Binary Search Tree (BST)

Prof. Harish D.G.
Dept. of Computer and IT
College of Engineering,Pune(COEP)
www.harishgadade.com

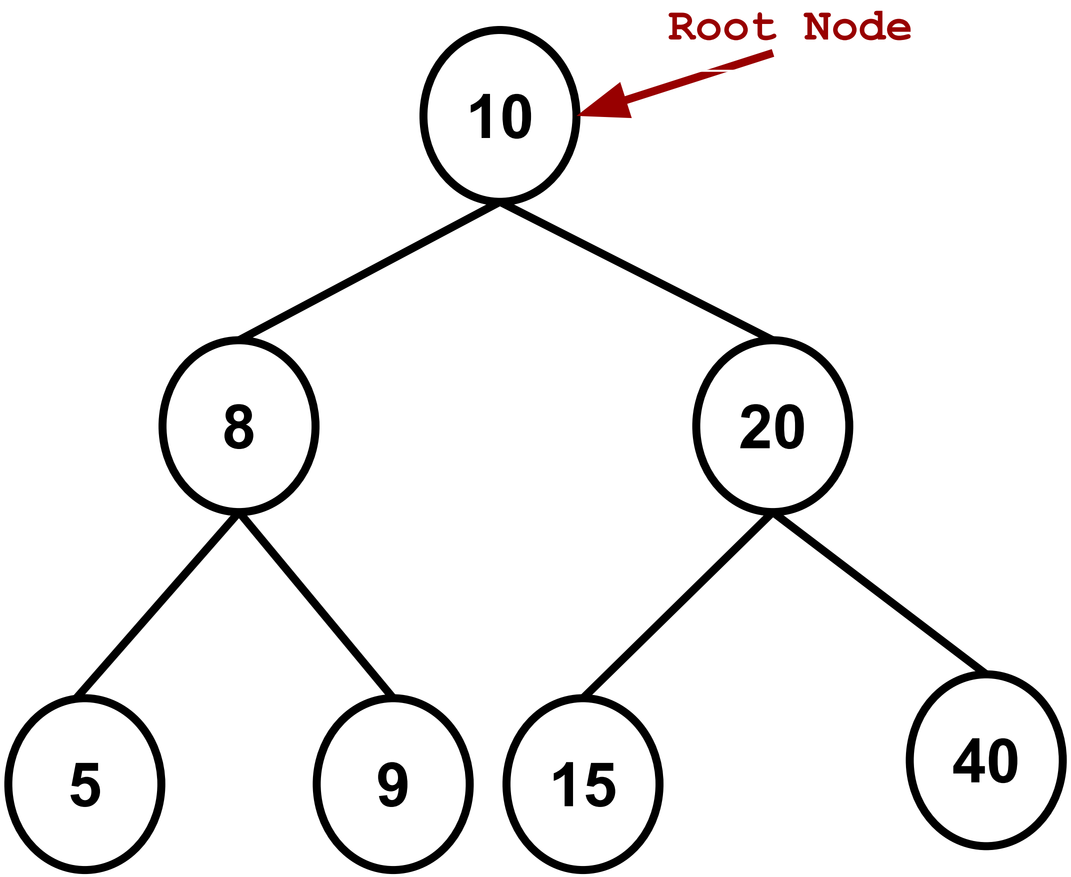
Binary Search Tree

Definition:

A binary search tree is binary tree, which can be either empty or non-empty. If it is non-empty then it satisfies the following properties.

1. Every element has a key and no two elements have the same key. i.e. the keys are distinct.
2. The keys in the left subtree are smaller than the key in the root.
3. The keys in the right subtree are larger than the key in the root.
4. The left and right subtrees are also Binary search tree.

Binary Search Tree



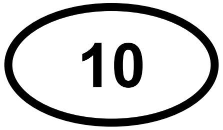
- Create the Binary Search Tree for the following

