Binary Tree : Info → Generic Tree

Structured based Data structure

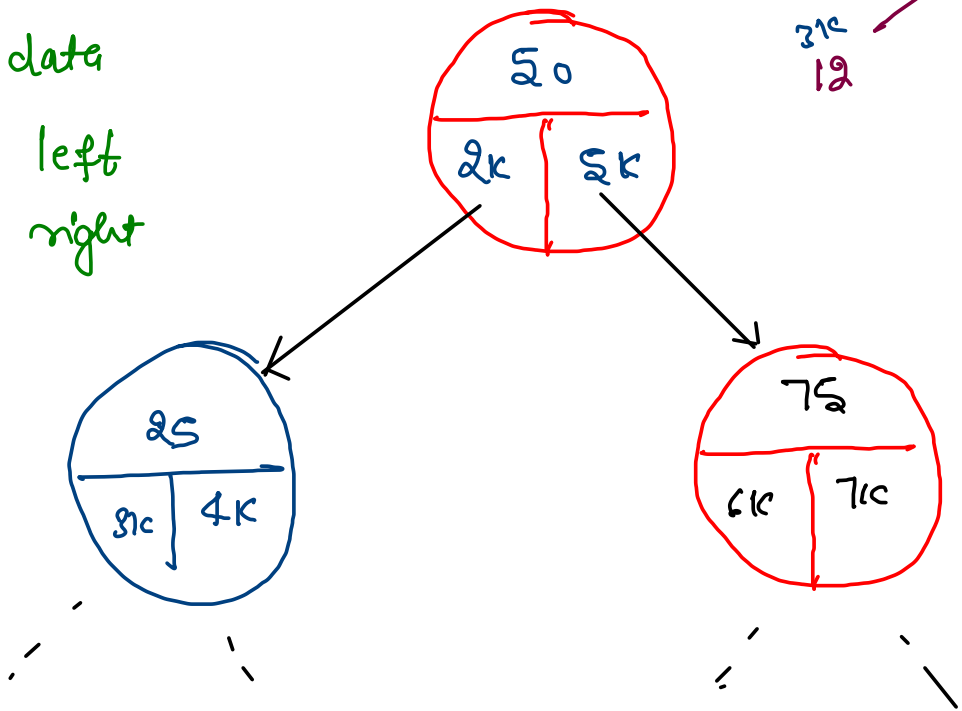
No. of child <= 2

Info. of Node →

int data

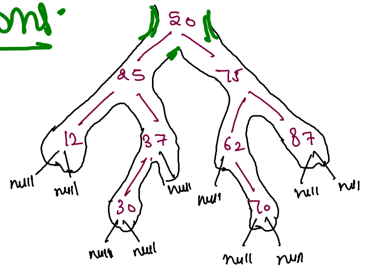
Node left

Node right



Constructions:

pre Order →

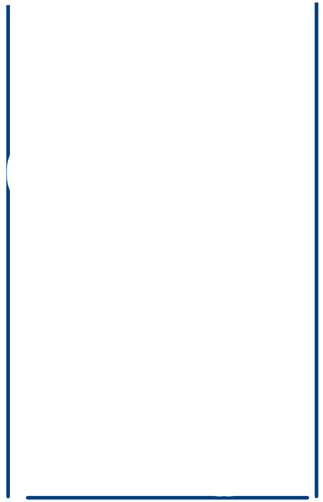


State = 0 (left child)

