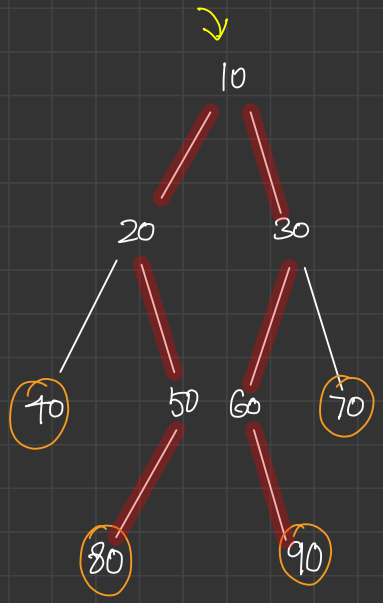
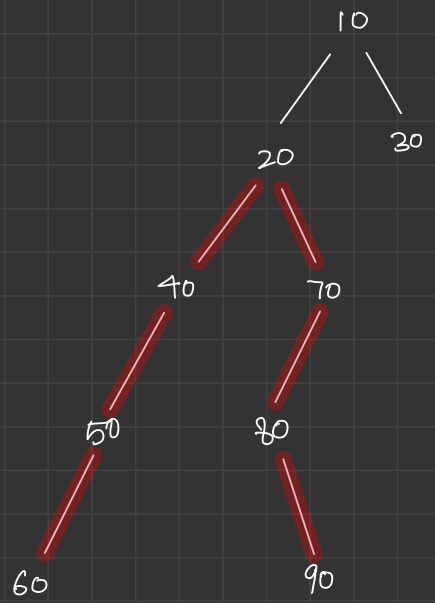
diameter of a tree { max^m dist. b/w any two leaf nodes }



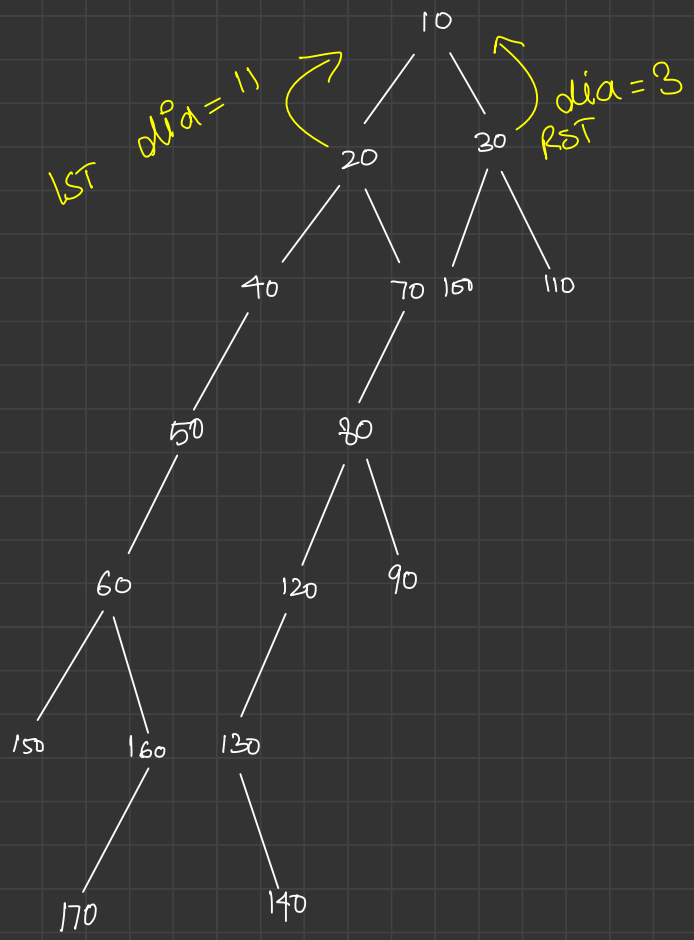
diameter = 7.