  a) 10, 8, 20, 5, 9, 15, 30
  b) 10, 20, 15, 5, 18
  c) 5, 2, 8, 4, 1, 9

- Search any keys (e.g. 10 , 1, 15 etc) from the above trees

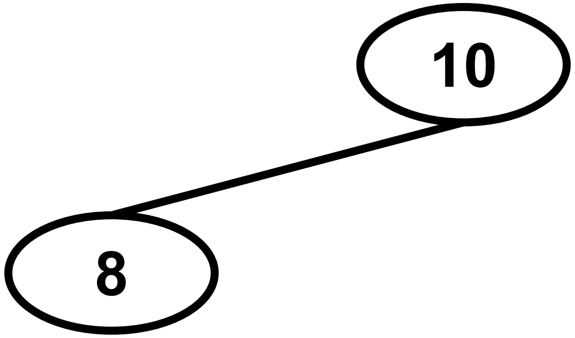
e.g. (a) 10, 8, 20, 5, 9, 15, 30

Insert : 10



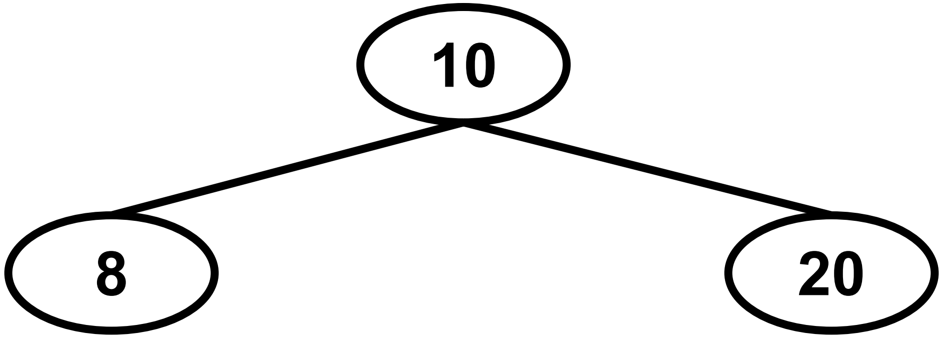