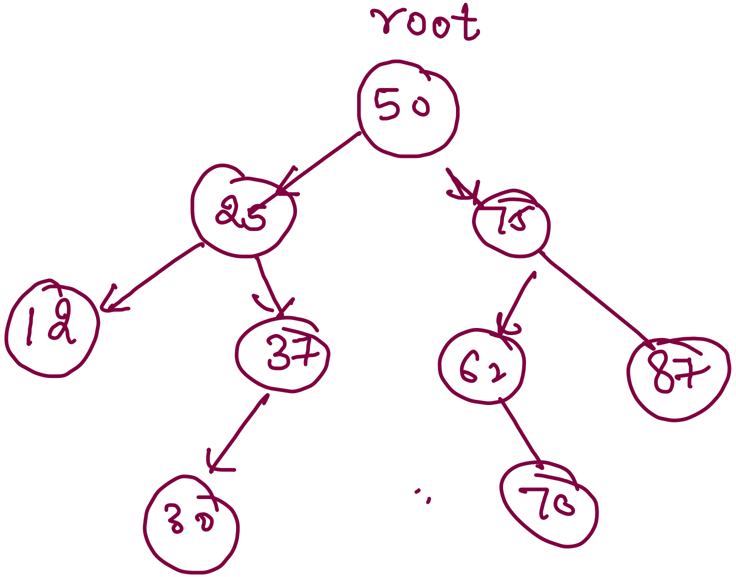
State = 1 (Right child)

State = 2 (pop)

- 50
- 25
- 12
- null
- null
- 87
- 30
- null
- null
- null
- 75
- 62
- null
- 70
- null
- null
- 87
- null
- null



node - State



→ Index loop (X)

```java
Node root = new Node(arr[0]);

Stack<Pair> st = new Stack<>();
st.push(new Pair(root, 0));

int indx = 0;
while(st.size() > 0) {
    Pair p = st.peek();
    if(p.state == 0) {
        // left child processing
        indx++;
        if(arr[indx] != null) {
            Node nn = new Node(arr[indx]);
            p.node.left = nn;
            st.push(new Pair(nn, 0));
        }
        p.state++;
    } else if(p.state == 1) {
        // right child processing
        indx++;
        if(arr[indx] != null) {
            Node nn = new Node(arr[indx]);
            p.node.right = nn;
            st.push(new Pair(nn, 0));
        }
        p.state++;
    } else {
        // pop out node-pair from stack
        st.pop();
    }
}
return root;
```

preorder

a

1 b

2 d

3 null

4 null

5 null

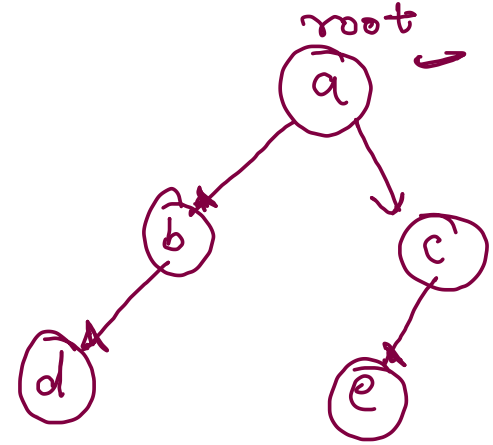
6 c

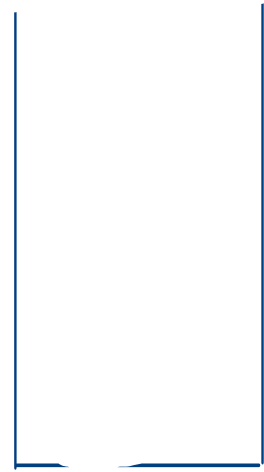
7 e

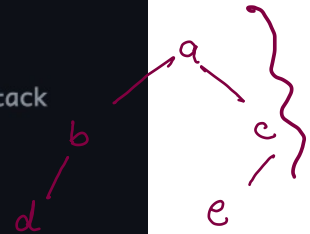
8 null

9 null

10 null







Indx=0 1 2 3 4 5 6 7 8 9 10

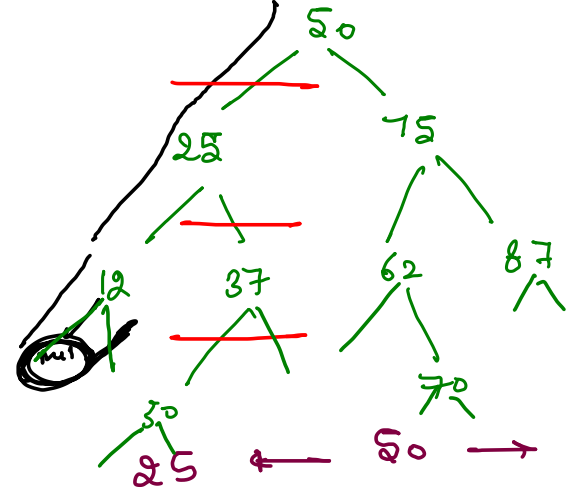
display :- →

High level thinking :) →

Expectation. → display(50)

faith → display(50.left);
display(50.right);

Merging → root.left.data ← root.data → root.right.data.
display(root.left);
display(root.right);



25 ← 50 → 75

12 ← 25 → 37

. ← 12 → .