diameter = 7

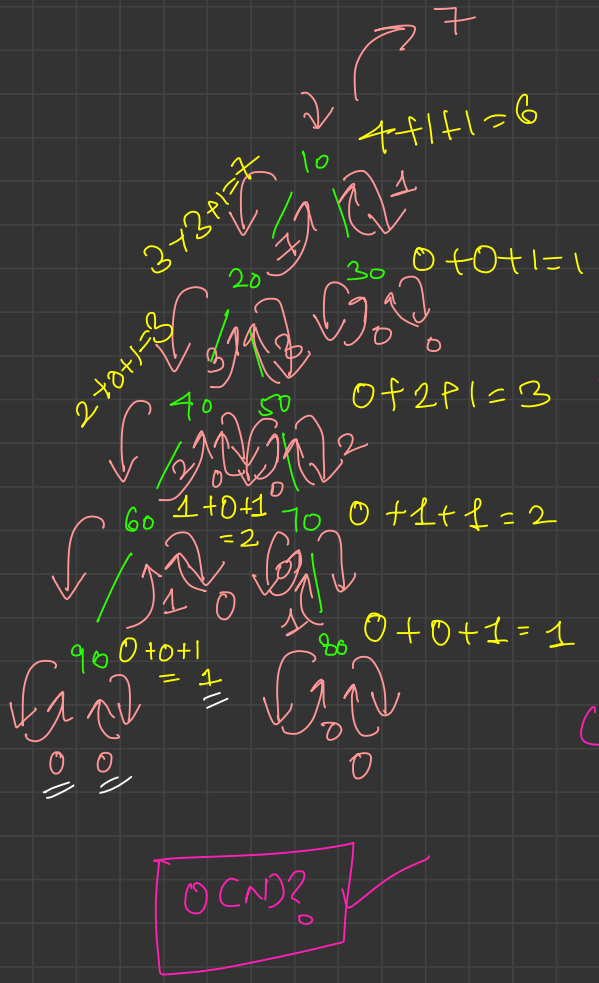
maxm(dia LST, dia RST, mdia) ⟶ diameter of tree

lh = 6, rh = 2, mdia = 6+2+1 = 9



int diameter (Node root)

{ faith : returns diameter of the tree

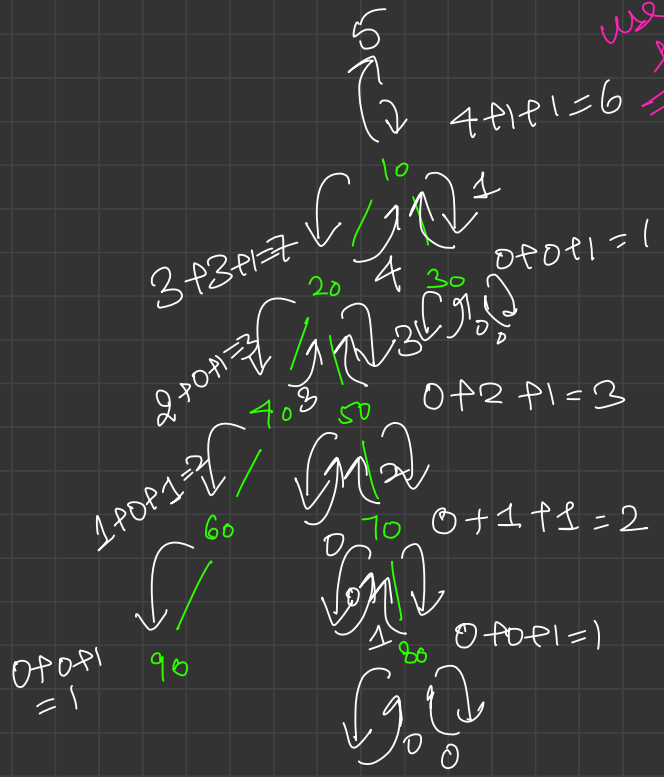


TC: $O(N^2)$

SC: $O(H)$

height of the tree (call stack space)

```
int diameter(Node root)
{
    if (root == null) return 0;

    int diaLST = diameter(root.left);

    int diaRST = diameter(root.right);

    diameter passing through me

    int hLST = height(root.left);
    int hRST = height(root.right);
    int midia = (hLST+1) + hRST;

    return Max(diaLST, diaRST, midia);
}
```