e.g. (a) 10, 8, 20, 5, 9, 15, 30

Insert : 8



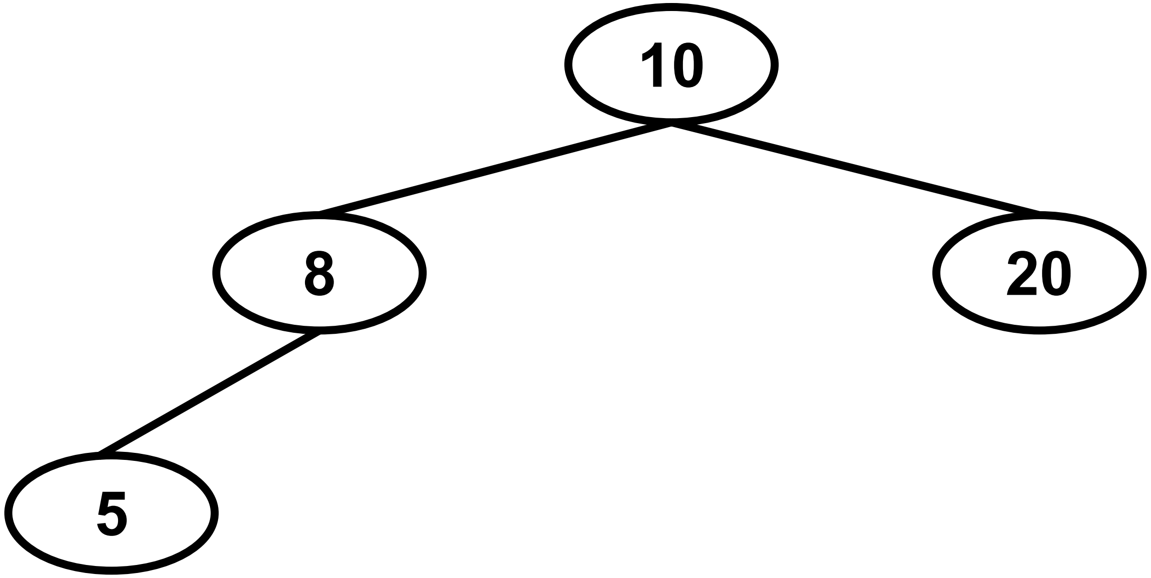
e.g. (a) 10, 8, 20, 5, 9, 15, 30

Insert : 20



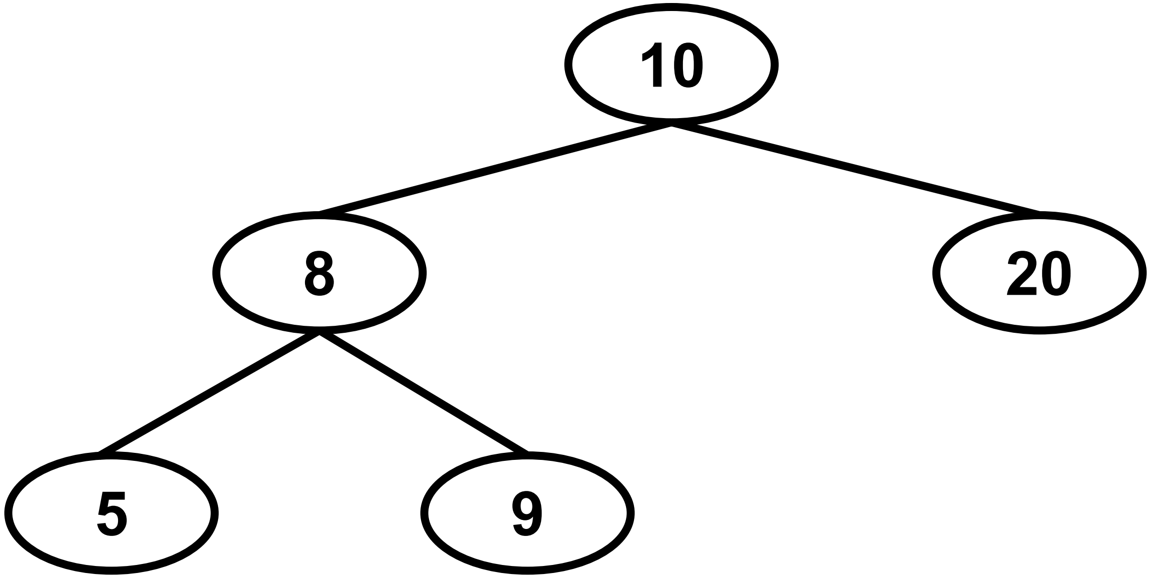
e.g. (a) 10, 8, 20, 5, 9, 15, 30

Insert : 5



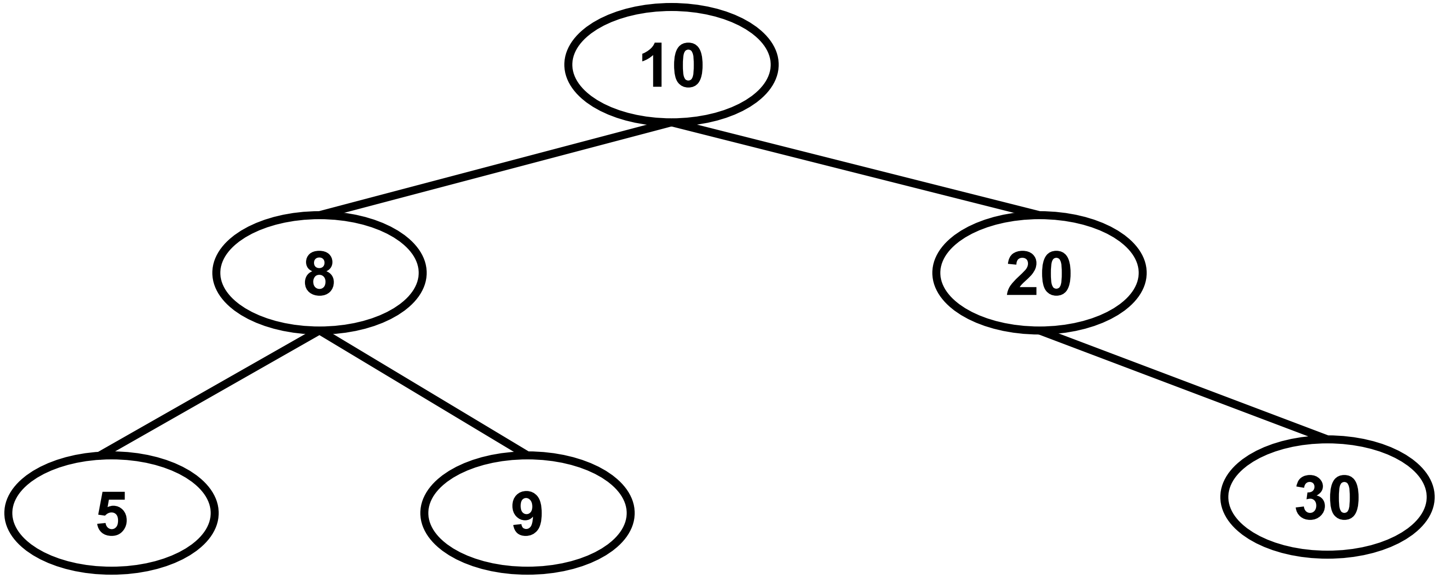
e.g. (a) 10, 8, 20, 5, 9, 15, 30

