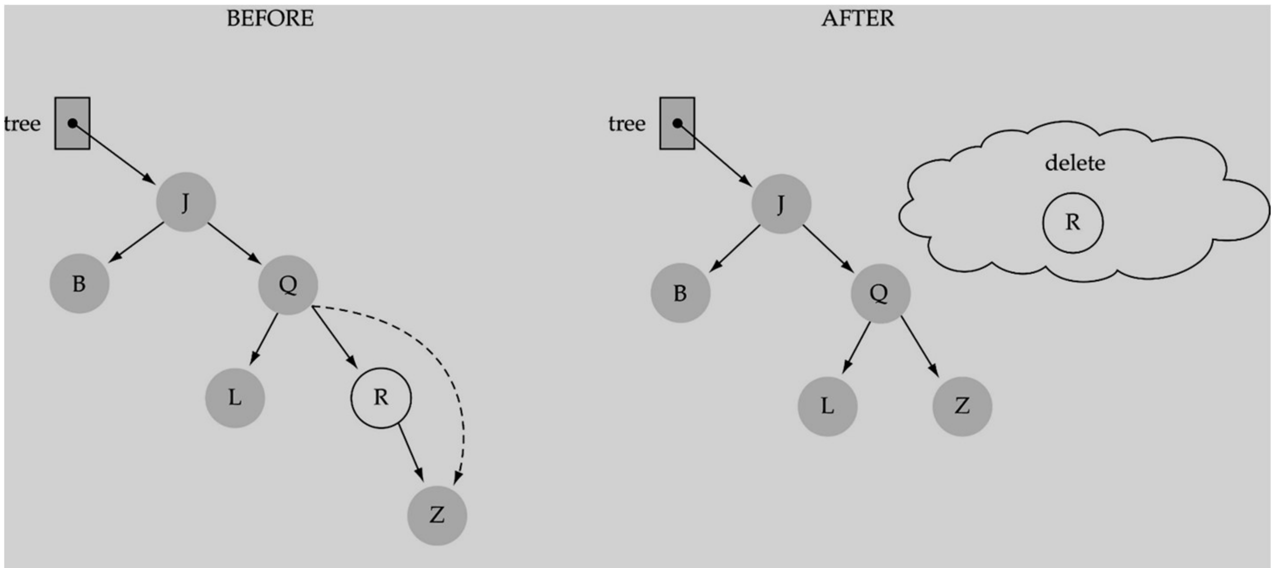


Example: delete K or H or B

(1) Deleting a leaf

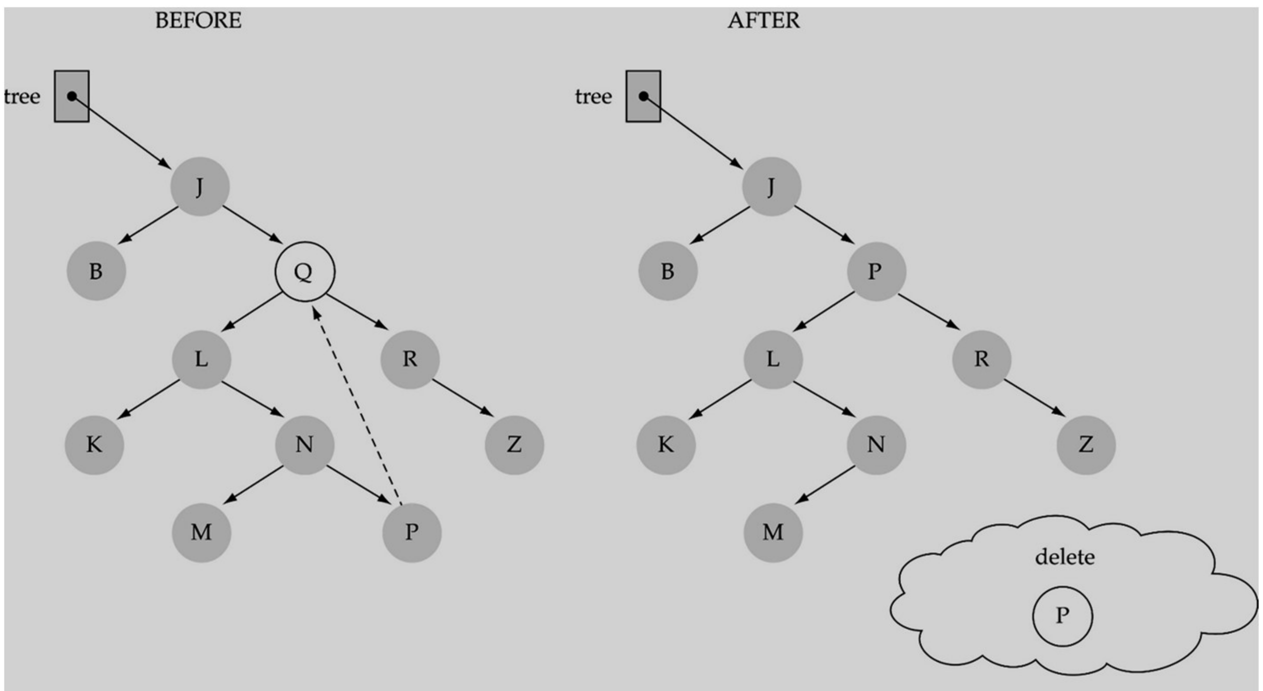


(2) Deleting a node with only one child



Delete the node containing R

(3) Deleting a node with two children



Delete the node containing Q

BST Operations: Delete

- *Why will case 2 always go to case 0 or case 1?*
- A: because when x has 2 children, its successor is the minimum in its right subtree