use is backtracked

↳ max of dia through each node

int dia = 0; 1 2 3 7

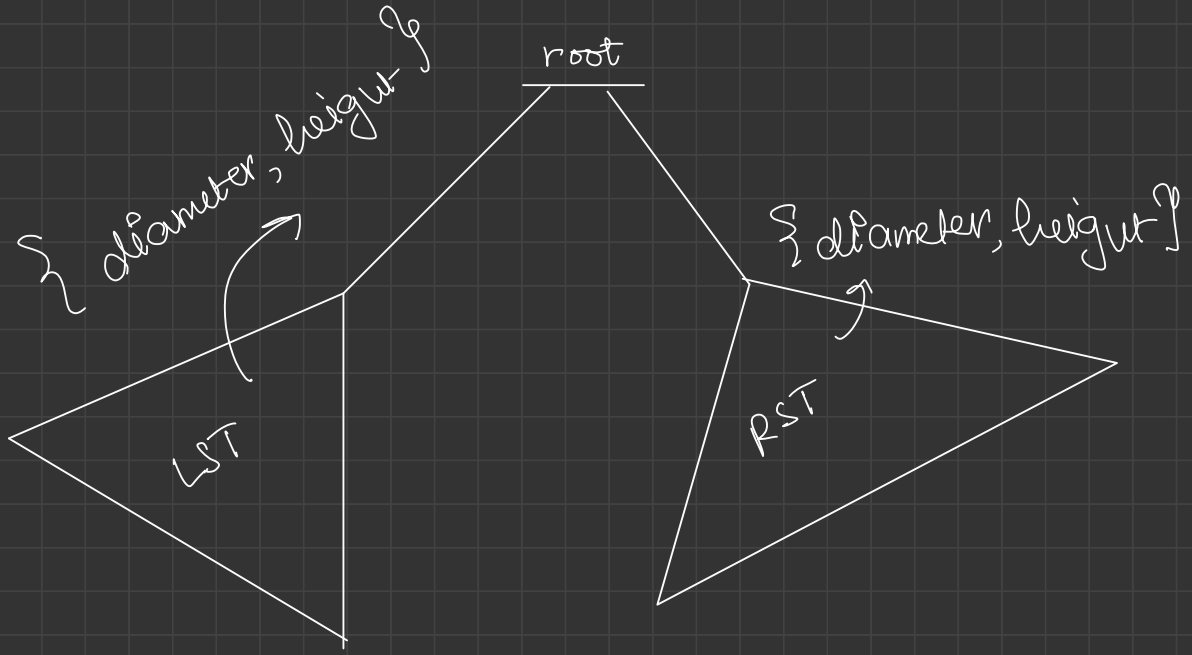
```java
public int heightOfTree(Node root) {
    if (root == null) {
        return 0;
    }
        You, 22 hours ago • Binary Trees 1
    int heightOfLST = heightOfTree(root.left);

    int heightOfRST = heightOfTree(root.right);

    int height = Math.max(heightOfLST, heightOfRST) + 1;

    return height;
}
```

int mdia = hLST + hRST + 1;

dia = max(dia, mdia);

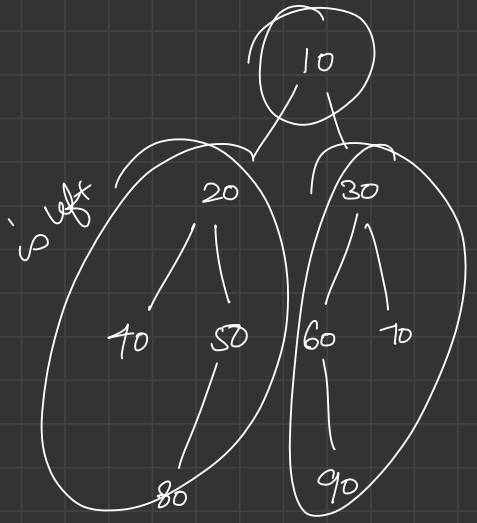
TC: O(N)

SC: O(H)