Insert : 9



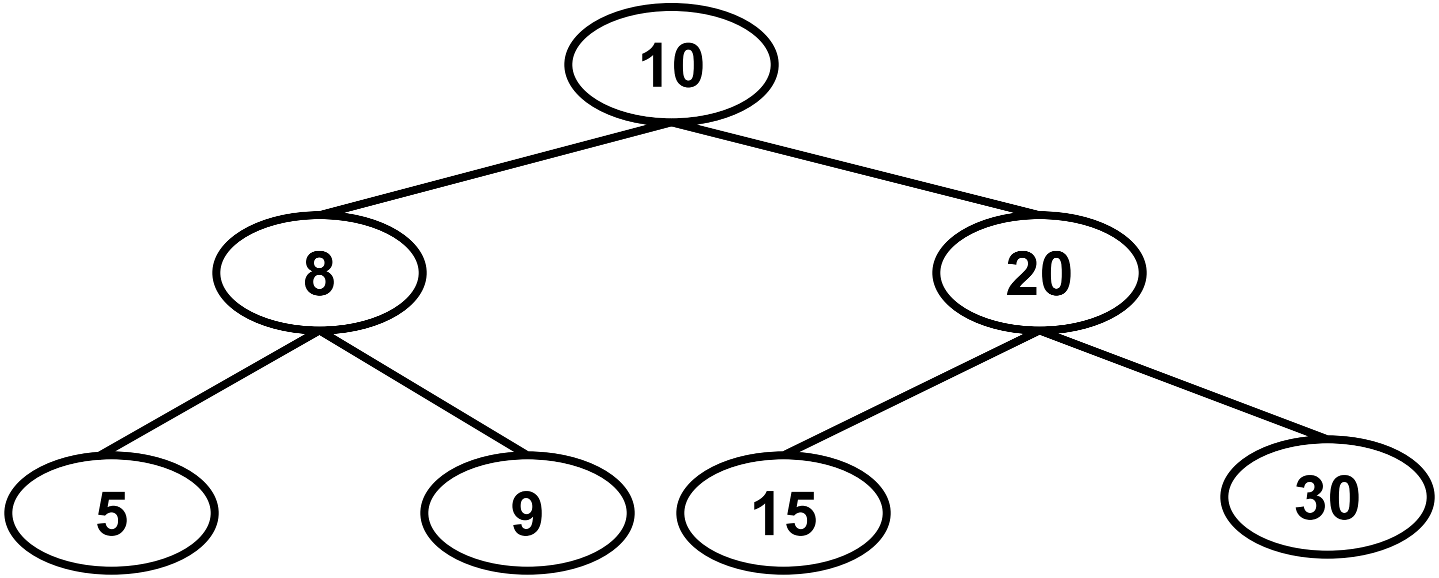
e.g. (a) 10, 8, 20, 5, 9, 15,30

Insert : 30



e.g. (a) 10, 8, 20, 5, 9, 15, 30

Insert : 15



Final Binary Search Tree

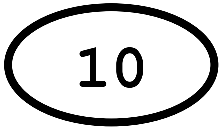
- Create the Binary Search Tree for the following

