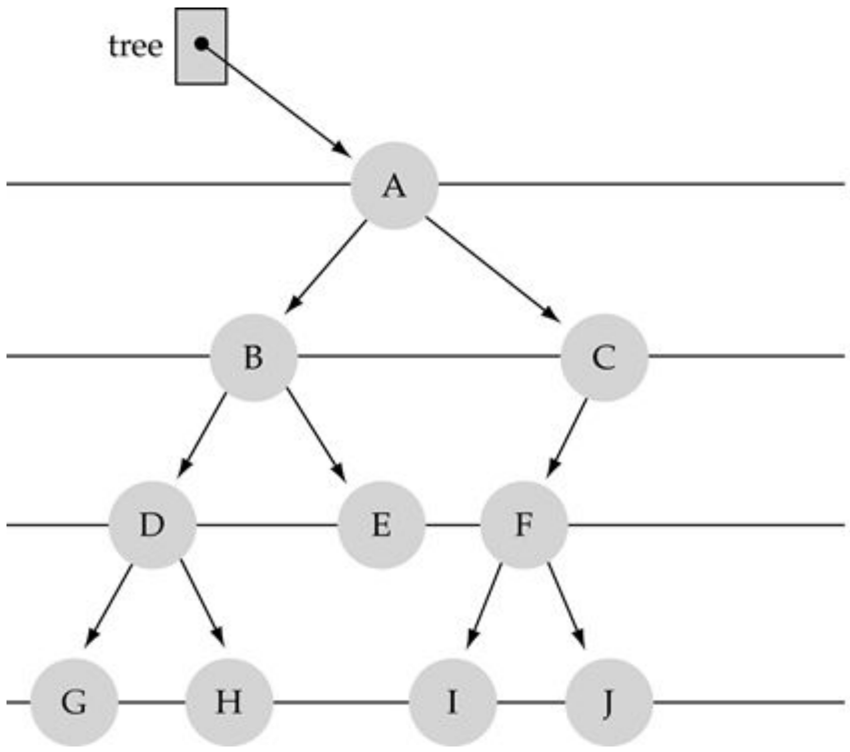
Binary Search Tree and AVL Tree

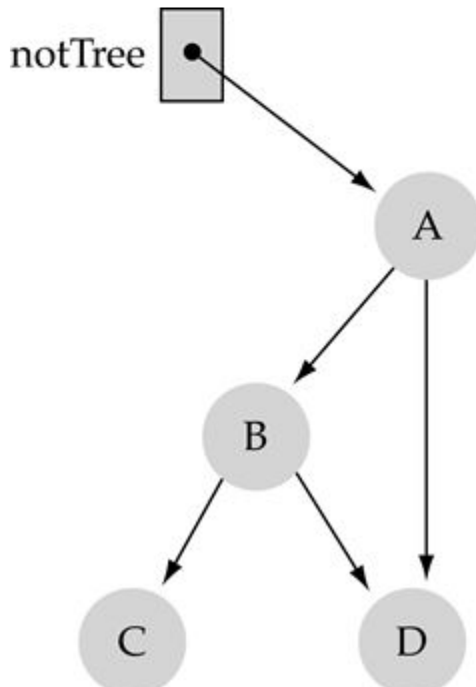
What is a binary tree?

Property 1: each node can have up to two successor nodes.



What is a binary tree? (cont.)

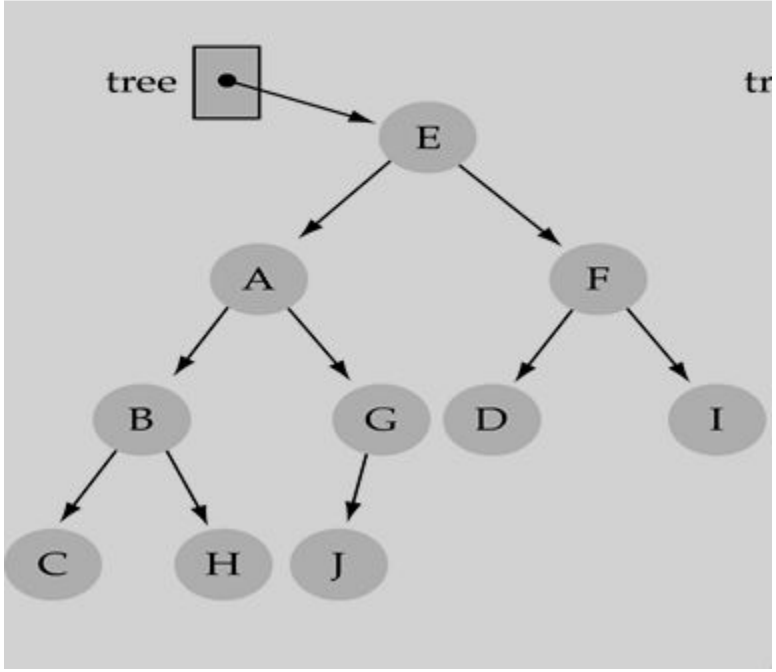
Property 2: a unique path exists from the root to every other node



Not a valid binary tree!

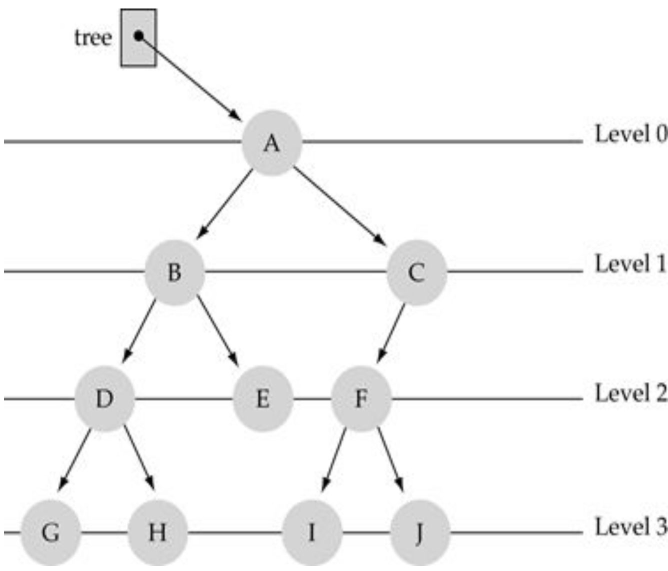
Some terminology

- The successor nodes of a node are called its children
- The predecessor node of a node is called its parent
- The "beginning" node is called the root (has no parent)
- A node without children is called a leaf



Some terminology (cont'd)

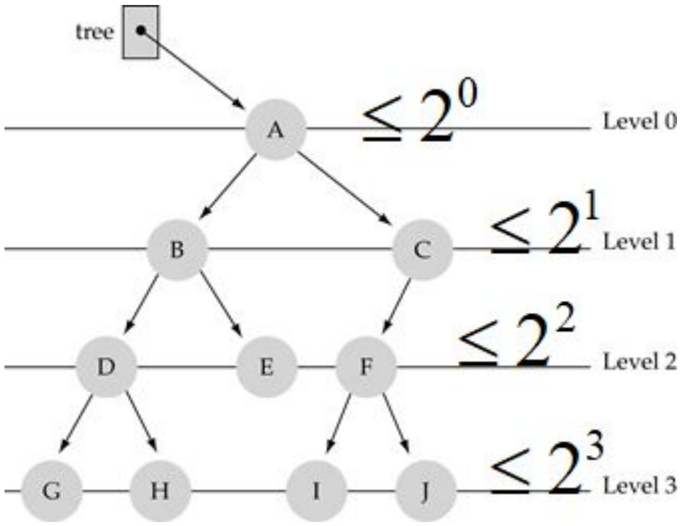
- Nodes are organized in levels (indexed from 0).
- Level (or depth) of a node: number of edges in the path from the root to that node.



