DSA through C++

Binary Search Tree

Saurabh Shukla (MySirG)

Agenda

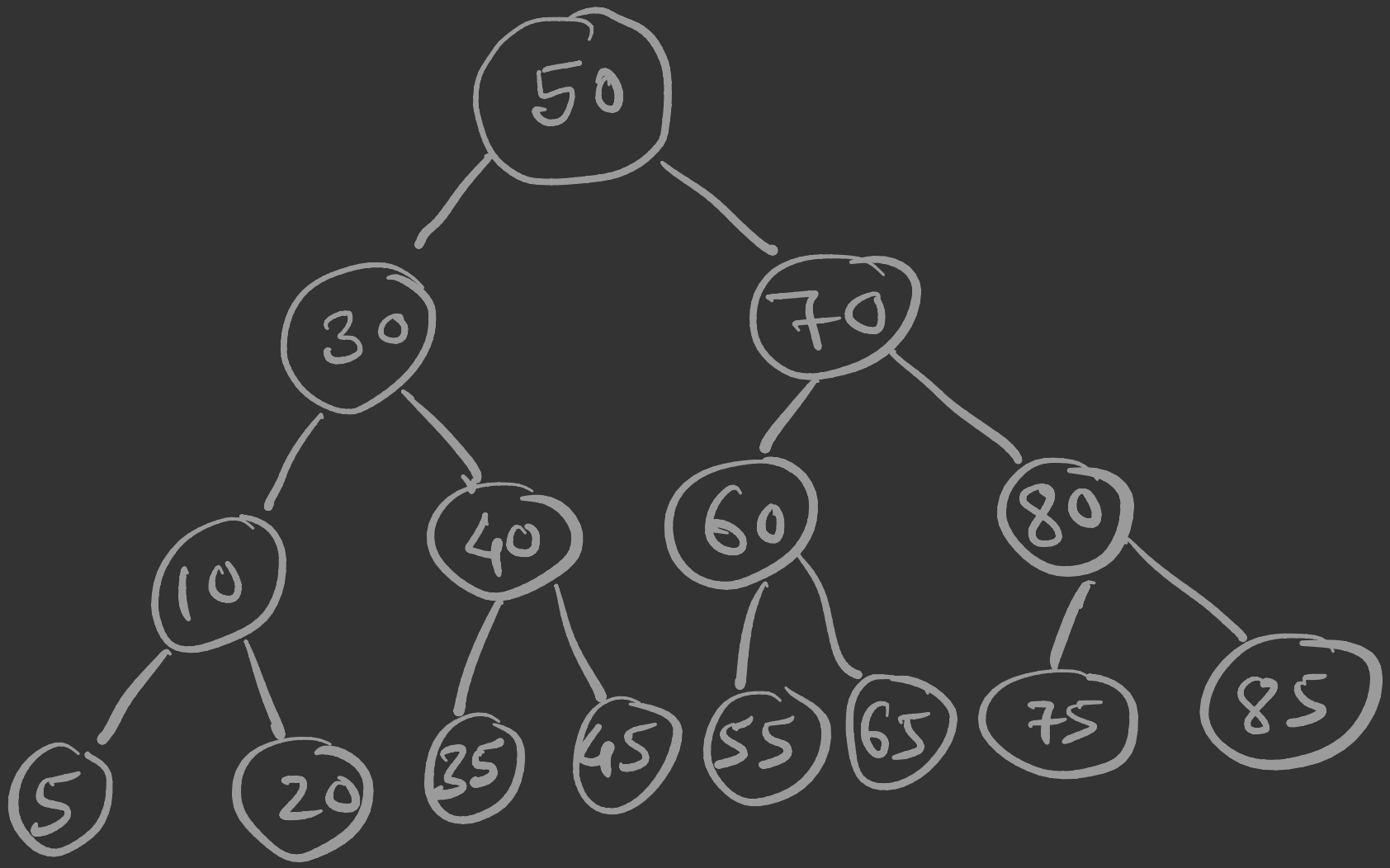
1. Binary Search Tree
2. Implementation of BST

Binary Search Tree

A binary search tree is the most important data structure, that enables one to search for and find an element with an average running time

$$f(n) = O(\log_2 n)$$

Duplicate values are not allowed in BST (By default)


50
30
70
10
40
60
80
5
20
35
45
55
65
75
85

Binary Search Tree is a binary tree with the value at node N is greater than every value in the left subtree of N and is less than every value in the right subtree of N.

Unless, explicitly said, BST doesn't allow duplicate values.