root
{ diameter, height }
{ diameter, height }
LST
RST

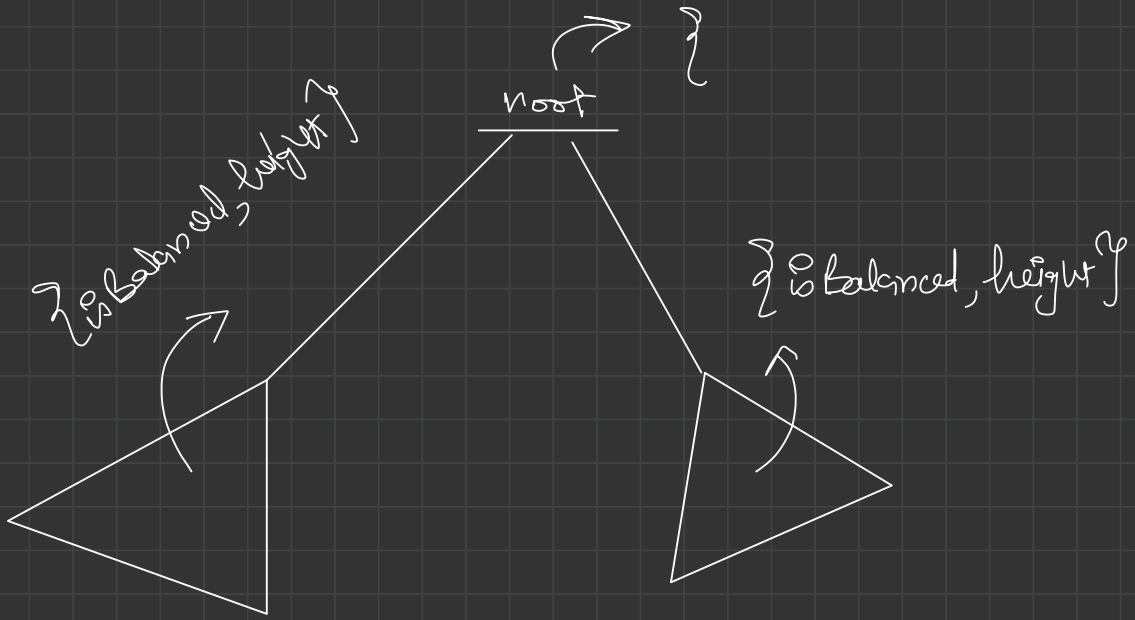
Is tree Balanced.

a node is Balanced. $|lh - rh| \leq 1$

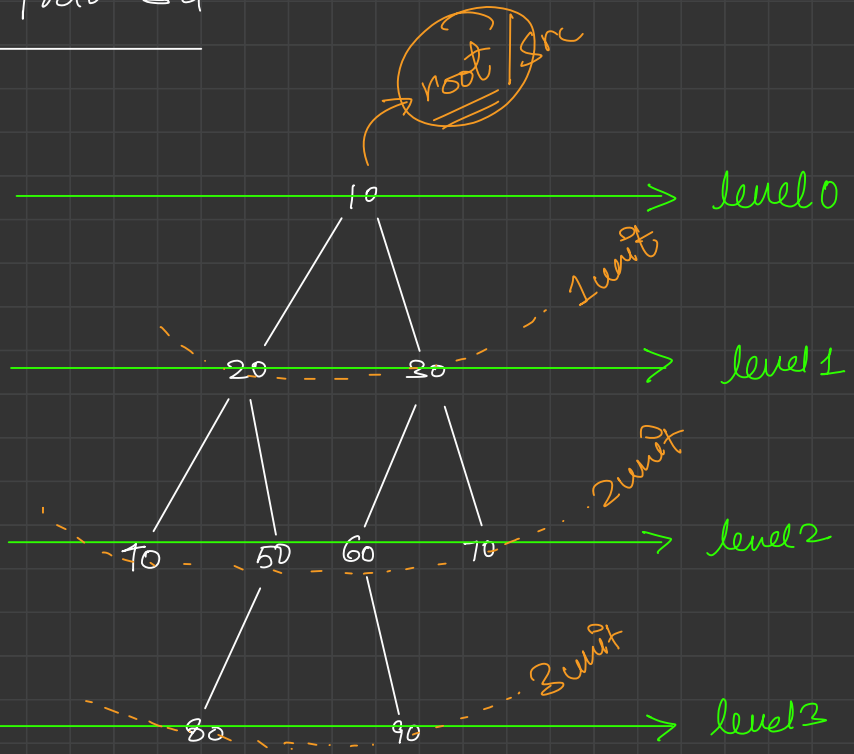


brute force:

TC: $O(N^2)$


root
{isBalanced, height}
{isBalanced, height}
{ }

level Order Traversal



o/p

0: 10
1: 20 30
2: 40 50 60 70
3: 80 90

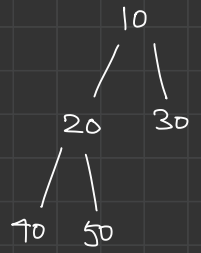
level No.

