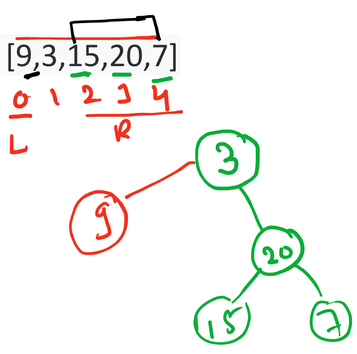
Lec-33

15/09/24 4:57 PM

Root Left Right

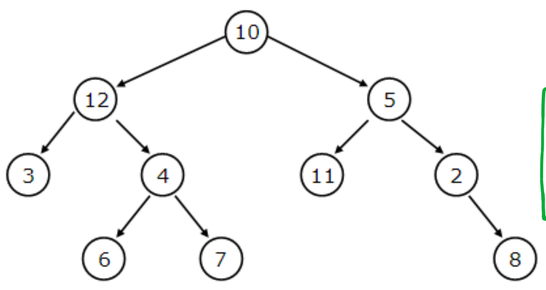
Left Root Right

preorder = [3,9,20,15,7], in-order = [9,3,15,20,7]



(i) pre-order 1st Node Root

(ii) in-order में pre-order के पहला Node की खोज उसको Search करेंगे

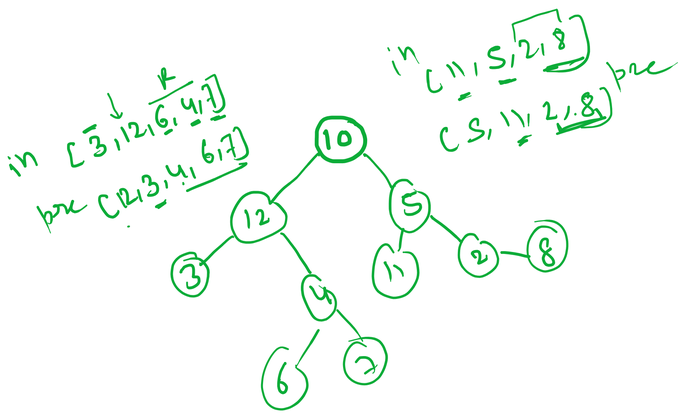


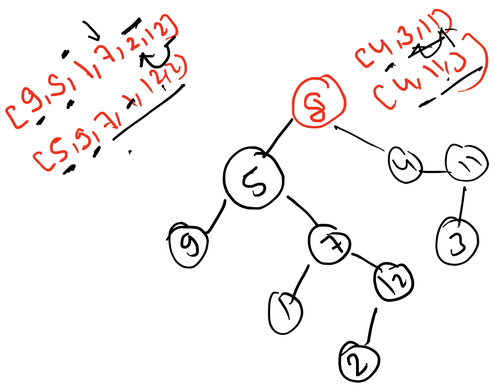
Levelorder tree traversal
10, 12, 5, 3, 4, 11, 2, 6, 7, 8

Inorder tree traversal
3, 12, 6, 4, 7, 10, 11, 5, 2, 8

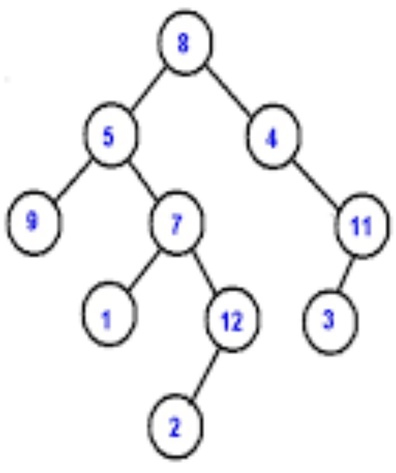
Preorder tree traversal
10, 12, 3, 4, 6, 7, 5, 11, 2, 8

