Lab 07

Binary Search Tree (BST)

Objective:

This lab will introduce you the concept of BST data structure

Activity Outcomes:

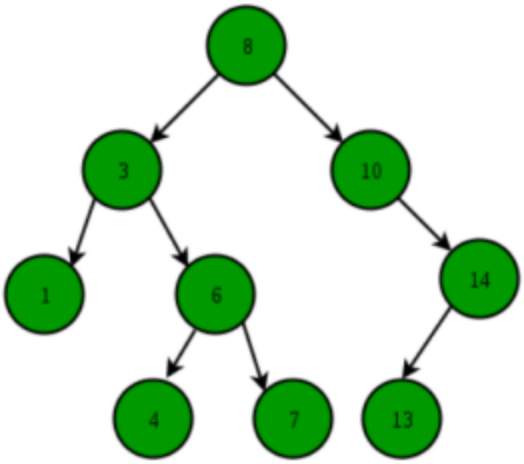
This lab teaches you the following topics:

- How to code BST as a special case of Binary Tree
- BST Traversals, PreOrder, InOrder and PostOrder
- Searching in BST

1) Useful Concepts

A 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.



The main operations are:

- Insert in BST − place an element in the existing structure of BST.
- Search in BST – check about presence of an element in BST, if found returns number of comparisons done for a successful search.
- Delete from BST − delete the element from BST
- Display BST/ Traversal – displays all the nodes in one of the three possible traversals.

Activity 1:

Insertion in BST

```
static void insert(TreeNode<Integer> root,int info) {
          TreeNode<Integer> node = new TreeNode<Integer>(info);
          TreeNode<Integer> p, q;
          p = q = root;

          while( (info !=
Integer.valueOf(p.getInfo().toString()).intValue()) &&
                              q != null)
          {
               p = q;
               if(info <
(Integer.valueOf(p.getInfo().toString())).intValue())
                    q = p.getLeft();
               else
                    q = p.getRight();
          }//end while end

          if(info ==
Integer.valueOf((p.getInfo()).toString()).intValue())
          {
               System.out.println("attempt to insert duplicate:
"+info);
          }
          else if(info <
Integer.valueOf((p.getInfo()).toString()).intValue())
          {
               p.setLeft(node);
          }
          else {
               p.setRight(node);
          }
     }//end insert
```

Activity 2:

Write down Traversal code for the above activity, perform Pre order, Post order and In order Traversals.

```
static void preorder(TreeNode<Integer> treeNode)
     {
         if( treeNode != null )
         {
             System.out.print( (treeNode.getInfo().toString())+"
");
             preorder(treeNode.getLeft());
             preorder(treeNode.getRight());
         }
     }


static void inorder(TreeNode<Integer> treeNode)
     {
         if( treeNode != null )
         {
             inorder(treeNode.getLeft());
             System.out.print((treeNode.getInfo().toString())+"
");
             inorder(treeNode.getRight());
         }
     }

static void postorder(TreeNode<Integer> treeNode)
     {
         if( treeNode != null )
         {
             postorder(treeNode.getLeft());
             postorder(treeNode.getRight());
             System.out.print ( (treeNode.getInfo().toString())+"
");
         }
     }
```

Task

1. Write down code to print and count Leaf Nodes of a BST
2. Introduce the method to delete a Node from BST, keep in mind that there are three possibilities
    a. Node without any Child
    b. Node with One Child
    c. Node with both the Children
3. Write down the method to count to number of nodes and find the sum of all nodes. (Hint use recursion)
4. Write down the method to find
    a. Minimum value from BST
    b. Maximum value from BST