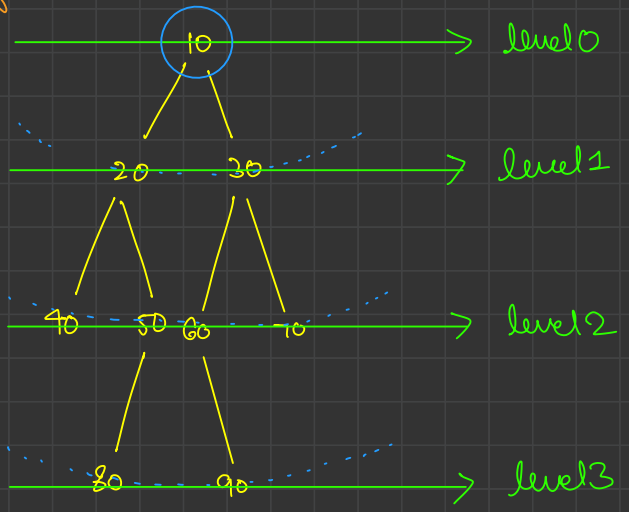
Traversal on Binary Trees

1. Pre-order
2. In-order
3. Post-order

4. level order traversal

level Order Traversal

BFS (Breath first search)



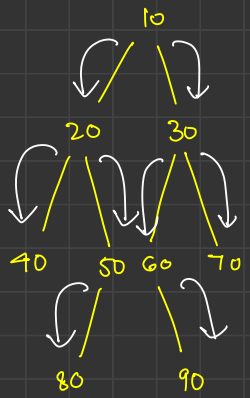
Q?

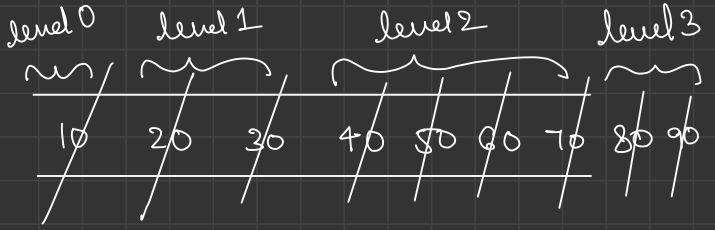
10

20 30

40 50 60 70

80 90





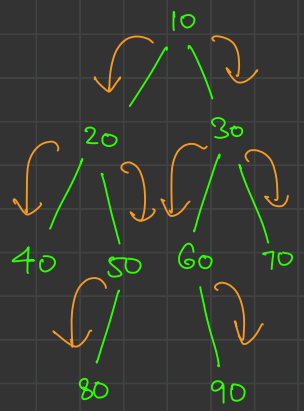
size = 10 20 30 40 50 60 70 80 90

O/P

10
20 30
40 50 60 70
80 90

```
Queue <TreeNode> que = new ArrayDeque<>();
que.add(root);
level = 0;
while(que.size() != 0)
{
    int size = que.size();
    while(size --> 0)
    {
        TreeNode rnode = que.remove();
        print(rnode.val);
        if(rnode.left != null)
            que.add(rnode.left);
        if(rnode.right != null)
            que.add(rnode.right);
    }
    level++;
}   println();
```

```java
public List<List<Integer>> levelOrder(TreeNode root) {
    List<List<Integer>> lo = new ArrayList<>();

    if (root == null) {
        return lo;
    }

    Queue<TreeNode> que = new ArrayDeque<>();
    que.add(root);

    while (que.size() != 0) {
        int size = que.size();

        List<Integer> currLevel = new ArrayList<>();

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

            currLevel.add(rnode.val);

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

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

        lo.add(currLevel);
    }

    return lo;
}
```



TC : O(N)
SC : O(2^h)