  a) 10, 8, 20, 5, 9, 15, 30
  b) 10, 20, 15, 5, 18
  c) 5, 2, 8, 4, 1, 9

- Search any keys (e.g. 10 , 1, 15 etc) from the above trees

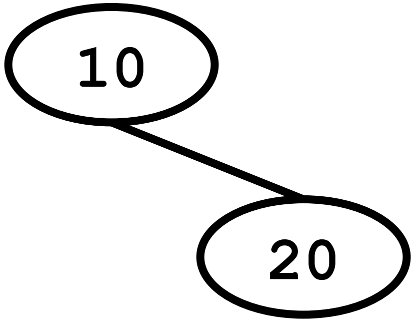
e.g. (b) 10, 20, 15, 5, 18

Insert : 10



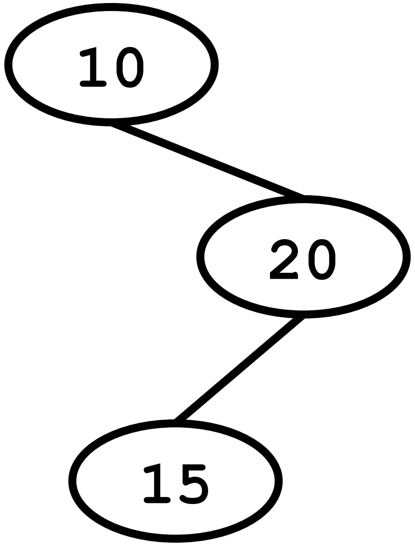
e.g. (b) 10, 20, 15, 5, 18

Insert : 20



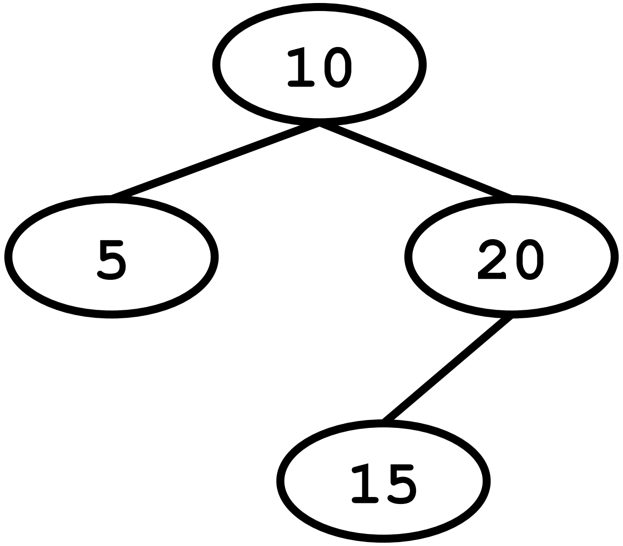
e.g. (b) 10, 20, 15, 5, 18

Insert : 15



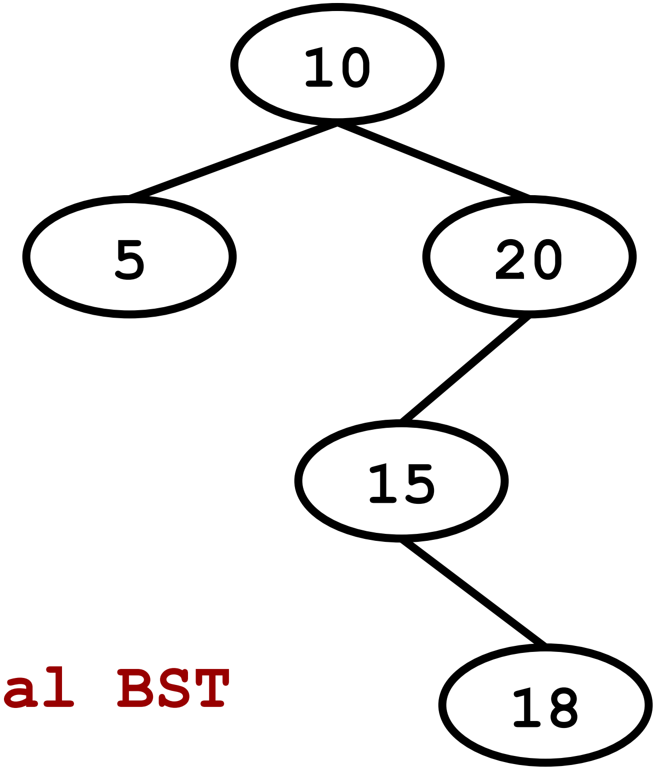
e.g. (b) 10, 20, 15, 5, 18

Insert : 5



e.g. (b) 10, 20, 15, 5, 18

Insert : 18



Final BST

Binary Tree Traversals

1. Binary Tree Traversals
    - Inorder Traversal
    - Preorder Traversal
    - Postorder Traversal

Binary Tree Traversals

1. Binary Tree Traversals
   - Inorder Traversal
   - Preorder Traversal
   - Postorder Traversal

Inorder Algorithm (LVR)

1. Traverse the left sub-tree in In-order
2. Visit the root
3. Traverse the Right sub-tree in In-order

Binary Tree Traversals

1. Binary Tree Traversals
   - Inorder Traversal
   - Preorder Traversal
   - Postorder Traversal

Preorder Algorithm (VLR)

1. Visit the root
2. Traverse the Left sub-tree in preorder
3. Traverse the Right sub-tree in preorder

Binary Tree Traversals

1. Binary Tree Traversals

   - Inorder Traversal

   - Preorder Traversal

   - Postorder Traversal

Postorder Algorithm (LRV)

1. Traverse the Left sub-tree in postorder
2. Traverse the Right sub-tree in postorder
3. Visit the root