#### كلية علوم الحاسوب والرياضيات College of Computer Science & Mathematics



# Software Engineering Dept.- Second year

Tree Data Structure

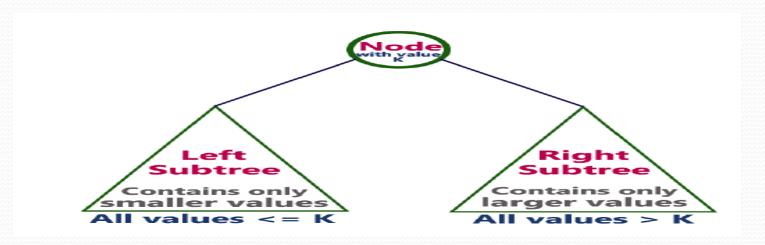
by

Dr.lalheelb M. Ilbralhim



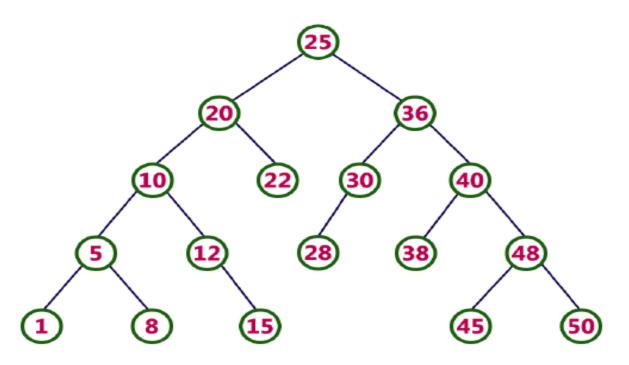
# Binary Search Tree

Binary Search Tree is a binary tree in which every node contains only smaller values in its left subtree and only larger values in its right subtree.



### Example

The following tree is a Binary Search Tree. In this tree, left subtree of every node contains nodes with smaller values and right subtree of every node contains larger values.



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

# **Operations on a Binary Search Tree**

• The following operations are performed on a binary search tree.

- 1. Search
- 2. Insertion
- 3. Deletion

# **Search Operation in BST**

The search operation is performed as follows...

**Step 1:** Read the search element from the user

**Step 2:** Compare, the search element with the value of root node in the tree.

**Step 3:** If both are matching, then display "Given node found!!!" and terminate the function

**Step 4:** If both are not matching, then check whether search element is smaller or larger than that node value.

**Step 5:** If search element is smaller, then continue the search process in left subtree.

**Step 6:** If search element is larger, then continue the search process in right subtree.

**Step 7:** Repeat the same until we found exact element or we completed with a leaf node

**Step 8:** If we reach to the node with search value, then display "Element is found" and terminate the function.

**Step 9:** If we reach to a leaf node and it is also not matching, then display "Element not found" and terminate the function.

## **Insertion Operation in BST**

In binary search tree, new node is always inserted as a leaf node. The insertion operation is performed as follows...

**Step 1:** Create a newNode with given value and set its **left** and **right** to **NULL**.

**Step 2:** Check whether tree is Empty.

**Step 3:** If the tree is **Empty**, then set **root** to **newNode**.

**Step 4:** If the tree is **Not Empty**, then check whether value of newNode is **smaller** or **larger** than the node (here it is root node).

**Step 5:** If newNode is **smaller** than **or equal** to the node, then move to its **left** child. If newNode is **larger** than the node, then move to its **right** child.

**Step 6:** Repeat the above step until we reach to a **leaf** node (e.i., reach to NULL).

**Step 7:** After reaching a leaf node, then isert the newNode as **left child** if newNode is **smaller or equal** to that leaf else insert it as **right child**.

#### **Deletion Operation in BST**

Deleting a node from Binary search tree has following **three cases.** 

- Case 1: Deleting a Leaf node (A node with no children)
- Case 2: Deleting a node with one child
- Case 3: Deleting a node with two children

#### Case 1: Deleting a leaf node

We use the following steps to delete a leaf node from BST...

**Step 1: Find** the node to be deleted using **search operation** 

**Step 2:** Delete the node using **free** function (If it is a leaf) and terminate the function.

### Case 2: Deleting a node with one child

We use the following steps to delete a node with one child from BST.

**Step 1: Find** the node to be deleted using **search operation** 

**Step 2:** If it has only one child, then create a link between its parent and child nodes.

**Step 3:** Delete the node using **free** function and terminate the function.

#### Case 3: Deleting a node with two children

We use the following steps to delete a node with two children from BST...

**Step 1: Find** the node to be deleted using **search operation** 

**Step 2:** If it has two children, then find the **largest** node in its **left subtree** (OR) the **smallest** node in its **right subtree**.