- Height of a tree h: #levels = L (Warning: some books define h as #levels-1).
- Full tree: every node has exactly
  - two children and all the
  - leaves are on the same level.

What is the max #nodes at some level l?

The max #nodes at level l is 2^l where l=0,1,2, L-1



What is the total #nodes N of a full tree with height h?

$$N = 2^0 + 2^1 + \ldots + 2^{h-1} = 2^h - 1$$

l=0 l=1 l=h-1

using the geometric series:

$$x^0 + x^1 + \ldots + x^{n-1} = \sum_{i=0}^{n-1} x^i = \frac{x^n - 1}{x - 1}$$

What is the height h of a full tree with N nodes?

$$2^h - 1 = N$$

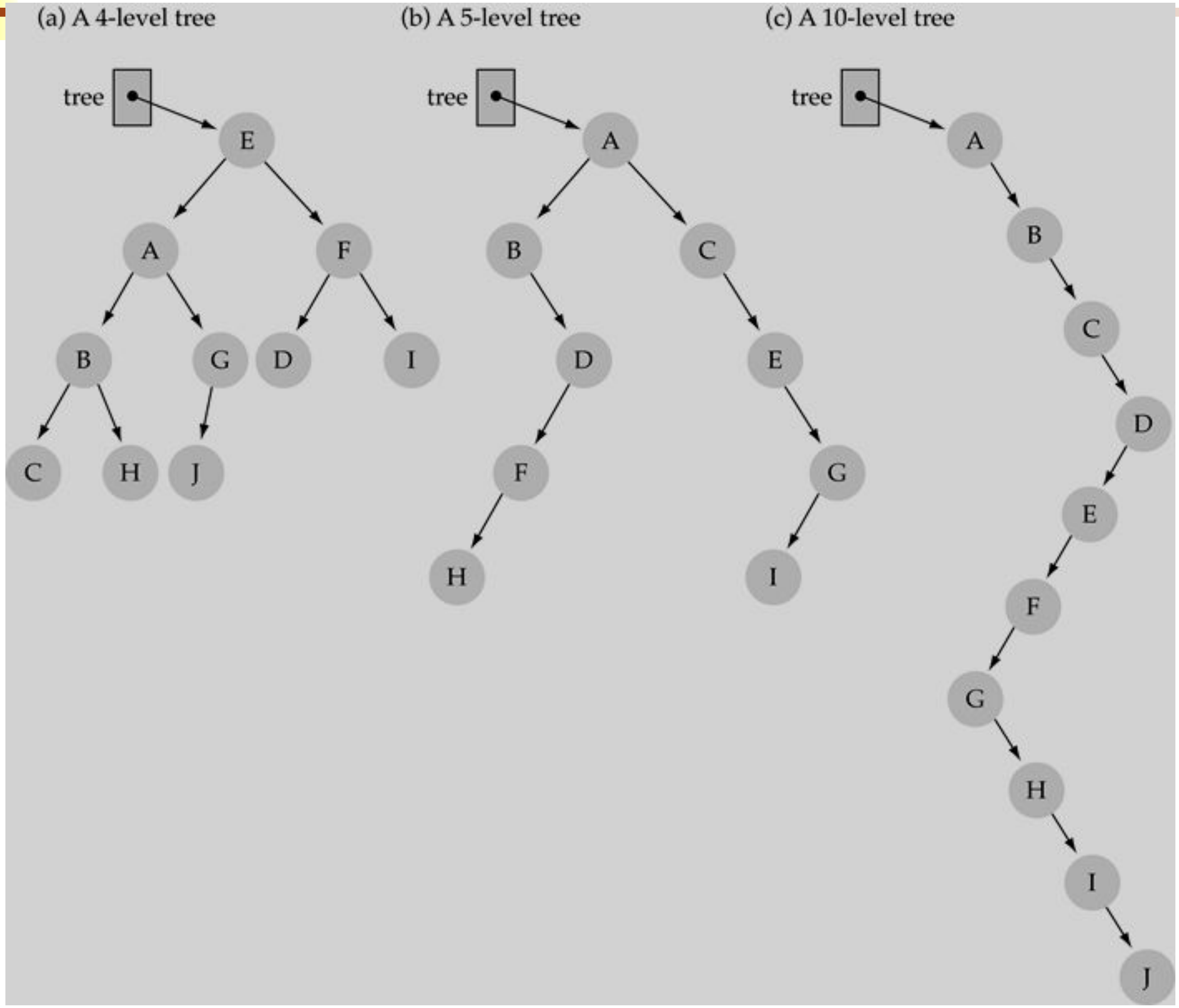
$$\Rightarrow 2^h = N + 1$$

$$\Rightarrow h = \log(N + 1) \rightarrow O(\log N)$$

Why is h important?

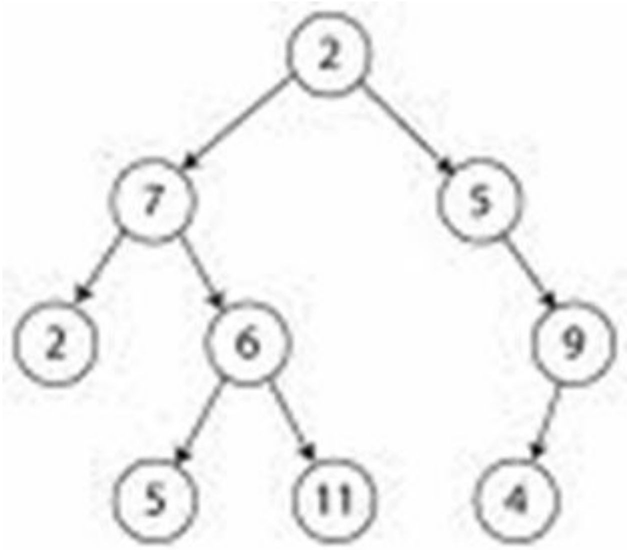
- Tree operations (e.g., insert, delete, retrieve etc.) are typically expressed in terms of h.

- So, h determines running time!