30 ← 37 → .

. ← 30 → .

62 ← 75 → 87

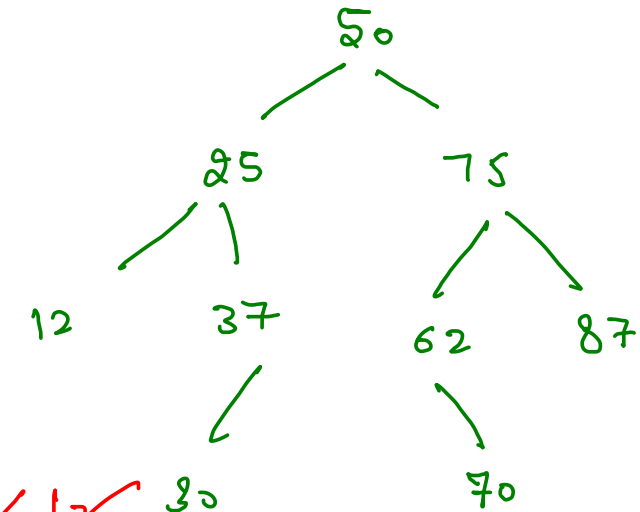
. ← 62 → 70

. ← 70 → .

. ← 87 → .

Level Order · steps: ① get ② remove → clubbed. ③ print ④ add children.

Size



~~50~~ ~~25~~ ~~75~~ ~~12~~ ~~37~~ ~~62~~ ~~87~~ ~~30~~ ~~70~~

→ 50

→ 25 75

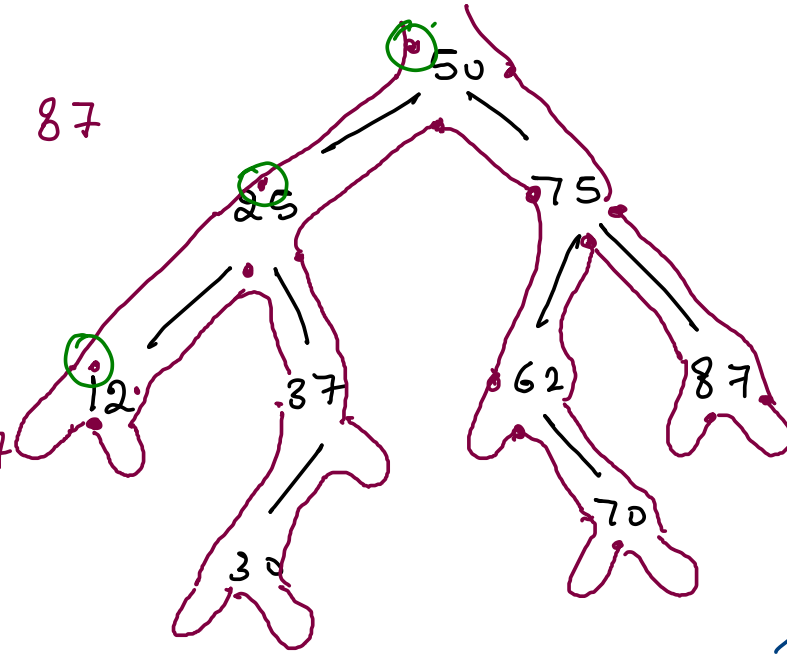
→ 12 37 62 87

→ 30 70

Traversals →

Pre order 50 25 12 37 30 75 62 70 87

Prea Area



Inorder: 12 25 30 37 50 62 70 75 87

In Area } Area between left and right call

PostOrder 12 30 37 25 70 62 87 75 50

Area after both calls.