Postorder tree traversal
3, 6, 7, 4, 12, 11, 8, 2, 5, 10



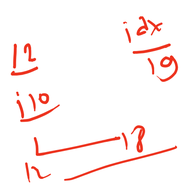


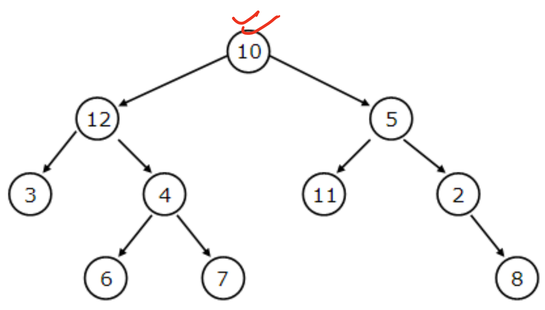
PreOrder - 8, 5, 9, 7, 1, 12, 2, 4, 11, 3
InOrder - 9, 5, 1, 7, 2, 12, 8, 4, 3, 11
PostOrder - 9, 1, 2, 12, 7, 5, 3, 11, 4, 8
LevelOrder - 8, 5, 4, 9, 7, 11, 1, 12, 3, 2



```
public TreeNode Create(int[] pre, int[] in, int ilo, int ihi, int plo, int phi) {
    TreeNode nn = new TreeNode(pre[plo]);
    int idx=Search(in, ilo, ihi, pre[plo]);
    nn.left=Create(pre, in, ilo, idx-1, ⊥);
    nn.right=Create(pre, in, idx+1, ihi, ⊥);
    return nn;

}
```





Levelorder tree traversal
10, 12, 5, 3, 4, 11, 2, 6, 7, 8

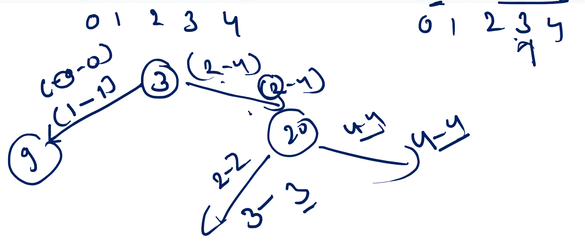
Inorder tree traversal
3, 12, 6, 4, 7, 10, 11, 5, 2, 8

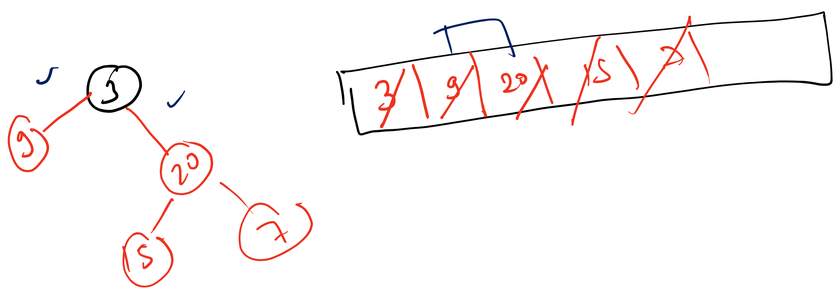
Preorder tree traversal
10, 12, 3, 4, 6, 7, 5, 11, 2, 8

Postorder tree traversal
3, 6, 7, 4, 12, 11, 8, 2, 5, 10

```
public TreeNode Create(int[] pre, int[] in, int ilo, int ihi, int plo, int phi) {
        if (ilo > ihi || plo > phi) {
            return null;
        }
        TreeNode nn = new TreeNode(pre[plo]);
        int idx = Search(in, ilo, ihi, pre[plo]);
        int c = idx - ilo;
        nn.left = Create(pre, in, ilo, idx - 1, plo + 1, plo + c);
        nn.right = Create(pre, in, idx + 1, ihi, plo + c + 1, phi);
        return nn;
    }
```

preorder = [3,9,20,15,7], in-order = [9,3,15,20,7]





```
private void Create_Tree() {
        int item = sc.nextInt();
        Node nn = new Node();
        nn.val = item;
        root = nn;
        Queue<Node> q = new LinkedList<>();
        q.add(nn);
        while (!q.isEmpty()) {
            Node n = q.poll();
            int c1 = sc.nextInt();
            int c2 = sc.nextInt();
            if (c1 != -1) {
                Node node = new Node();
                node.val = c1;
                n.left = node;
                q.add(node);
            }
            if (c2 != -1) {
                Node node = new Node();
                node.val = c2;
                n.right = node;
                q.add(node);
            }
        }
    }
```