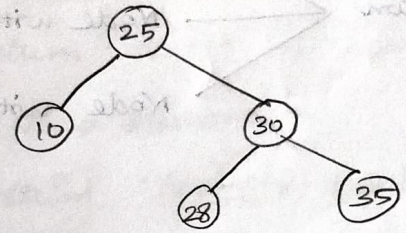
Binary Search Tree ADT

Binary Search Tree (BST) is a binary tree in which for every node x, the values of all the keys in the left subtree are smaller than the key value x, values of all the keys in the right subtree are greater than the key value x.



Ex: Binary Search Tree.

Construction of Binary Search Tree:

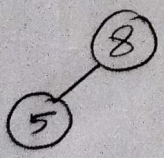
To Insert 8, 5, 10, 15, 20, 18, 3

① Insert 8

8 is considered as root

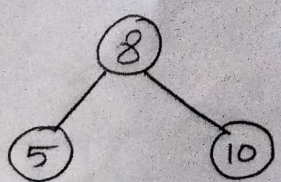
② Insert 5

5 < 8, left child



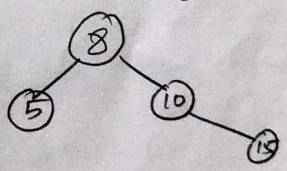
③ Insert 10

10 > 8 → right child

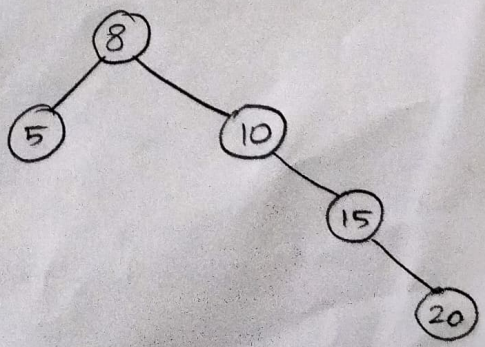


④ Insert 15

15 > 8 → right subtree



⑤ Insert 20

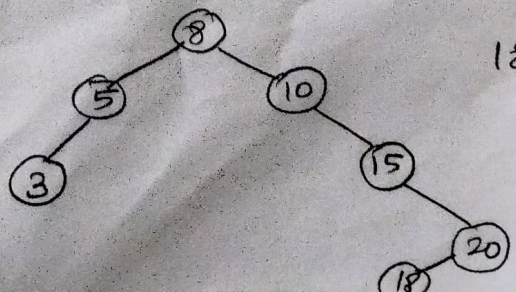


⑥ Insert 18, 3

18 > 8 ⇒ 18 > 10 ⇒ 18 > 15

⇓

18 < 20



Binary Search Tree.

Operations:

1. Insertion
2. Searching
3. Findmin
4. Findmax
5. Deletion
   - leaf node
   - Node with one children
   - Node with two childrens

Inseation

Example 2: To insert 8, 4, 1, 6, 5, 7, 10.

Routine for insertion:

Searching.

To search a key value X, Compare X with root node, if key value is less than root traverse towards left subtree else traverse towards right subtree.

routine:

```
find ( Element X, Tree T)
{
    if (T == Null)
        return Null;
    if ( X < T -> Element)
        return find (X, T -> left);
    else if (X > T -> Element)
        return find (X, T -> right);
    else
        return T;
}
```

Findmin

To find the minimum element in the tree.

```
findmin (Tree T)
{
    if (T == Null)
        return Null;
    if (T -> left == Null)
        return T;
    else
        return findmin (T -> left);
}
```

Find max.

To find the maximum element in the Tree

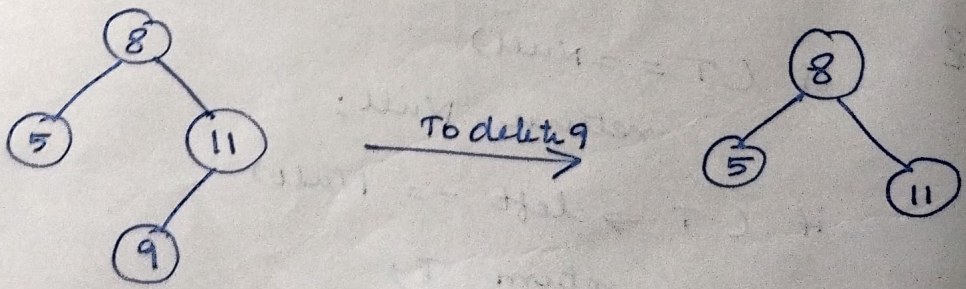
```
findmax (Tree T)
{
    if (T == Null)
        return Null;
    if (T -> right == Null)
        return T;
    else
        return findmax (T->right);
}
```

Deletion.