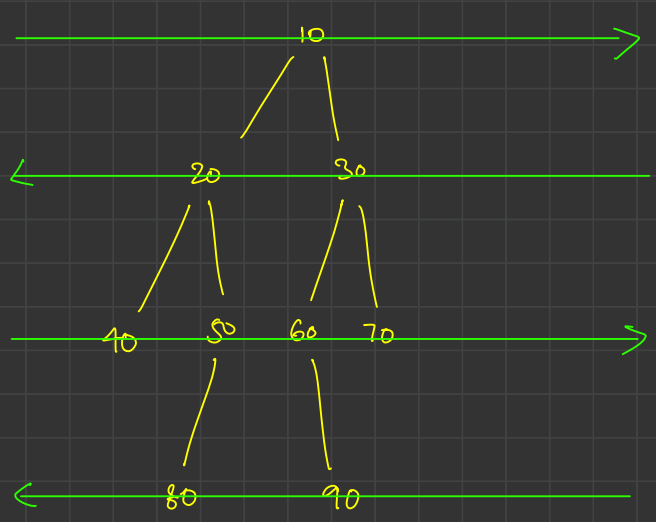
10 20 30 40 50 60 70 80 90 —

size = 1 2 4 4 2 1

{ {10} {20, 30} {40, 50, 60, 70}, {80, 90} }

Zig-Zag Traversal

Even level → Left to Right
Odd level → Right to Left



level = 0

level = 1

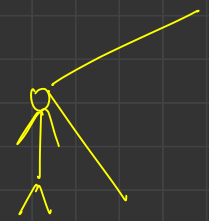
level = 2

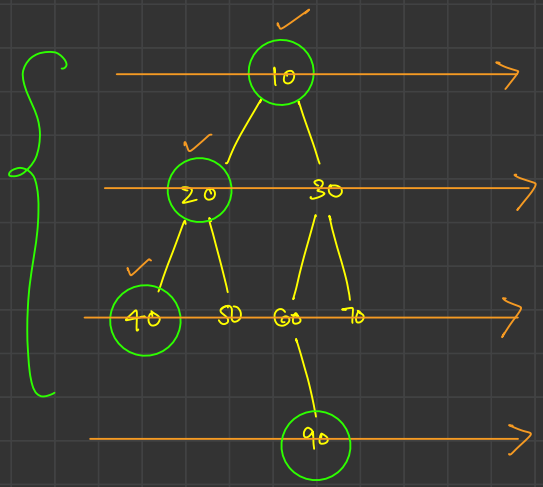
level = 3

o/p

10
30 20
40 50 60 70
90 80

Left View





o/p