(a) A 4-level tree
(b) A 5-level tree
(c) A 10-level tree
tree
E
A
F
B
G
D
I
C
H
J
tree
A
B
C
D
E
F
G
H
I
tree
A
B
C
D
E
F
G
H
I
J

How to search a binary tree?

(1) Start at the root
(2) Search the tree level by level, until you find the element you are searching for or you reach a leaf.

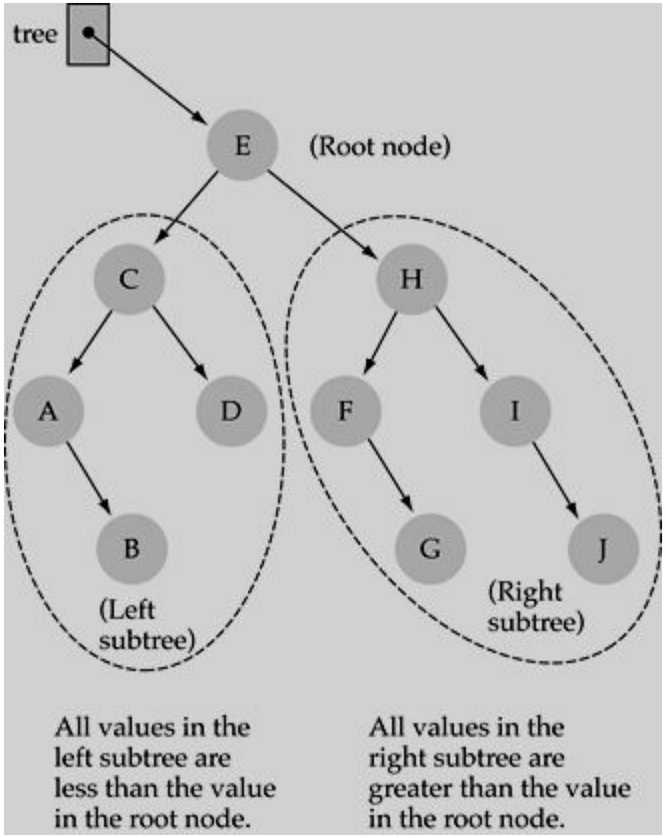


Is this better than searching a linked list?

No → O(N)

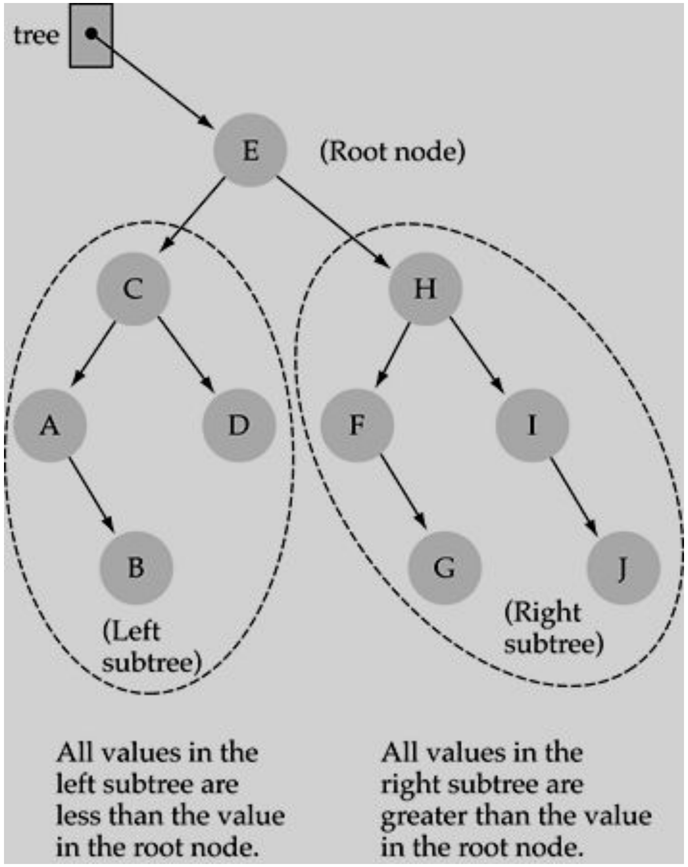
Binary Search Trees (BSTs)

- Binary Search Tree Property:
  The value stored at a node is greater than the value stored at its left child and less than the value stored at its right child



Binary Search Trees (BSTs)

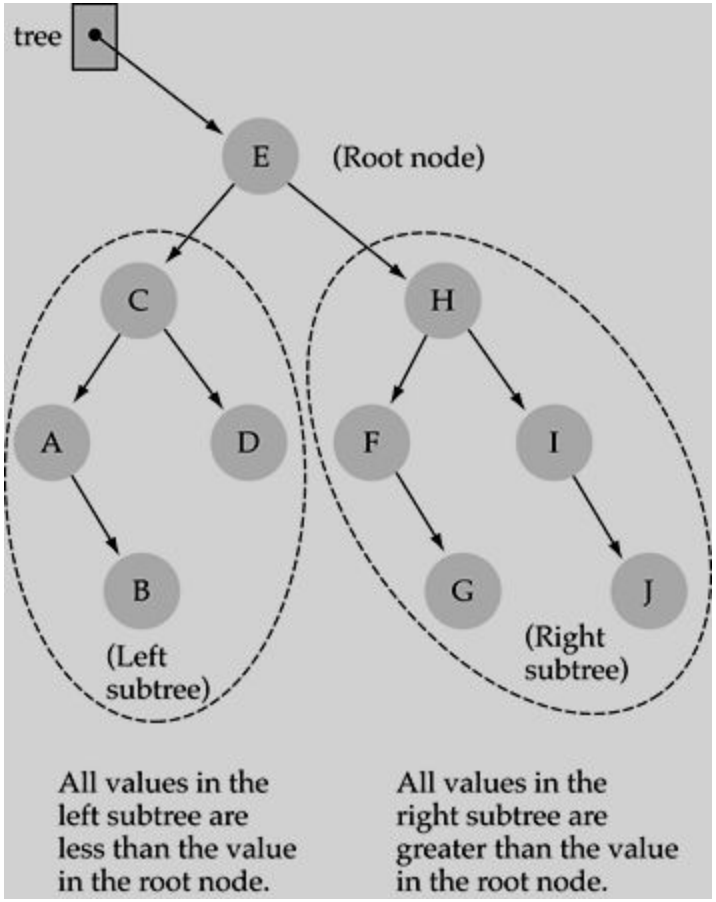
- In a BST, the value stored at the root of a subtree is greater than any value in its left subtree and less than any value in its right subtree!



