Lab 10: Binary Search Tree (BST)

CLO:

02,03

Objectives:

In this lab you will learn about

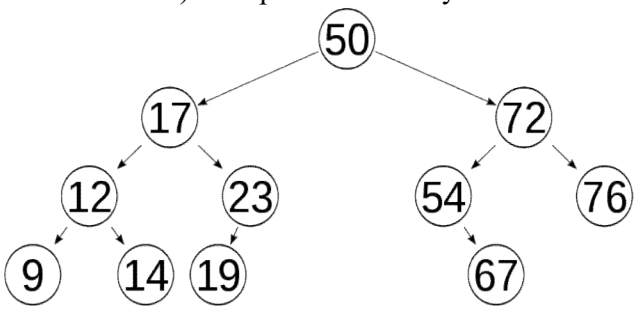
- Implementing binary search trees.
- Basic functions that can be performed on BST.

Binary Search Tree (BST):

Binary Search Tree is a node-based binary tree data structure which has the following properties:

- The left subtree of a node contains only nodes with keys lesser than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.
- The left and right subtree each must also be a binary search tree.

A binary tree is made of nodes, where each node contains a "left" pointer, a "right" pointer, and a data element. The "root" pointer points to the topmost node in the tree. The left and right pointers recursively point to smaller "subtrees" on either side. A null pointer represents a binary tree with no elements -- the empty tree. The formal recursive definition is: a binary tree is either empty (represented by a null pointer), or is made of a single node, where the left and right pointers (recursive definition ahead) each point to a binary tree.



Basic Operations

Following are the basic operations of a tree −

1. Search − Searches an element in a tree.
2. Insert − Inserts an element in a tree.
3. Pre-order Traversal − Traverses a tree in a pre-order manner.
4. In-order Traversal − Traverses a tree in an in-order manner.
5. Post-order Traversal − Traverses a tree in a post-order manner.

Search Operation: Whenever an element is to be searched, start searching from the root node. Then if the data is less than the key value, search for the element in the left subtree. Otherwise, search for the element in the right subtree. Follow the same algorithm for each node.

Insert Operation: Whenever an element is to be inserted, first locate its proper location. Start searching from the root node, then if the data is less than the key value, search for the empty location in the left subtree and insert the data. Otherwise, search for the empty location in the

right subtree and insert the data.

Sample Code: (Structure of Node in a BST)

```
private class Node
{
   Key key;
   Value val;
   Node left, right;
}
The code for searching in a BST
public Value get(Key key)
{
   Node x = root;
   while (x != null)
   {
      int cmp = key.compareTo(x.key);
      if (cmp == 0)
         return x.val;
      else if (cmp < 0)
         x = x.left;
      else if (cmp > 0)
         x = x.right;
   } return null;
}
```

Lab Tasks

Create your own class of Binary Tree which will have the following functions.

1. Function called Insert to insert a node in Tree.
2. Function called Search(key) to search a node in Tree.
3. Function called Display to display the content of Tree.
   Pre-Order()
   In-Order()
   Post-Order()
4. Function called Max to find the maximum value in Tree.
5. Function called Max to find the maximum value in Tree.
6. Function called Delete(key) to delete a node in Tree.