PES UNIVERSITY

Department of Computer Science and Engineering

PES UNIVERSITY

UE19CS202: Data Structures and its Applications (4-0-0-4-4)

Trees
Binary Search Tree (BST) and its Implementation using Dynamic Allocation: Insertion

Dr. Shylaja S S
Ms. Kusuma K V

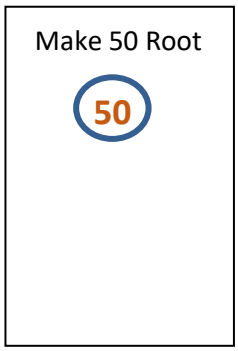
Binary Search Tree: is a Binary tree that is either empty or in which every node contains a key and satisfies the conditions:

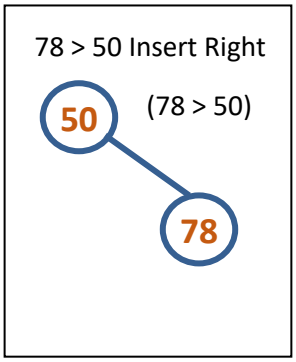
- The key in the left side of a child (if it exists) is less than the key in the parent node
- The key in the right side of a child (if it exists) is greater than the key in the parent node
- The left and right subtrees of the root are again binary search trees

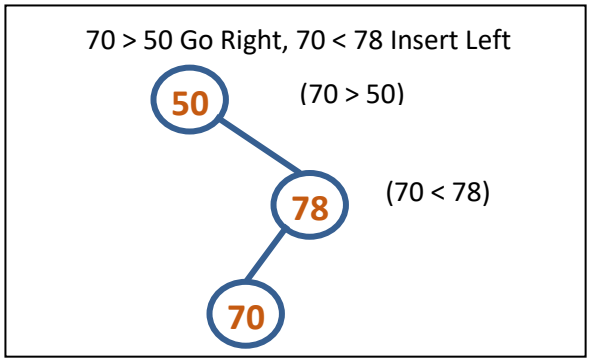
If in case of equal keys we can modify the definition but normally we take distinct keys.

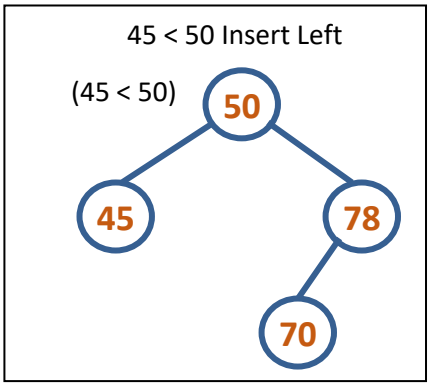
Construct a Binary Search Tree for the elements inserted in the following order:

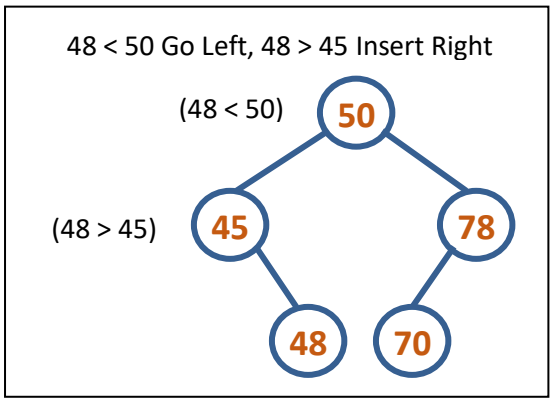
50, 78, 70, 45, 48, 35, 85

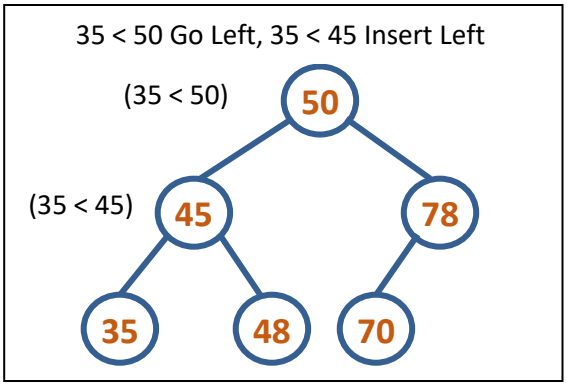


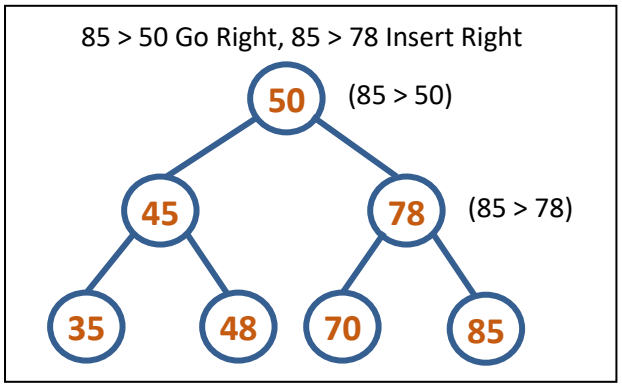












BST: Linked Representation

In linked representation each node in a binary tree has three fields, the left child field, information field and the right child field. Pictorially a node used for linked representation may be as shown in Figure3. If the left subtree is empty then the corresponding node's left child field is set to null. If the right subtree is empty then the corresponding node's right child field is set to null. If the tree itself is empty the root pointer is set to null.