Binary Search Trees (BSTs)

- Where is the smallest element?
  Ans: **leftmost element**

- Where is the largest element?
  Ans: **rightmost element**

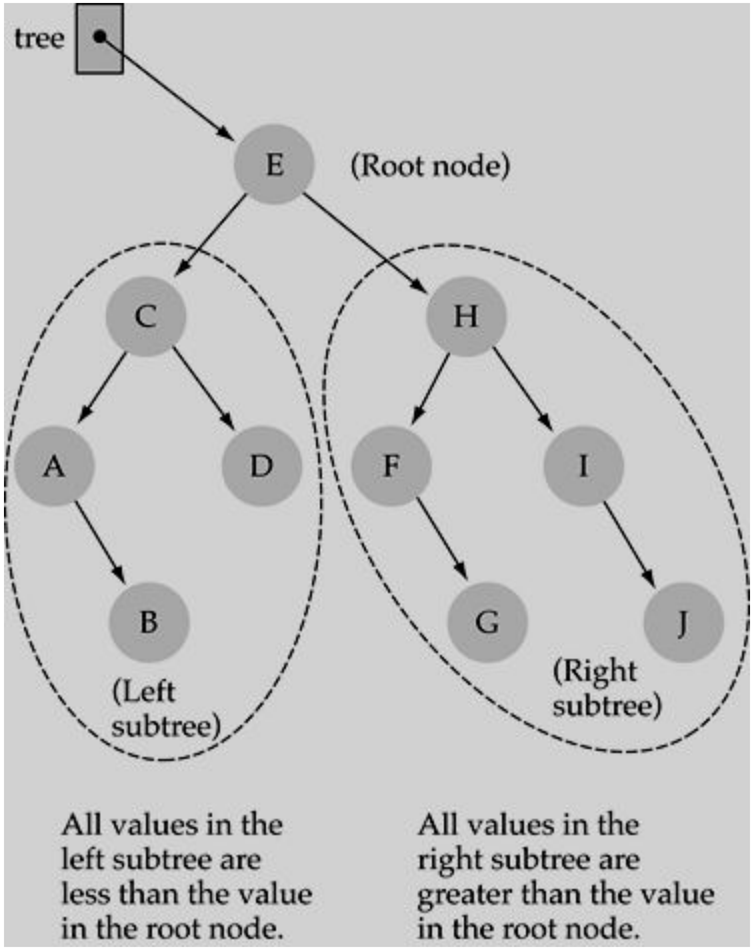


How to search a binary search tree?

(1) Start at the root

(2) Compare the value of the item you are searching for with the value stored at the root

(3) If the values are equal, then item found; otherwise, if it is a leaf node, then not found

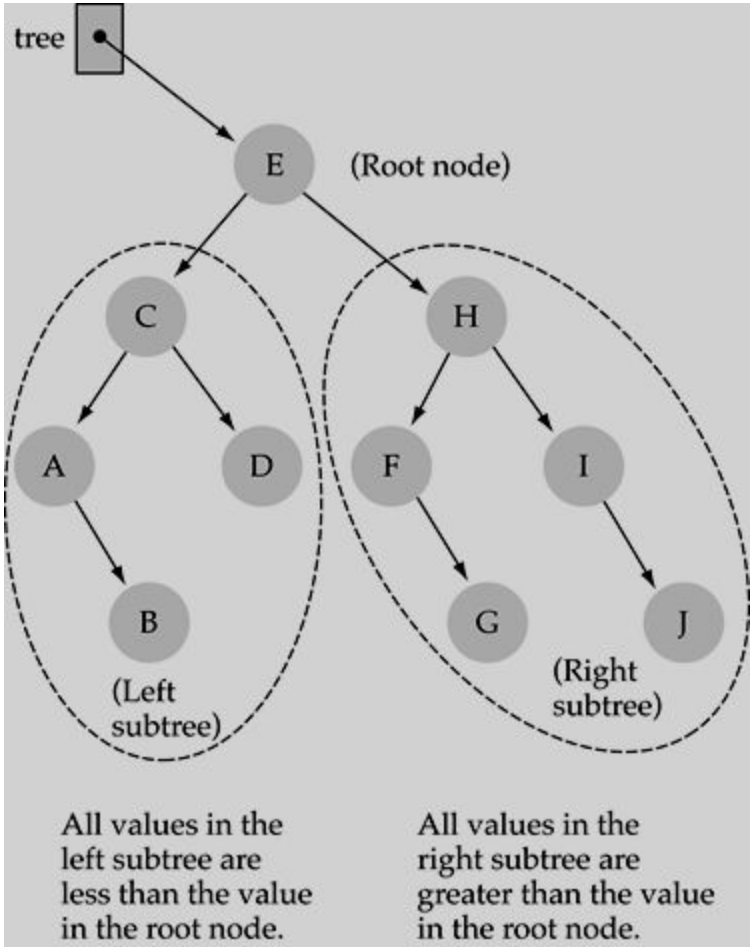


How to search a binary search tree?

(4) If it is less than the value stored at the root, then search the left subtree

(5) If it is greater than the value stored at the root, then search the right subtree

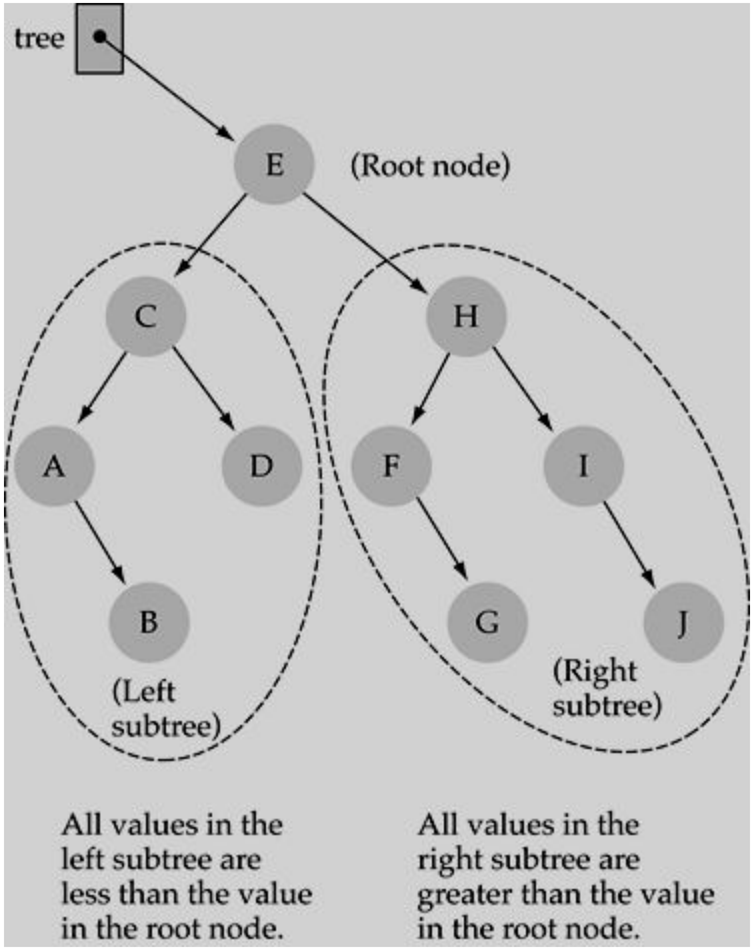
(6) Repeat steps 2-6 for the root of the subtree chosen in the previous step 4 or 5