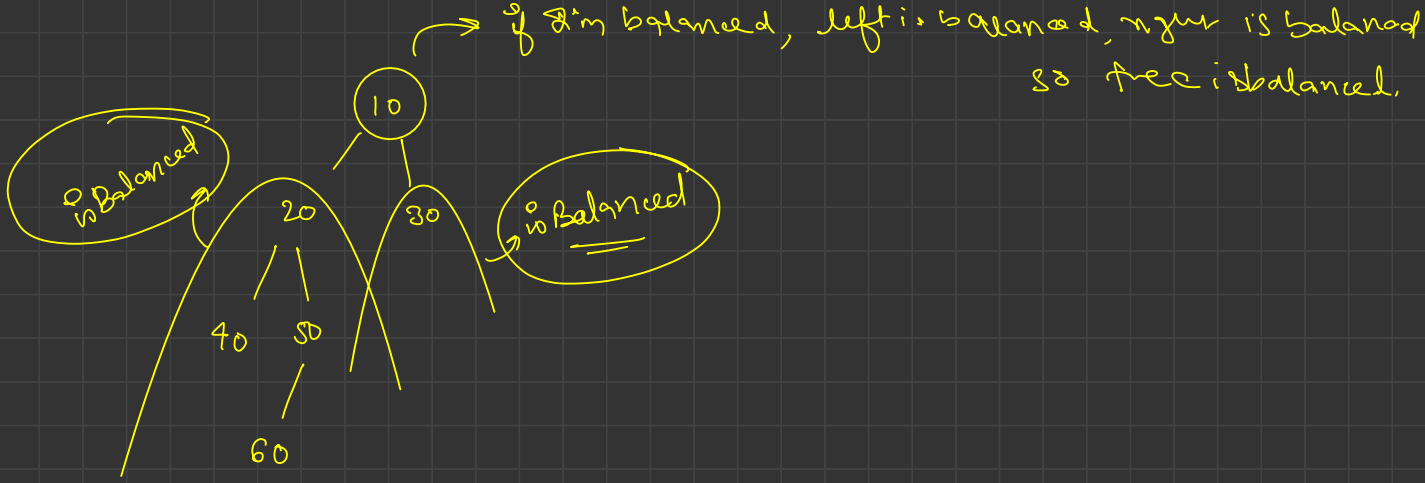
10, 20, 40, 90

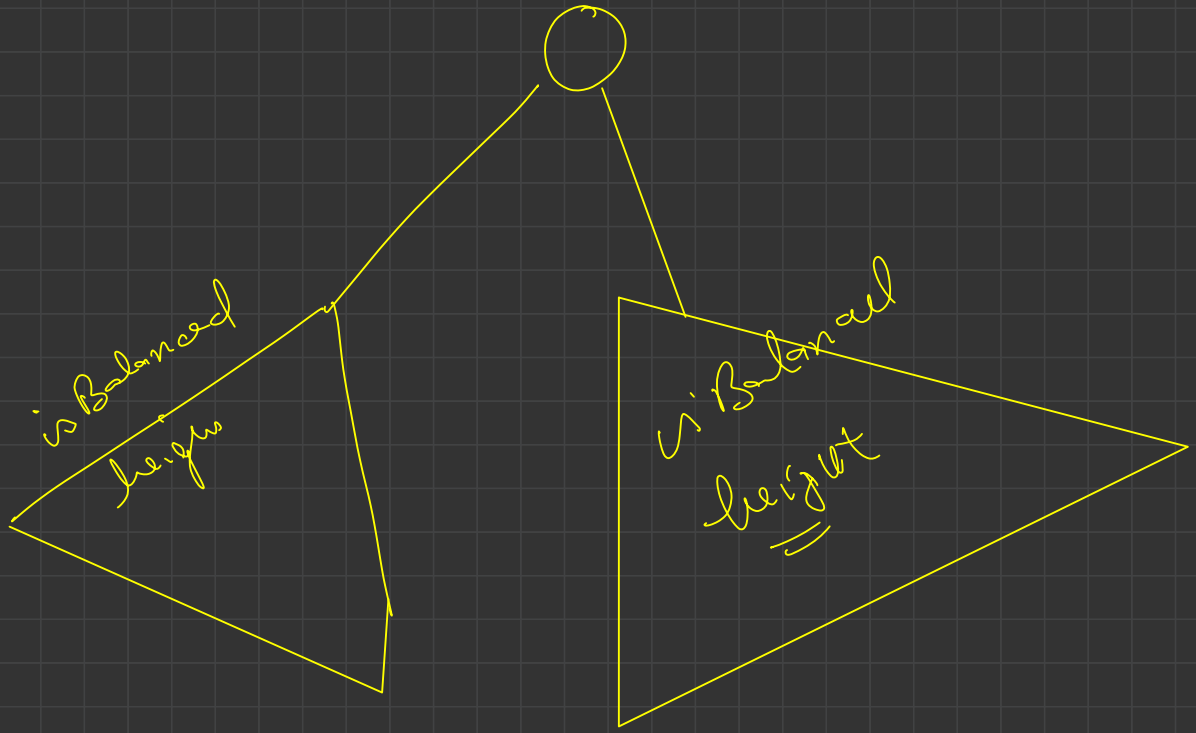
first Node each level.

Balanced Binary tree

$|lh - rh| \leq 1$ → to tell if a node is balanced.

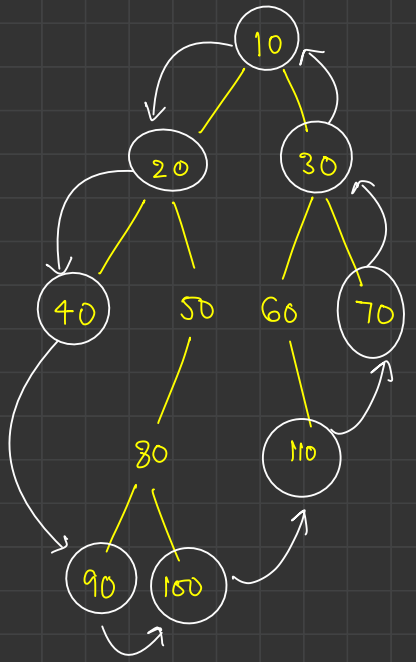
→ when each node is balanced!


If 10 is balanced, left is balanced, right is balanced
so tree is balanced.
10
20
30
40
50
60
isBalanced
isBalanced