**Step 3: Swap** both **deleting node** and node which found in above step.

**Step 4:** Then, check whether deleting node came to **case 1** or **case 2** else goto steps 2

**Step 5:** If it comes to **case 1**, then delete using case 1 logic.

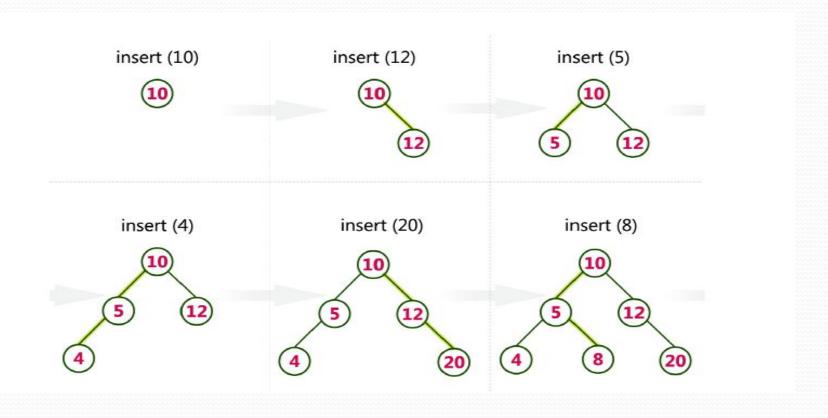
**Step 6:** If it comes to **case 2**, then delete using case 2 logic.

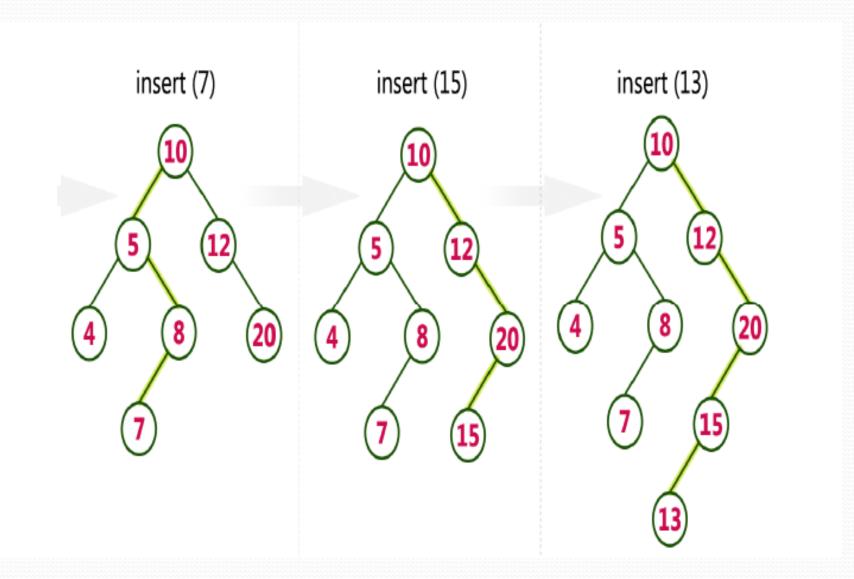
**Step 7:** Repeat the same process until node is deleted from the tree.

**Example** Construct a Binary Search Tree by inserting the following sequence of numbers...

• 10,12,5,4,20,8,7,15 and 13

Above elements are inserted into a Binary Search Tree as follows.





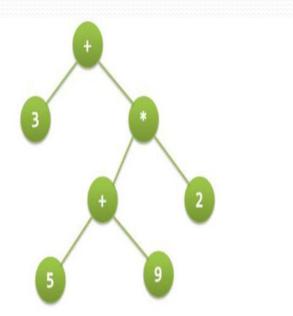
## **Expression Tree**

Expression tree is a binary tree in which each internal node corresponds to <u>operator</u> and each leaf node corresponds to <u>operand</u>.

example expression tree for 3 + ((5 + 9) \* 2)

would be:

Inorder traverse

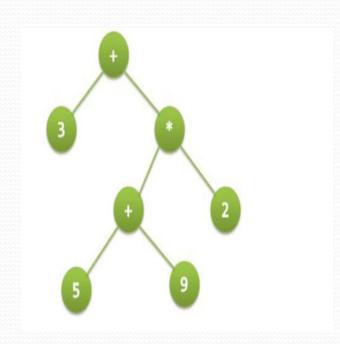


• Inorder traversal of expression tree produces infix version of given postfix expression (same with preorder traversal it gives prefix expression).

#### **H.W**<sub>1</sub>

You have the following tree ,Write the

- 1- in-order traversal
- 2- pre-order traversal
- 3- post-order traversal



#### HW2

You have the following Arithmetic Expression

$$A + B / C \wedge (V - M) * 4$$

,Write the

- 1- in-order traversal
- 2- pre-order traversal
- 3- post-order traversal