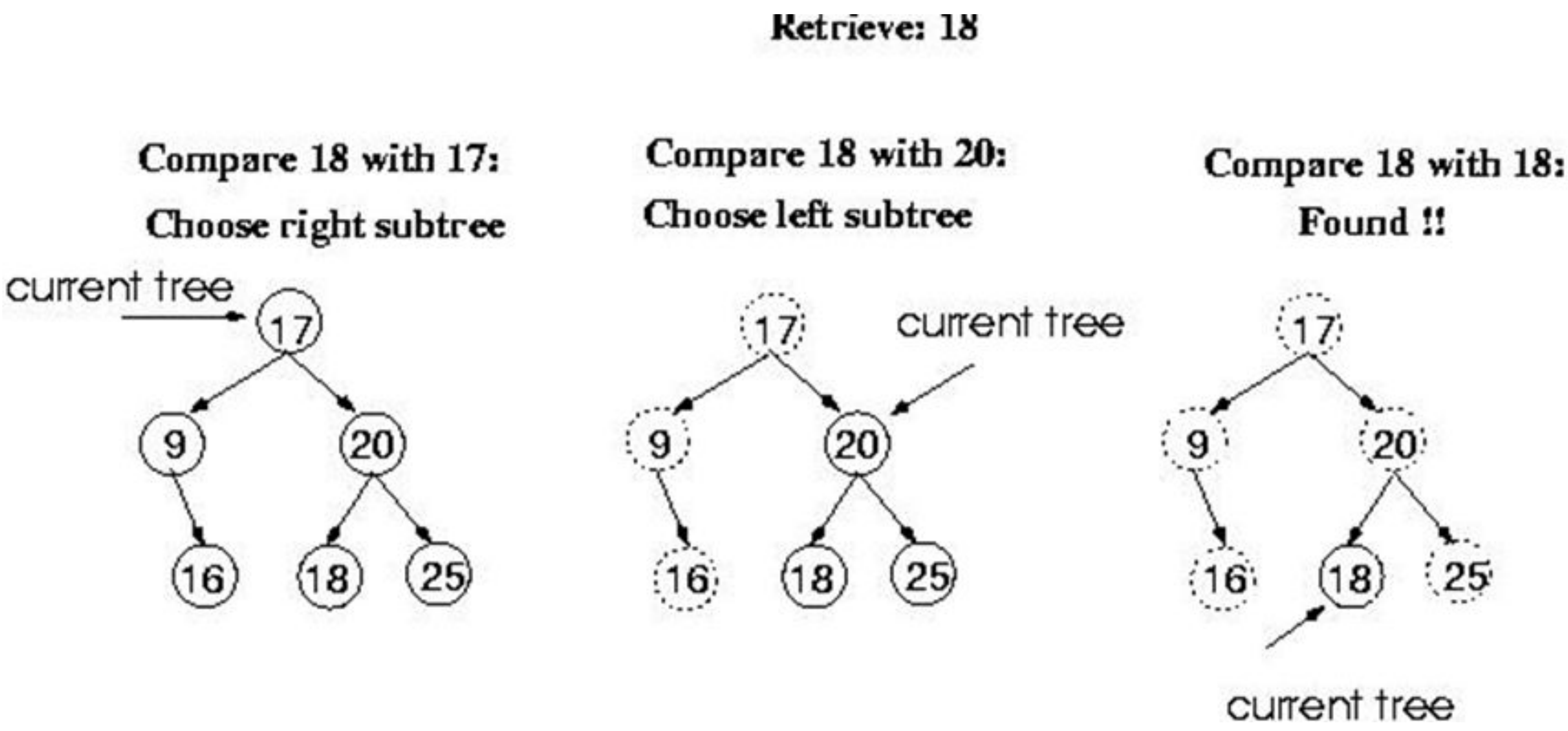
How to search a binary search tree?

How to search a binary search tree?

Yes !! ---> O(logN)