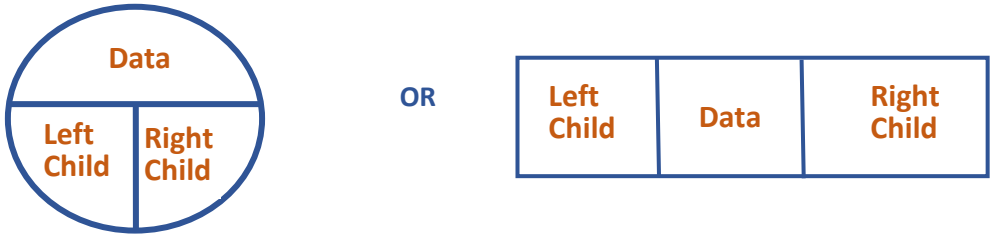


Figure 3: Pictorial representation of a node in Linked representation

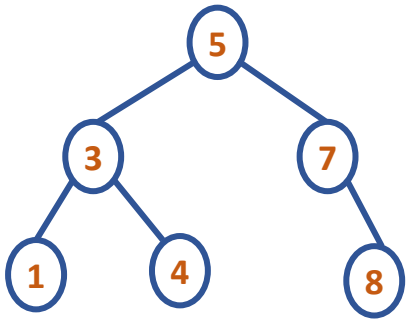


Figure 4: Binary Search Tree

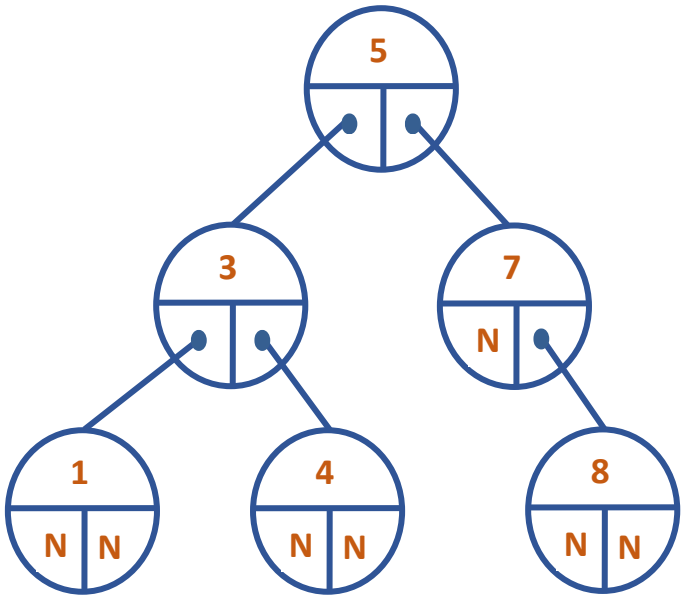


Figure 5: Linked Representation of BST in Figure 4

```
//BST Linked Representation
#include<stdio.h>
#include<stdlib.h>

typedef struct node
{
        int info;
        struct node *left,*right;
}NODE;
```

```
typedef struct tree
{
	NODE *root;
}TREE;

void init(TREE *pt)
{
	pt->root=NULL;
}

void creat(TREE *pt)
{
	NODE *temp,*p,*q;
	int wish;

	printf("Enter the root info\n");
	pt->root=(NODE*)malloc(sizeof(NODE));
	scanf("%d",&pt->root->info);

	pt->root->left=NULL;
	pt->root->right=NULL;
	do{
		printf("Enter an element\n");
		temp=(NODE*)malloc(sizeof(NODE));
		scanf("%d",&temp->info);
		temp->left=NULL;
		temp->right=NULL;
		q=NULL;
		p=pt->root;

		while(p!=NULL)
		{
			q=p;
			if(temp->info < p->info)
				p=p->left;
			else
				p=p->right;
		}
		if(temp->info < q->info)
			q->left=temp;
		else
			q->right=temp;
```

```
                printf("Do you wish to add another? 1/0\n");
                scanf("%d",&wish);
        }while(wish);
}

void intr(NODE* p)
{
        if(p!=NULL)
        {
                intr(p->left);
                printf("%d ",p->info);
                intr(p->right);
        }
}

void intrav(TREE *pt)
{
        intr(pt->root);
}

int main()
{
        TREE tobj;

        creat(&tobj);
        intrav(&tobj);

        return 0;
}
```