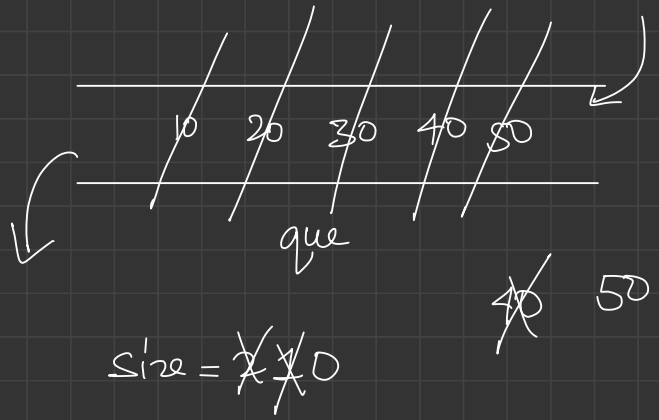
10
20 30
40 50 60 70
80 90

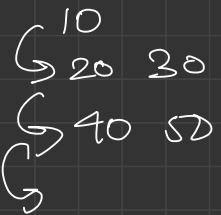
size = ~~7~~ 10

~~80~~ ~~90~~

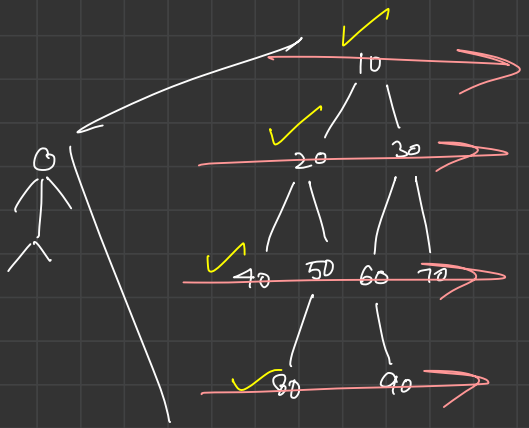
```java
Queue<TreeNode> que = new ArrayDeque<>();

que.add(root);

while (que.size() != 0) {
    int size = que.size();
    while (size--> 0) {
        TreeNode rnode = que.remove();

        System.out.print(rnode.val + " ");

        if (rnode.left != null) {
            que.add(rnode.left);
        }

        if (rnode.right != null) {
            que.add(rnode.right);
        }
    }
    System.out.println();
}
```







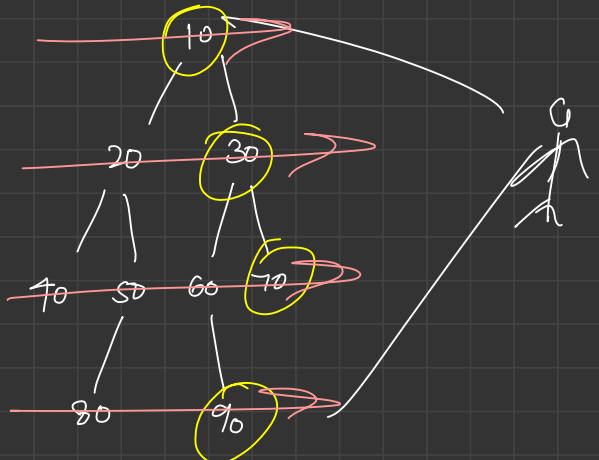
Views.