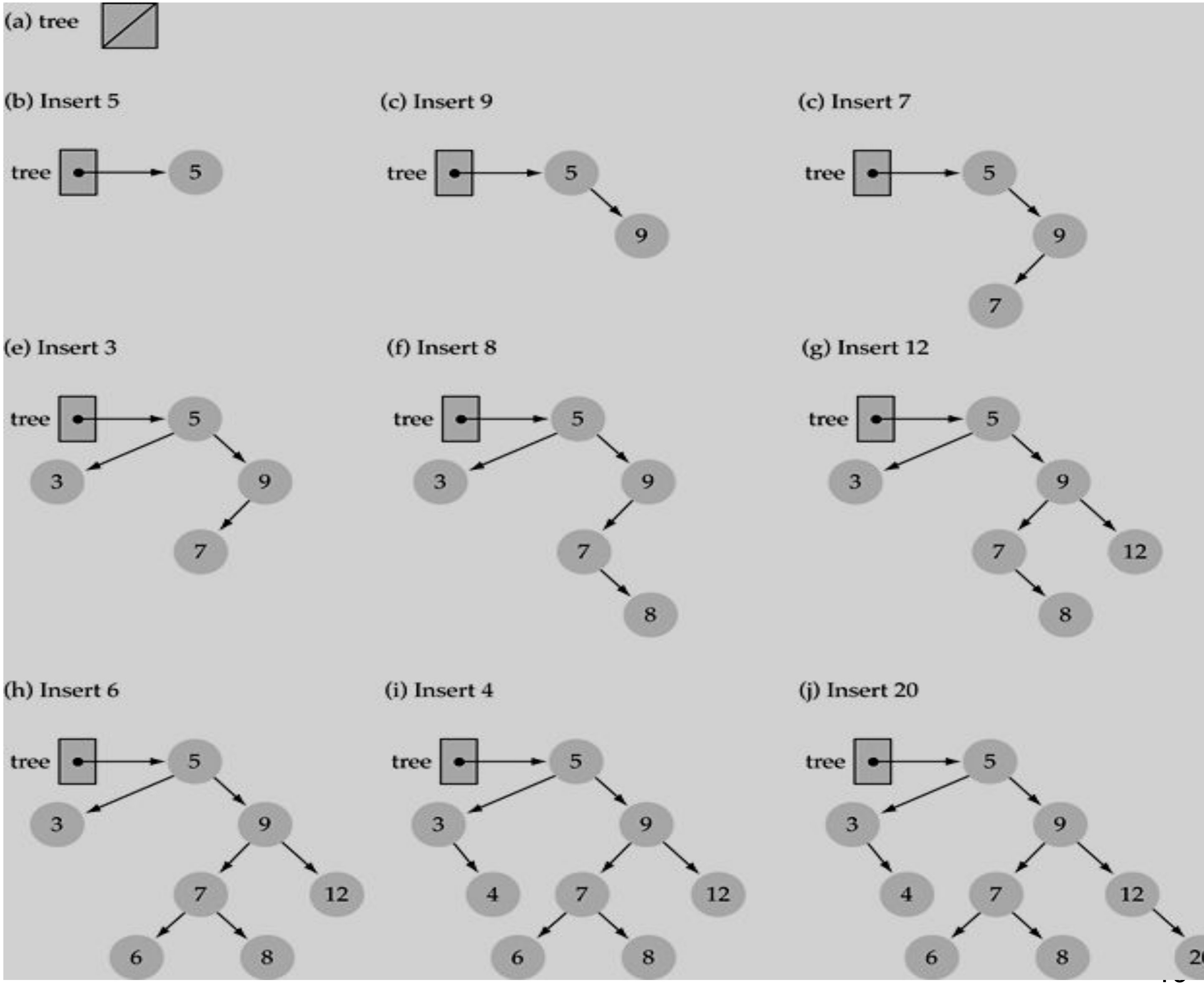


Function Retrieve Item



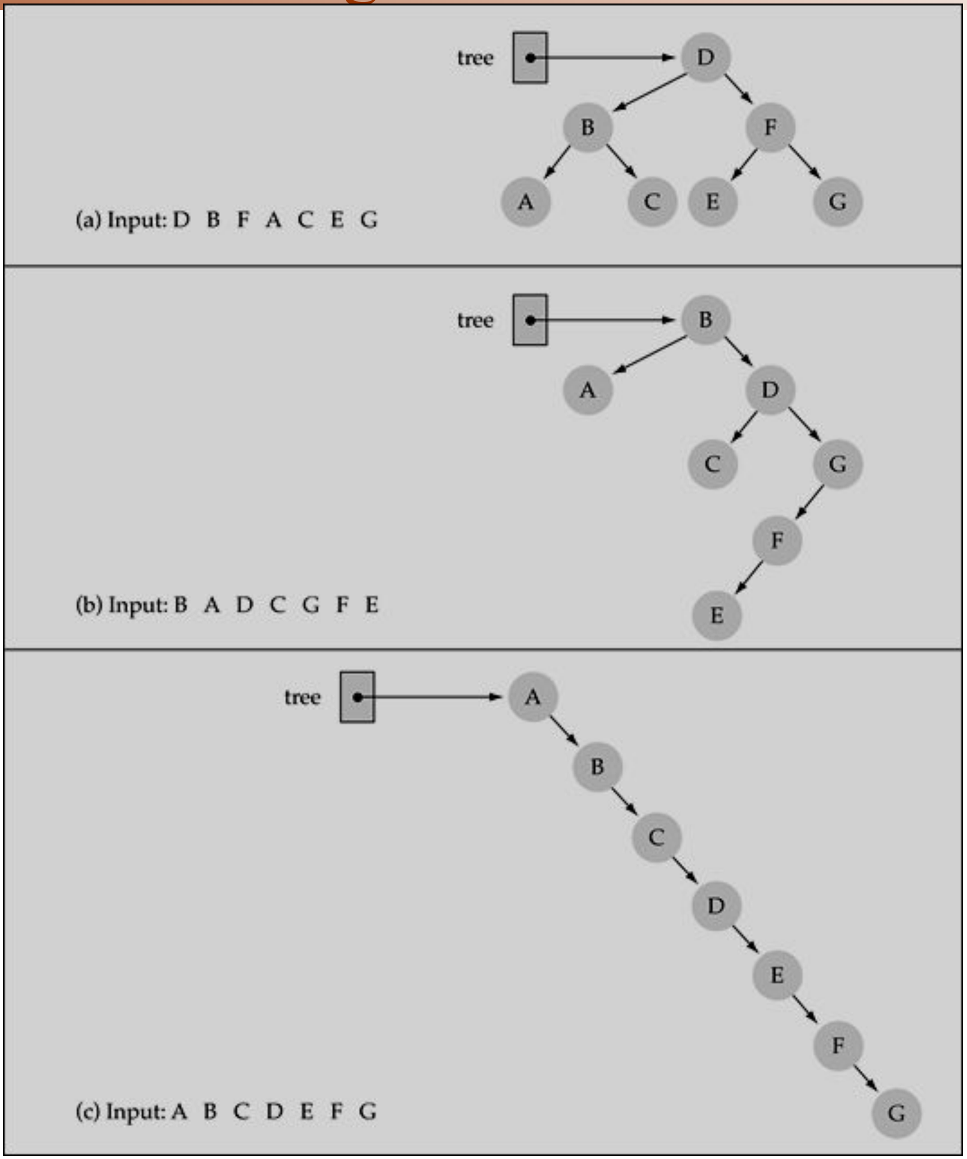
Function Insert Item

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



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'd)

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

Function 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

Deleting leaf node

- Find the node in the given BST
- Simply delete that Node.

Deleting a node with only one child

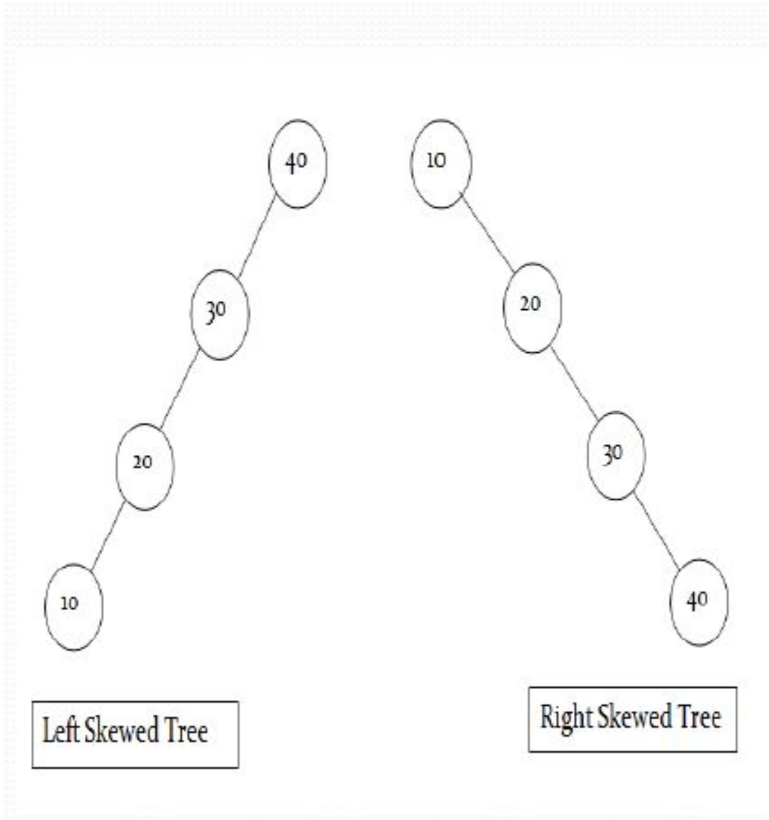
- Find the node in the BST
- Delete this node by connecting grandfather and grandchildren

If the node contains 2-children

- Find the node in the given BST.
- Delete that by replacing inorder successor or predecessor.