Implementation

1. node
2. Insertion
3. Traversing
4. Search
5. Deletion

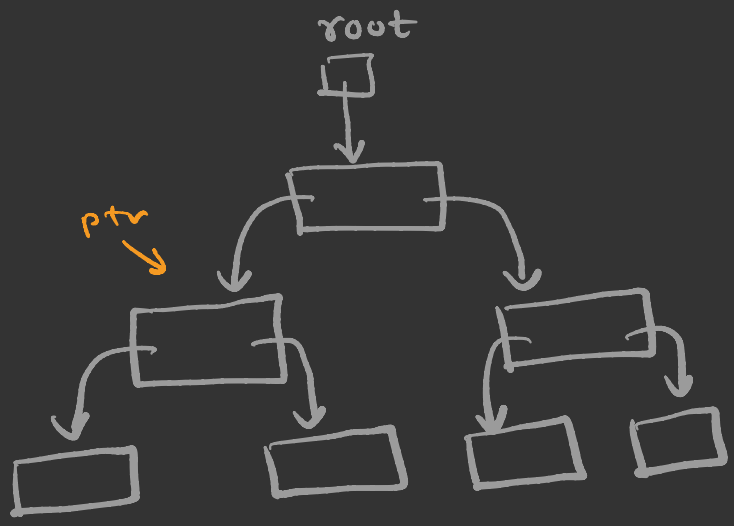
node

```
Struct node
{
    node *left;
    int item;
    node *right;
};
```

Insertion

data

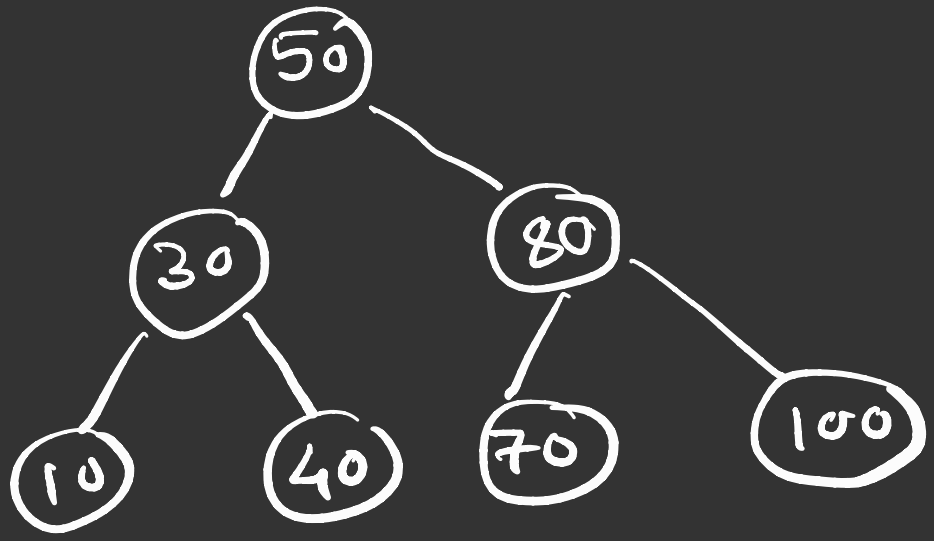
```
ptr = root;

while (ptr)
{
    if (ptr -> item == data)
    {
        break;
    }
    else if (data < ptr -> item)
    {
        if (ptr -> left == NULL)
        {
            node *n = new node;
            n -> left = NULL;
            n -> item = data;
            n -> right = NULL;
            ptr -> left = n;
            break;
        }
        else
            ptr = ptr -> left;
    }
    else {
        if (ptr -> right == NULL)
        {
            node *n = new node;
            n -> left = NULL;
            n -> item = data;
            n -> right = NULL;
            ptr -> right = n;
            break
        }
        else
            ptr = ptr -> right;
    }
}
```



