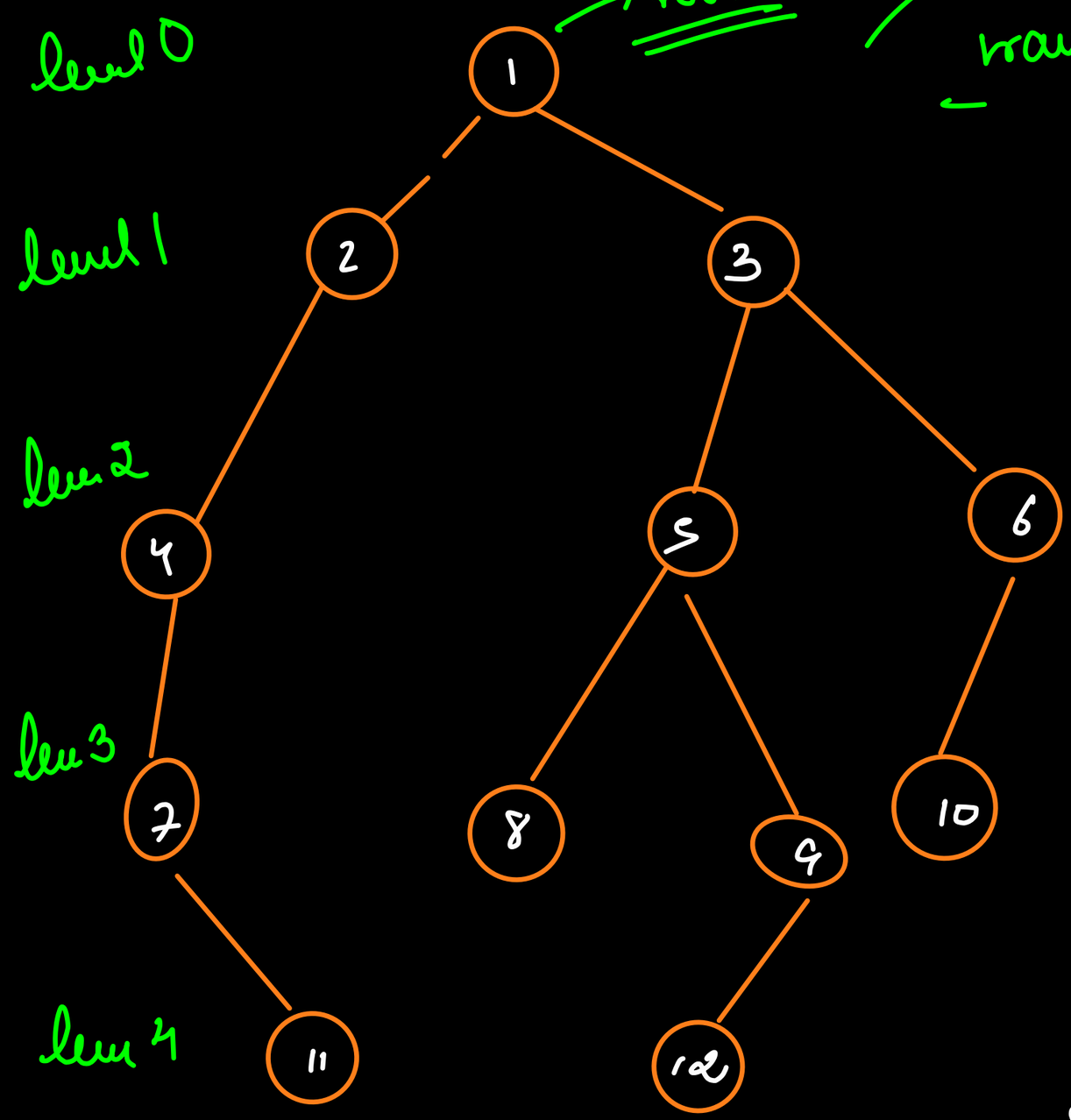


level order traversal

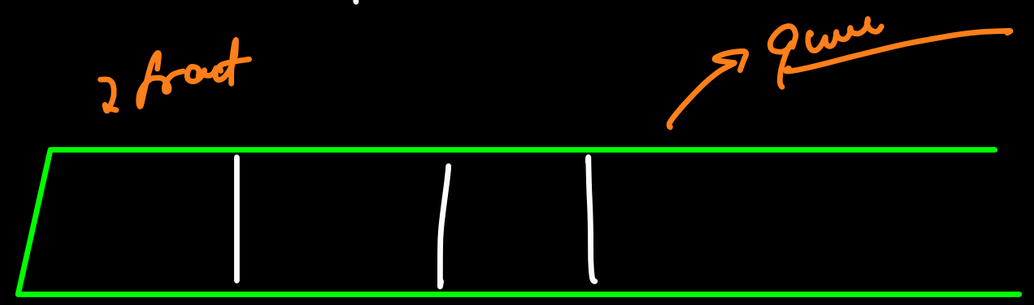
FCFS

BFS

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

assume, (2) will be read

before (3) i.e left child will be read before right child

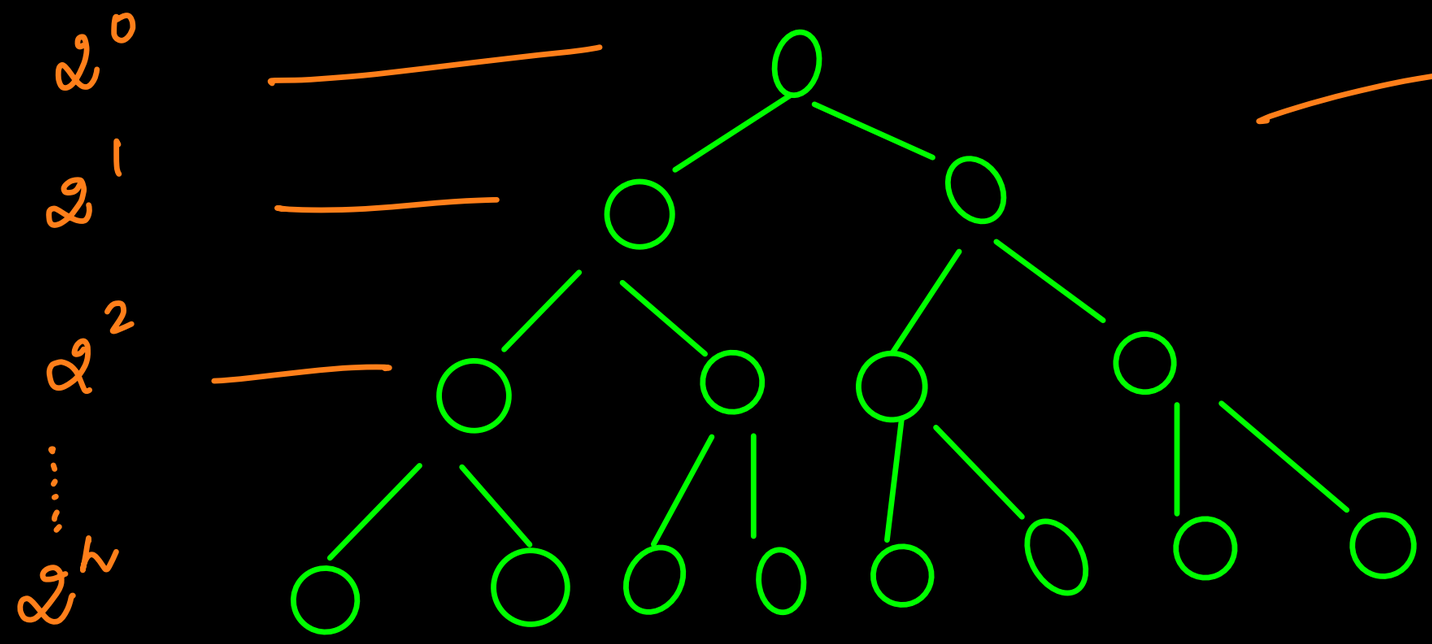


1 2 3 4 5 6 7

8 9 10 11 12

O(n) → O(n) Space

```
qu.enqueue(root);
while(not qu.empty()){
    front = qu.getfront();
    qu.dequeue();
    if(front.left != null)