Skewed BST

- If a tree which is dominated by left child node or right child node, is said to be a Skewed Binary Tree.
- In a left skewed tree, most of the nodes have the left child without corresponding right child.
- In a right skewed tree, most of the nodes have the right child without corresponding left child.



Limitation of BST

- The average search time for a binary search tree is directly proportional to its height: O(h). Most of the operation average case time is O(log2n).
- BST's are not guaranteed to be balanced. It may be skewed tree also.
- For skewed BST, the average search time becomes O(n). So, it is working like an linear array.
- To improve average search time and make BST balanced, AVL trees are used.

AVL Tree

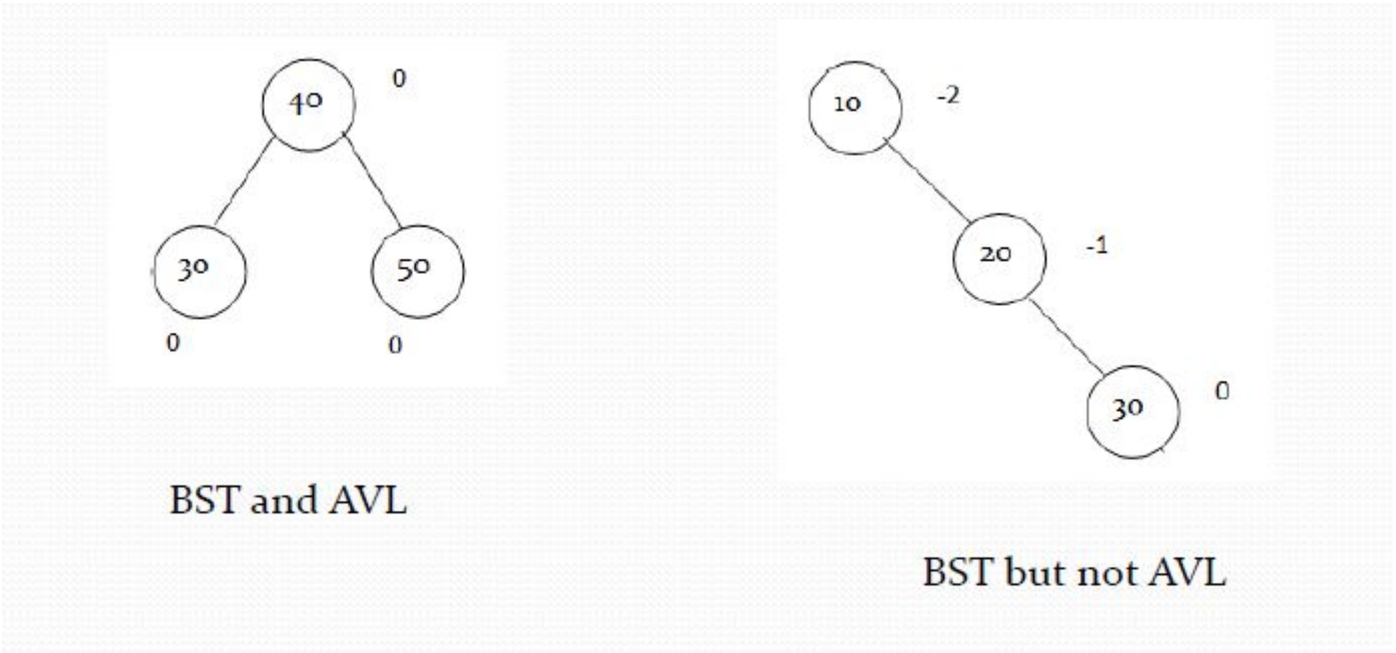
- AVL tree is a height balanced tree.
- It is a self-balancing binary search tree.
- It was invented by Adelson-Velskii and Landis.
- AVL trees have a faster retrieval.
- It takes O(logn) time for insertion and deletion
- operation.
- In AVL tree, difference between heights of left and
- right subtree cannot be more than one for all
- nodes.

AVL Tree

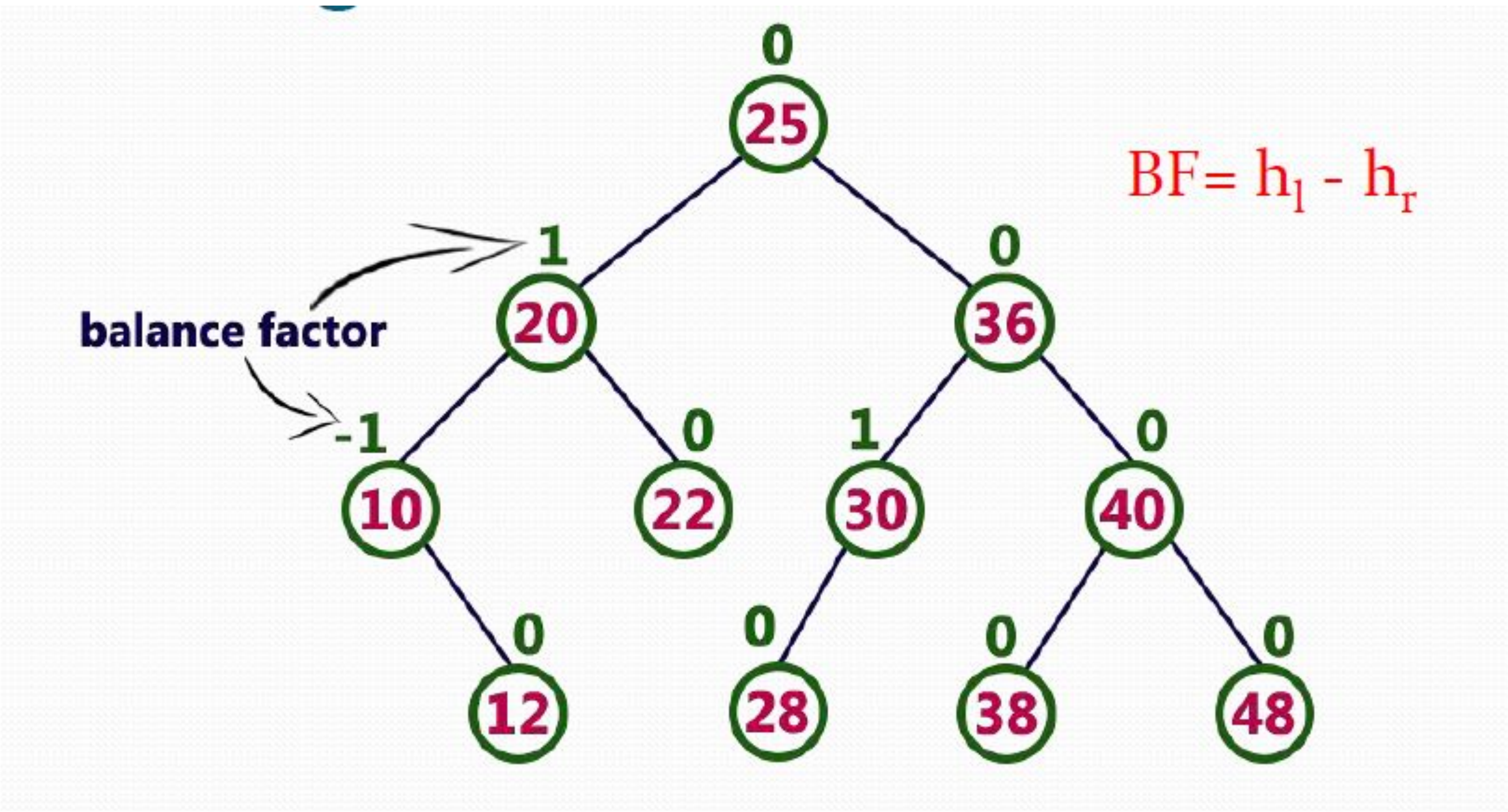
- Balance Factor of node is:
  - Height of left subtree – Height of Right subtree
- Balance Factor is calculated for every node of AVL tree.
- At every node, height of left and right subtree can differ by no more than 1.
- For AVL tree, the possible values of balance factor are -1, 0, 1
- Balance Factor of leaf nodes is 0 (zero).

