Data Structure Lab

Lab # 08

Binary Search Tree

Instructor: Muhammad Saood Sarwar

Email: saood.sarwar@nu.edu.pk

Course Code: CL2001

Department of Computer Science,

National University of Computer and Emerging Sciences FAST Peshawar Campus

Binary Search Tree

Binary Search Tree is a form of tree which has a root, mid nodes and leaf nodes.

Binary Search Tree is more important and frequently used in various applications of computer science. When binary trees are in sorted form, they facilitate quick search, insertion and deletion.

A binary tree is either empty, or it consists of a node called the root together with two binary trees called the left subtree or the right subtree of the root. This definition is that of a mathematical structure. To specify binary trees as an abstract data type, we must state what operations can be performed on binary trees.

A binary search tree is a binary tree that is either empty or in which every node contains a key and satisfies the conditions:

1. The key in the left child of a node (if it exists) is less than the key in its parent node.
2. The key in the right child of a node (if it exists) is greater than the key in its parent node.
3. The left and right subtrees of the root are again binary search trees.

The first two properties describe the ordering relative to the key in the root node, and that the third property extends them to all nodes in the tree; hence we can continue to use the recursive structure of the binary tree.

Root node:

Node at the very top of tree

Leaf nodes:

Nodes at the end of tree

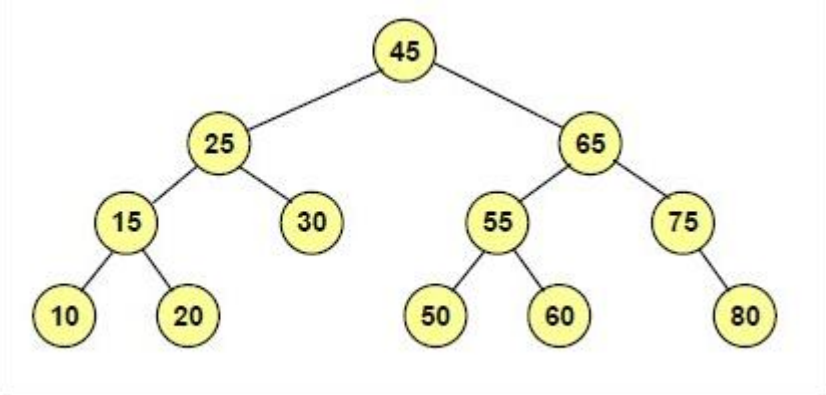
Mid nodes:

Nodes that are followed by root node but are above leaf nodes.

Every node has data, left pointer and right pointer in it except for leaf nodes. Leaf nodes only have data in them.

The specialty of BST is that every node (except for leaf nodes) has a node on it’s right side. containing value less than the node itself and on its left side containing value greater than the node itself.

Every node except for Root node and leaf nodes is a subtree.



Binary Search Tree Basic Implementation.

```
/*
Basic implementation of binary search tree
*/
#include <iostream>
using namespace std;
class Node{
   public:
   int data;
   Node *left,*right;
   Node(int v=0){
        data = v;
        left=right = NULL;
   }
};
class BST{
   public:
   Node *root;
   BST(){
        root=NULL;
```

```
    }
    Node* insert(Node *r,int val);
    void inOrderTraversal(Node *r);
   };
int main(int argc, char const *argv[])
{
    BST t1;
    t1.insert((t1.root),8);
    t1.insert((t1.root),2);
    t1.insert((t1.root),12);
    cout<<"_____"<<endl;
    t1.inOrderTraversal(t1.root);
    cout<<"_____"<<endl;
    return 0;
}


Node* BST::insert(Node *r,int val){
    if(r==NULL){
        Node *temp = new Node(val);
        if(r==root){
        /*
        this would run when first node in tree is entered
        so assigning the root to the first node
        */
           root = r = temp;
        }
        return temp;
```

```
    }
    else{
        if(r->data== val){
            cout<<"Already Exist"<<endl;
            return r;
        }
        else if(val<r->data)
        r->left = insert(r->left,val);
        else{
        r->right = insert(r->right,val);
        }
    return r;
    }
}
void BST::inOrderTraversal(Node *temp){
    if(temp==NULL) return;
    inOrderTraversal(temp->left);
    cout<<temp->data<<endl;
    inOrderTraversal(temp->right);
}
```