        qu.enqueue(front.left);
    if(front.right != null)
        qu.enqueue(front.right);
    print(front.val);
}
```



→ Total n nodes

2^h → no. of nodes at last level

} How many nodes are at the last level

Total no. of nodes in a perfect bt

↓

n

$$2^0 + 2^1 + 2^2 + 2^3 \cdots\cdots 2^{h-2} + 2^{h-1} + 2^h = n$$

GP

$\frac{a(r^n - 1)}{r - 1}$

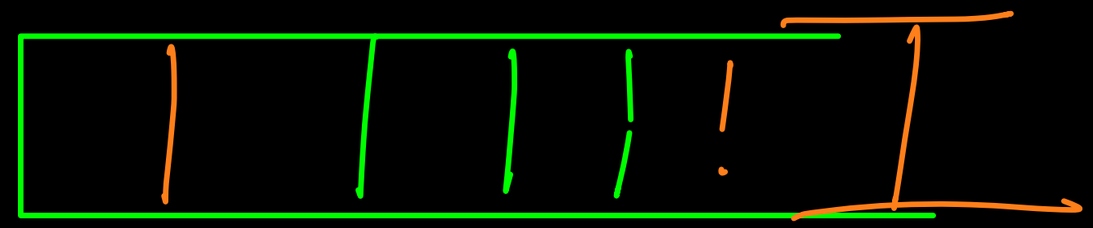
$$\frac{2^0 \times (2^h - 1)}{2 - 1} + 2^h = n$$