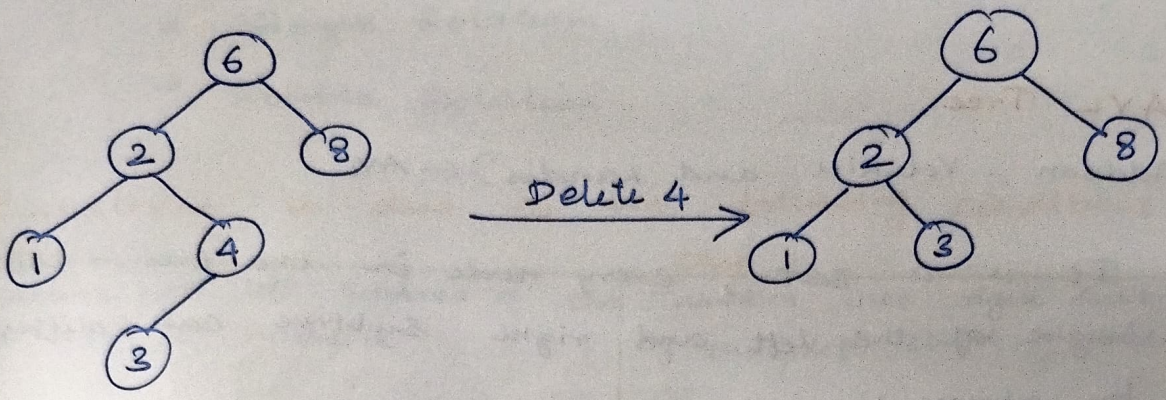
3 Case

1. Leaf node
2. Node with one child
3. Node with two childrens.

1. Leaf Node to be deleted

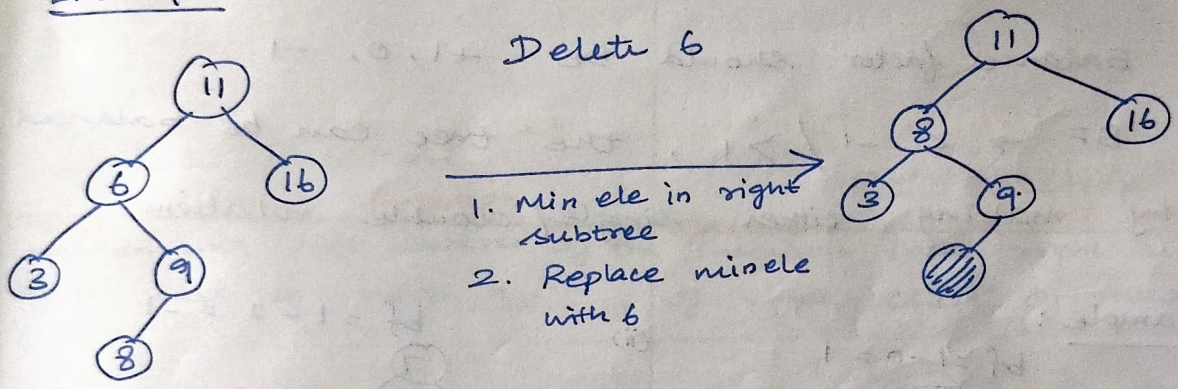


2. Node with one child

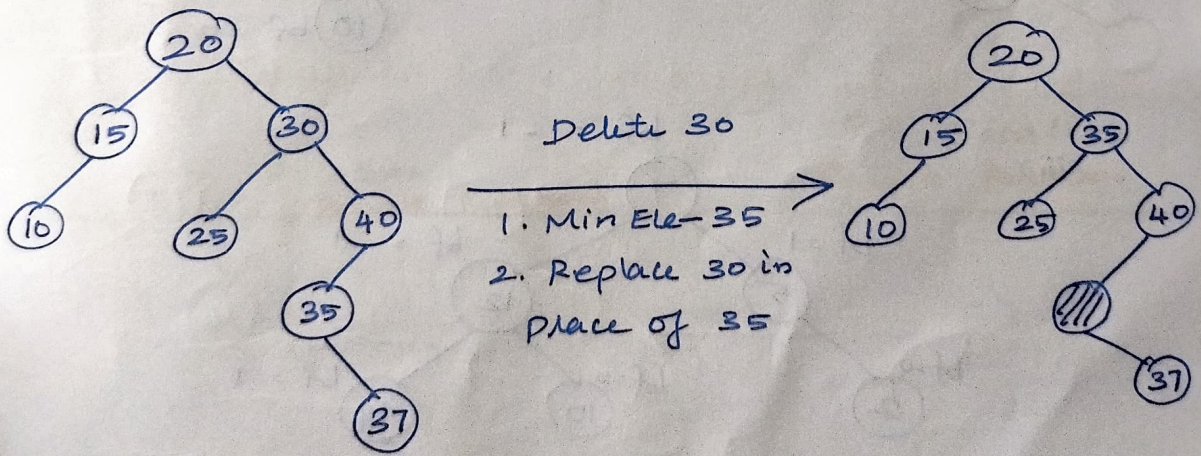


3. Node with 2 childrens

Example 1:



Example 2:



Final Solution