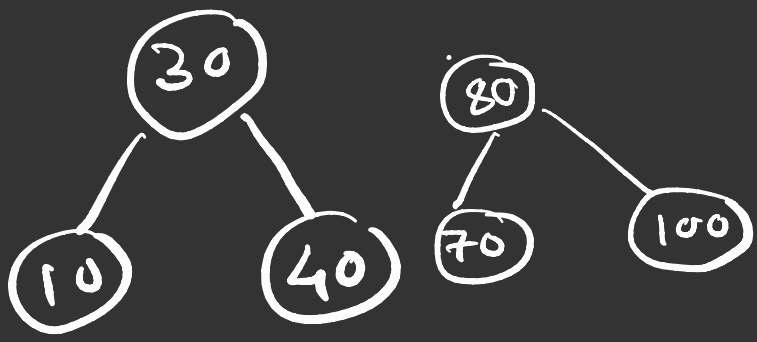
traversing

pre order traversing

Root | Left Sub Tree | Right Sub tree



50

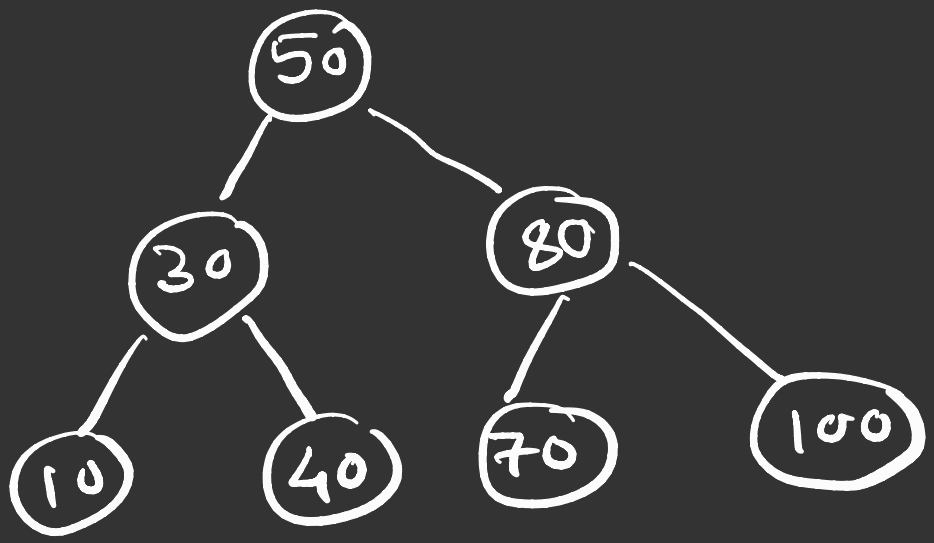


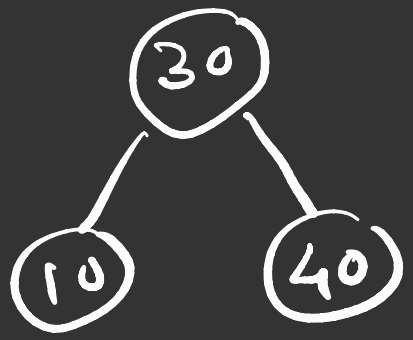
30 10 40 80 70 100

50 30 10 40 80 70 100

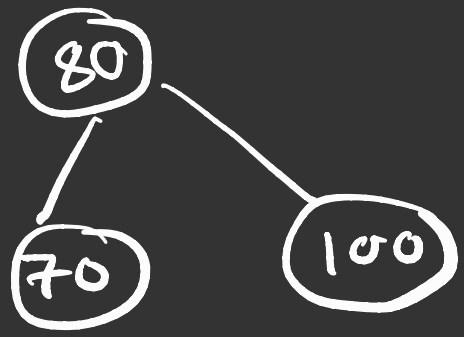
Inorder traversing

Left Subtree | Root | Right Subtree





50



10 30 40

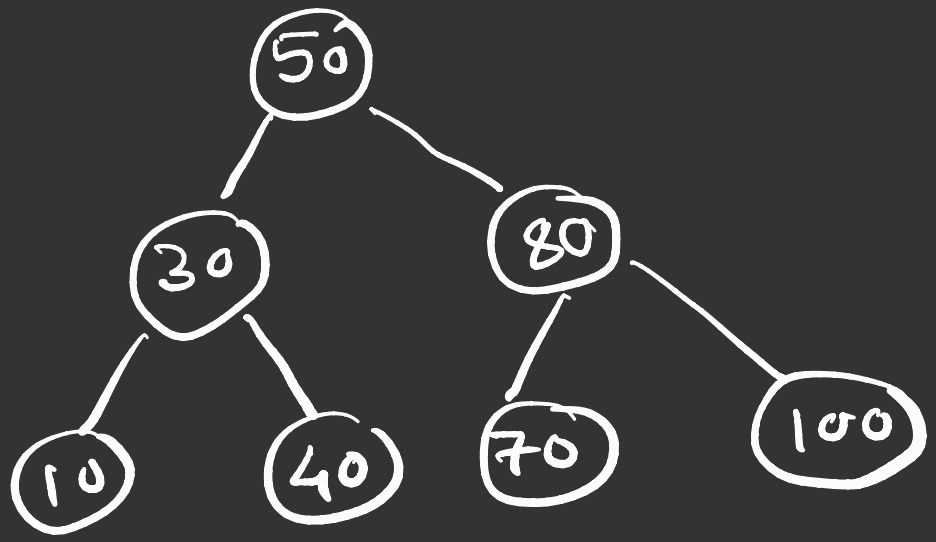
70 80 100

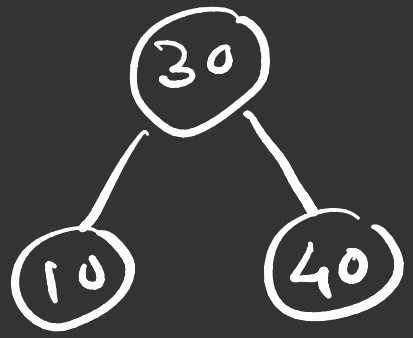
10 30 40 50 70 80 100

Inorder traversing of BST always gives sorted order of elements.

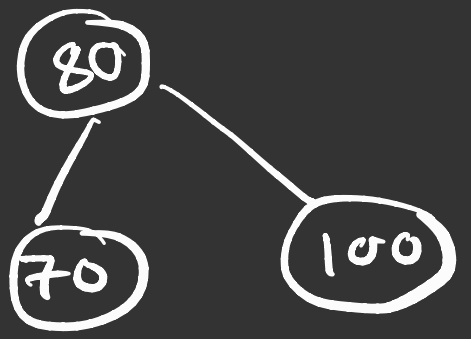
Postorder traversing

Left Subtree Right Subtree Root





10 40 30



70 100 80

50

50

10 40 30 70 100 80 50

Deletion

10 No child

40 Single child

80 Two children