10, 20, 70, 80

↓

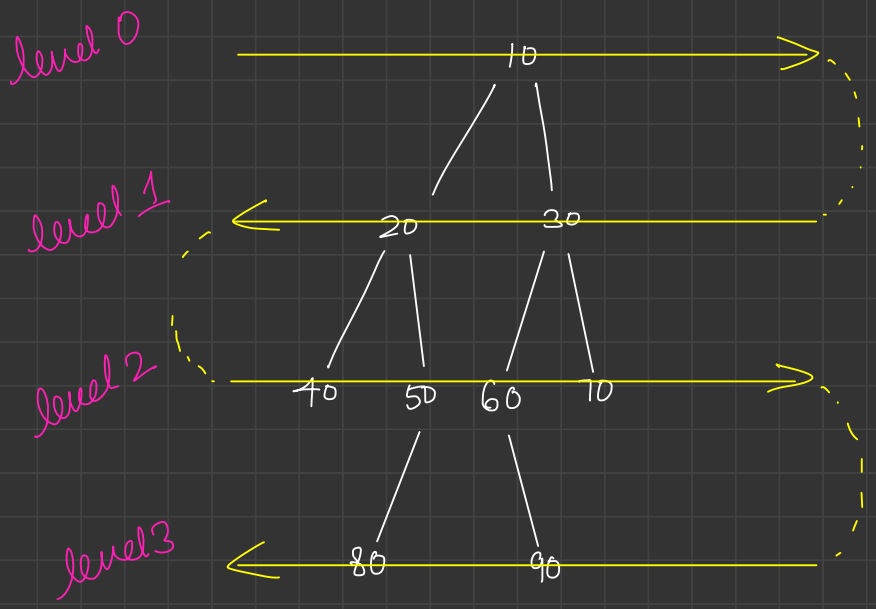
left View



{10, 30, 70, 90}

↳ right View

Zig-Zag Traversal

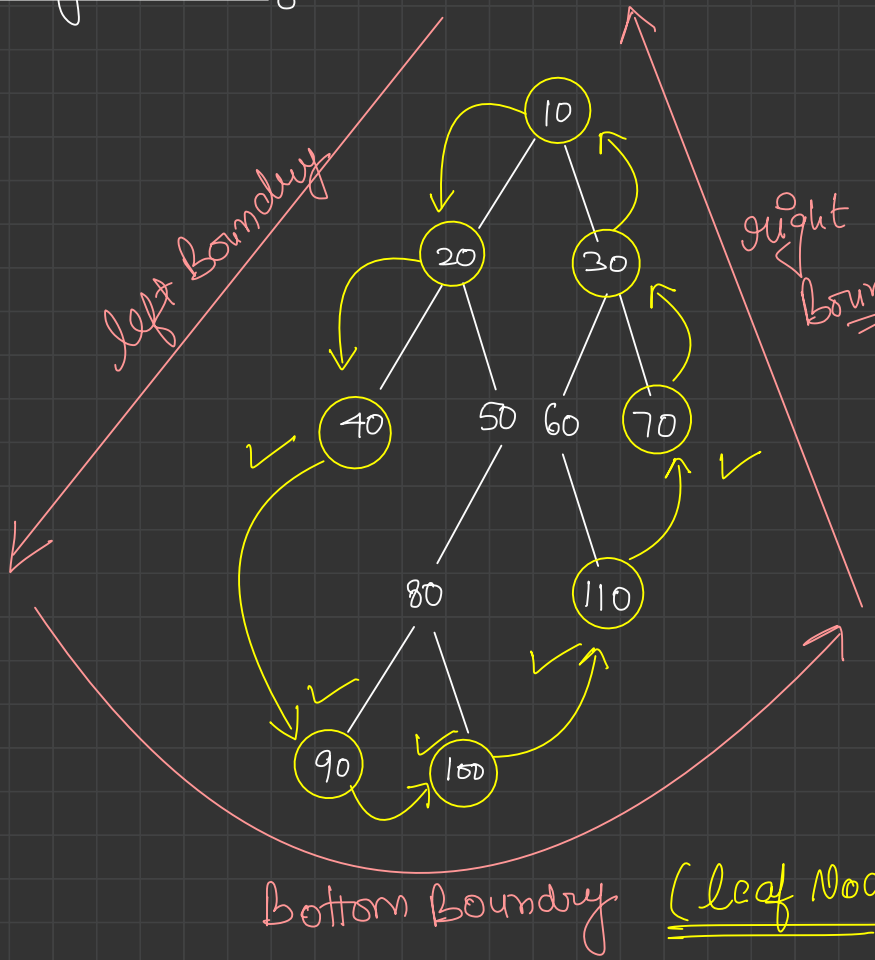


O/P

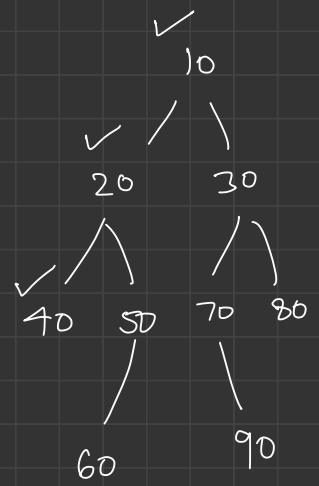
10
30 20
40 50 60 70
90 80

odd levels → right to left
even levels → left to right

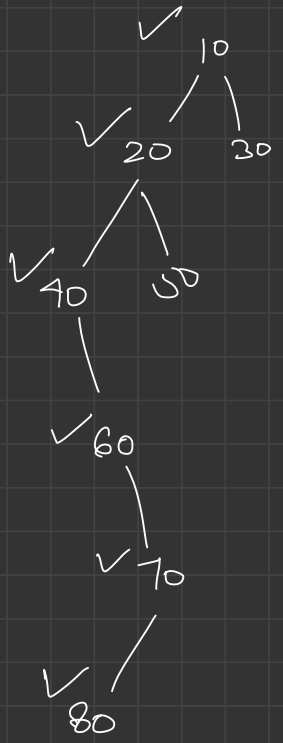
Boundry Traversal



{10, 20, 40, 90, 100, 110, 70, 30}



lb: 10, 20, 40



lb: 10, 20, 40, 60, 70, 80