***Introduction to Data Structure and Algorithm in PYTHON (*TREES*)***

Activities for this lab:

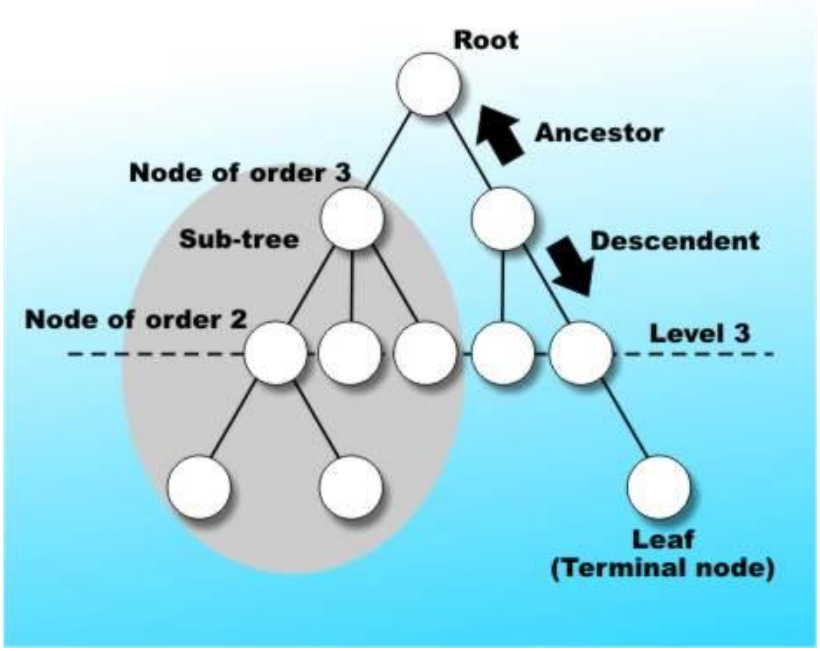
- **Explain the concepts of TREE/BST**
- ***TREE creation and implementation using BST***
- ***BS-TREE Traversal methods, InOrder, PostOrder and PreOrder***
- ***Lab exercise***

# TREES

## OVERVIEW

A tree is a data structure consisting of nodes organized as a hierarchy. A tree T is a set of nodes storing elements such that the nodes have a parent-child relationship that satisfies the following :

- if T is not empty, T has a special tree called the root that has no parent
- each node v of T different than the root has a unique parent node w; each node with parent w is a child of w

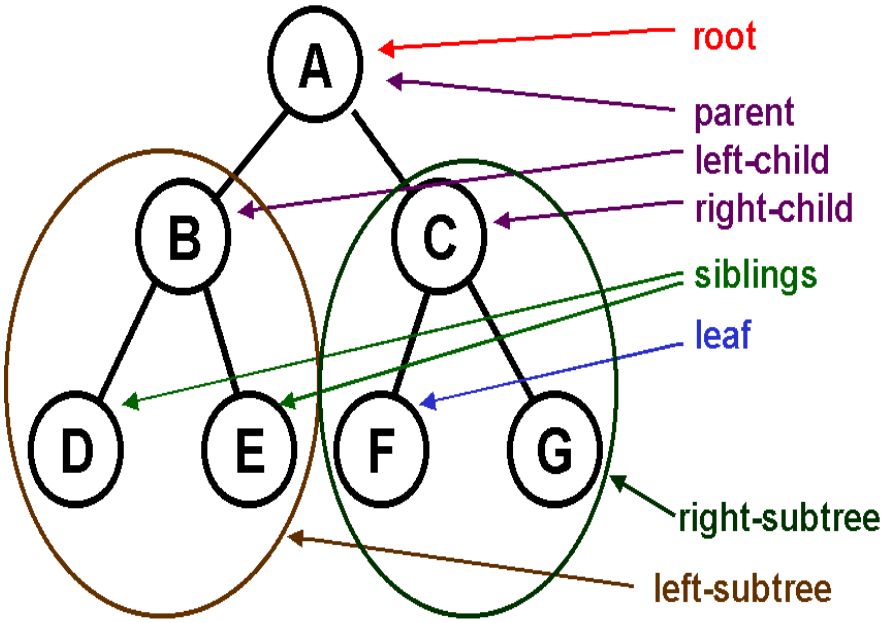


**AIM:**

- To perform Binary tree operations:

## Binary Search Tree:

A tree whose elements have at most 2 children is called a binary tree. Since each element in a binary tree can have only 2 children, we typically name them the left and right child. It is called a search tree because it can be used to search for the presence of a number in `O(log(n))` time.




Code fragment to create BST class

```
class BSTtree:
      def __init__(self): #constructor of class
          self.root = None
      def create(self,val):
          if self.root == None:
             self.root = Node(val)
          else:
             current = self.root
             while 1:
                 if val < current.info:
```

```
                    if current.left:
                       current = current.left
                    else:
                       current.left = Node(val)
                       break;
                 elif val > current.info:

                     if current.right:
                        current = current.right
                     else:
                        current.right = Node(val)
                        break;
                 else:
                    break
```

## Inorder Traversal

In an InOrder traversal, we recursively do an InOrder traversal on the left subtree, visit the root node, and finally do a recursive InOrder traversal of the right subtree

1. Traverse left subtree
2. root node,
3. traverse right subtree

## Sample Algorithm of Inorder

```
1. Initialize current as root
 2. While current is not NULL
Traverse the leftsubtree,  current = current->left
  If current does not have left child
   a. Print current's data
   b. Go to the right, (Traverse Right Subtree)i.e., current =
current->right
  Else
   a. In current's left subtree, make current the right child of
the rightmost node
   b. Go to this left child, i.e., current = current->left
```

Code segment for InOrder

```
def inorder(self,node):

            if node is not None:
```

```
            self.inorder(node.left)
            print (node.info)
            self.inorder(node.right)
```

**PostOrder Traversal**

In a preorder traversal, children node are visited before the node

- Traverse left subtree,
- traverse right subtree,
- visit node

Code segment for Postorder

```
def postorder(self,node):

        if node is not None:
            self.postorder(node.left)
            self.postorder(node.right)
            print (node.info)
```

**Pre-order Traversal**

In a preorder traversal, a node is visited before its descendants

- Visit node(Root),
- traverse left subtree,
- traverse right subtree

Code segment for PreOrder

```
    def preorder(self,node):

        if node is not None:
            print (node.info)
            self.preorder(node.left)
            self.preorder(node.right)
```

## Lab Exercise (Indivudual)

Modify the code given in the lab session, Complete the Post order of the tree. Create a Menu that enable user to add node to the tree and enable option for following operations

- Add the insertRightChild / insertLeftChild
- Print PostOrder Traversal ()
- Print Preorder Traversal ()
- Print Preorder Traversal ()
- exit

## Group Assignment

Modify/Extend the given code in class, (BST) to enable user insert some nodes, getSize of the tree(), num children(), CoutrightChild(), CountleftChild(), deleteLeaf() node and find a given target node from the list of nodes.

**SUBMIT BEFORE or By 11.59pm 22nd October, 2017**