Example

Every AVL Tree is a binary search tree but all the Binary Search Tree need not to be AVL trees.



Finding Balance Factor



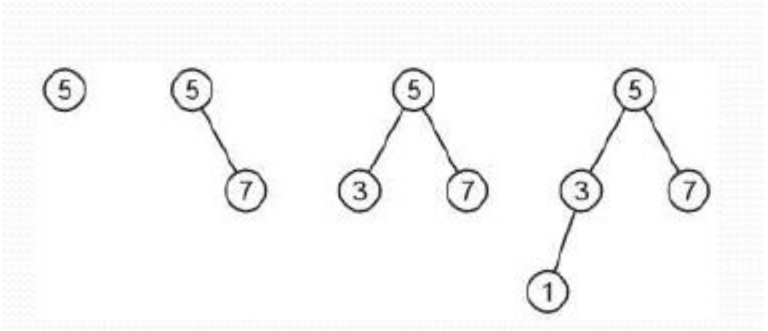
Height of AVL Tree

- By the definition of complete trees, any complete binary search tree is an AVL tree
- Thus, an upper bound on the number of nodes in an AVL tree of height h a perfect binary tree with $2^{h+1} - 1$ nodes.
- What is a lower bound?

Height of AVL Tree

- Let F(h) be the fewest number of nodes in a tree of height h.

- From a previous slide:
  - F(0) = 1
  - F(1) = 2
  - F(2) = 4



- Then what is F(h) in general?

Height of AVL Tree

The worst-case AVL tree of height h would have:

- A worst-case AVL tree of height $h - 1$ on one side,
- A worst-case AVL tree of height $h - 2$ on the other, and
- The root node

$$\boxed{\text{We get: } F(h) = F(h-1) + F(h-2) + 1}$$

This is a recurrence relation:

$$F(h) = \begin{cases} 1 & h = 0 \\ 2 & h = 1 \\ F(h-1) + F(h-2) + 1 & h > 1 \end{cases}$$

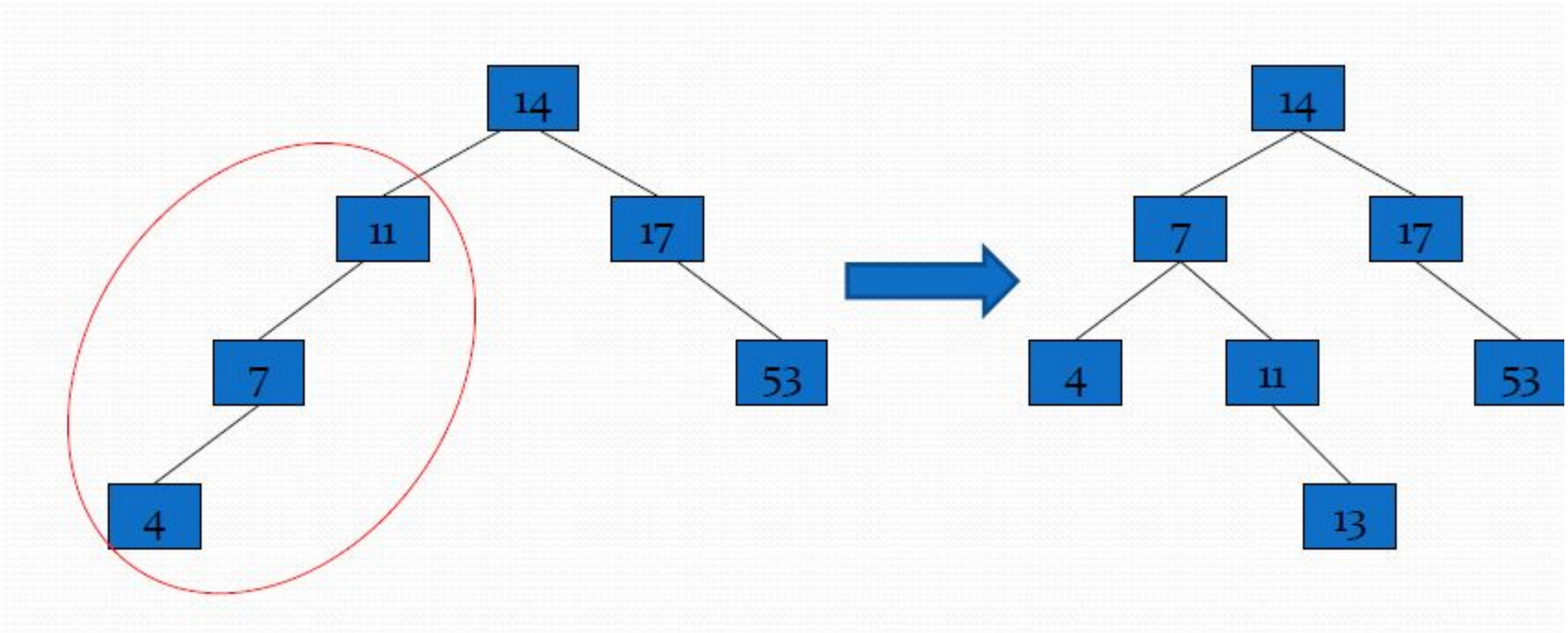
Imbalance

After an insertion, when the balance factor of node A is –2 or 2, the node A is one of the following four imbalance types

1. LL: new node is in the left subtree of the left subtree of A
1. LR: new node is in the right subtree of the left subtree of A
1. RR: new node is in the right subtree of the right subtree of A
1. RL: new node is in the left subtree of the right subtree of A

AVL Tree Example

- Insert 14, 17, 11, 7, 53, 4, 13 into an empty AVL tree



Types of Rotation

Rotation- To switch children and parents among two or three adjacent nodes to restore balance of a tree.