is Balanced
height
is Balanced
height

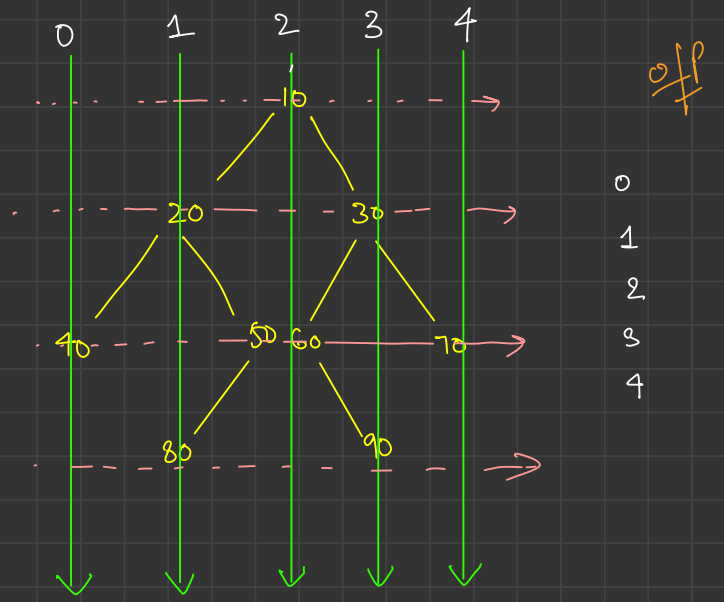
Boundry Traversal

left wall + bottom wall + right wall

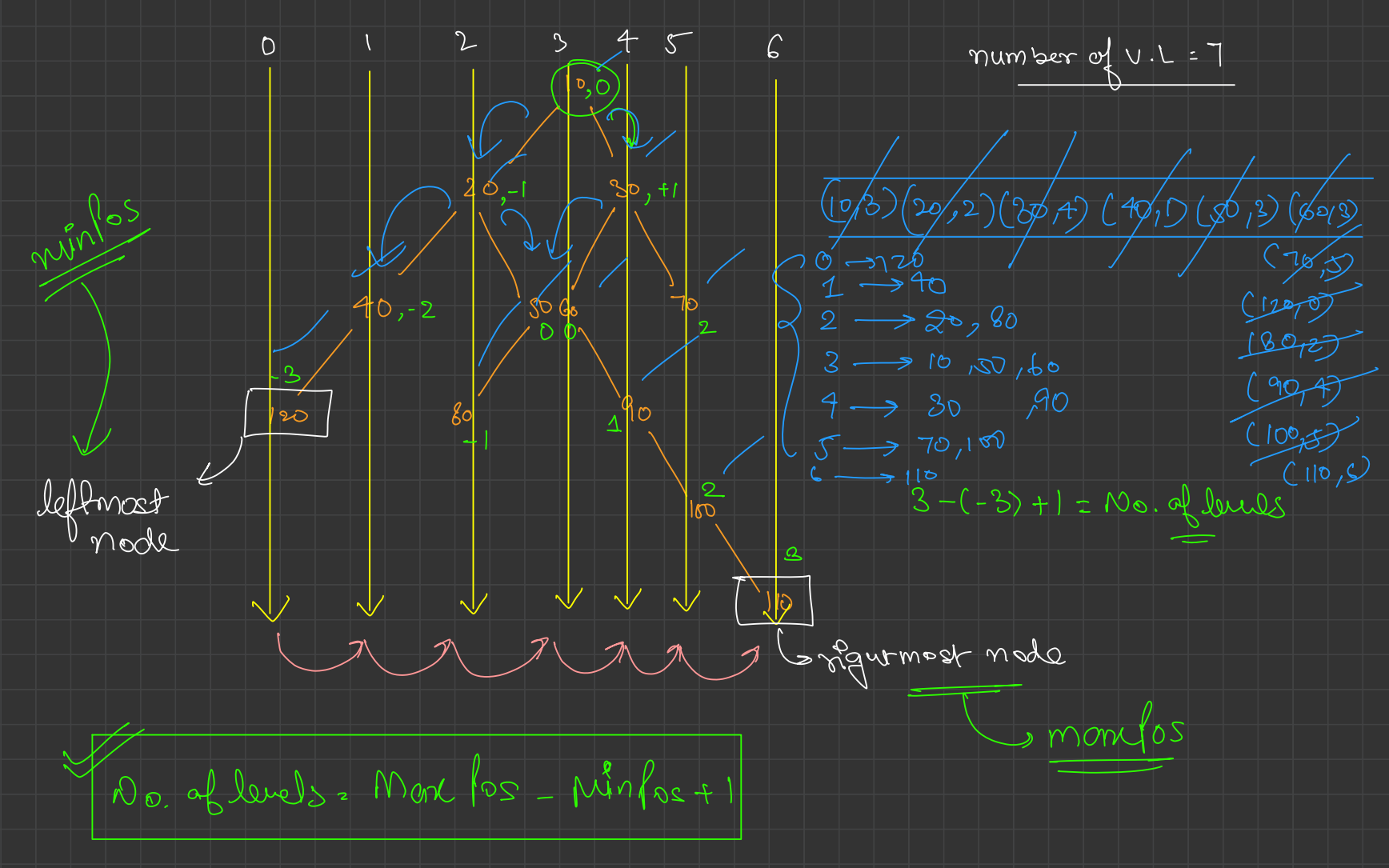


{10, 20, 40, 90, 100, 110, 70, 80}

Vertical Order Traversal



① Calc. no. of vertical levels


0 1 2 3 4 5 6
number of V.L = 7
(10,3)(20,2)(30,4)(40,1)(80,3)(60,3)
(70,5)
(120,0)
(180,2)
(90,4)
(100,5)
(110,6)
0 → 120
1 → 40
2 → 20, 80
3 → 10, 50, 60
4 → 30, 90
5 → 70, 100
6 → 110
3 - (-3) + 1 = No. of levels
minPos
leftmost node
rightmost node
maxPos
No. of levels = Max Pos - Min Pos + 1

Priority Queue

- give person with lowest V.O.L.
- if V.O.L. is same then give person with smaller value.