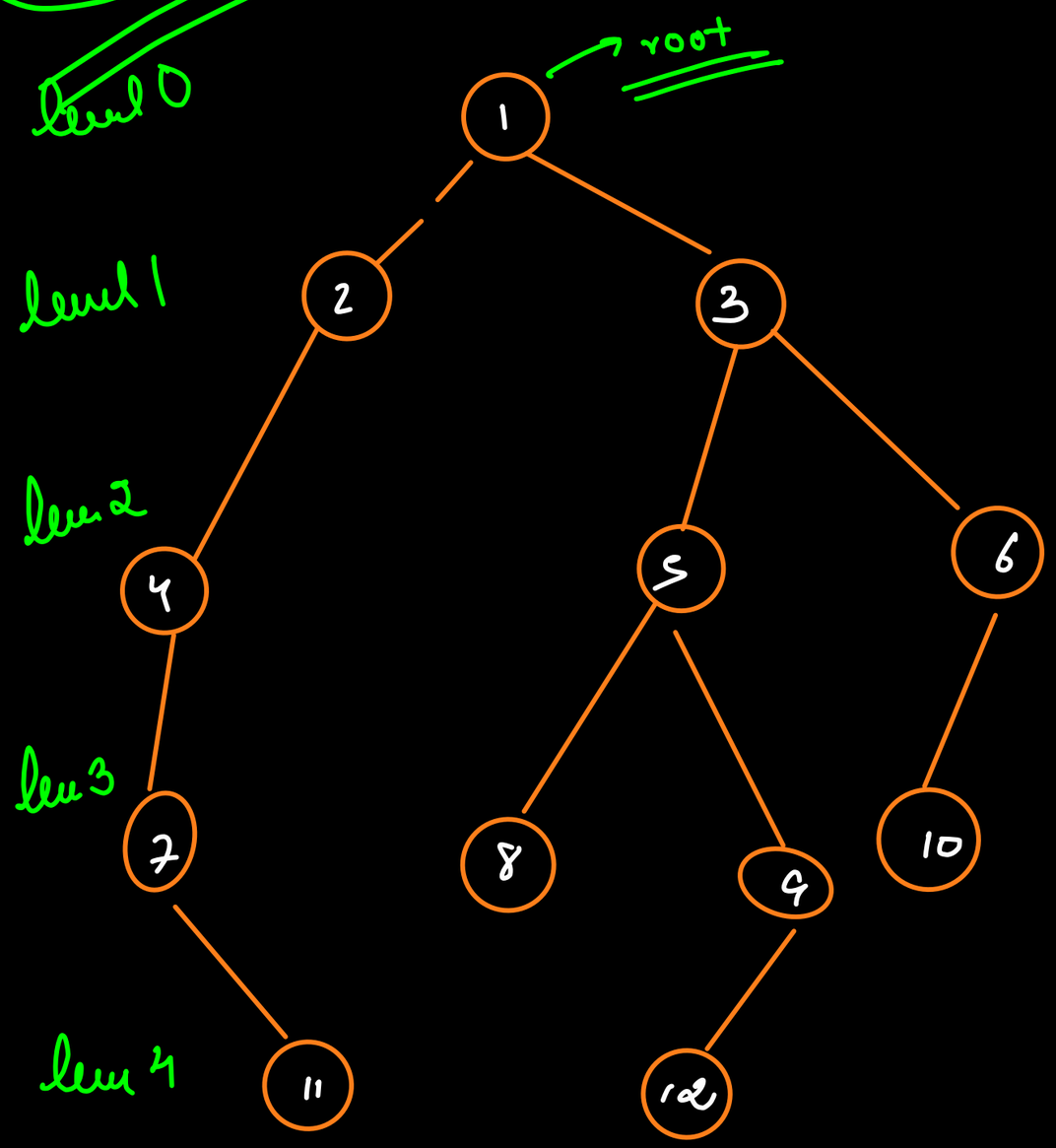
$$2^h - 1 + 2^h = n \Rightarrow 2 \times 2^h = n + 1$$

$$2^h = \frac{(n+1)}{2}$$ → approx $\log_2$

level order
level wise

marker to represent end of a level

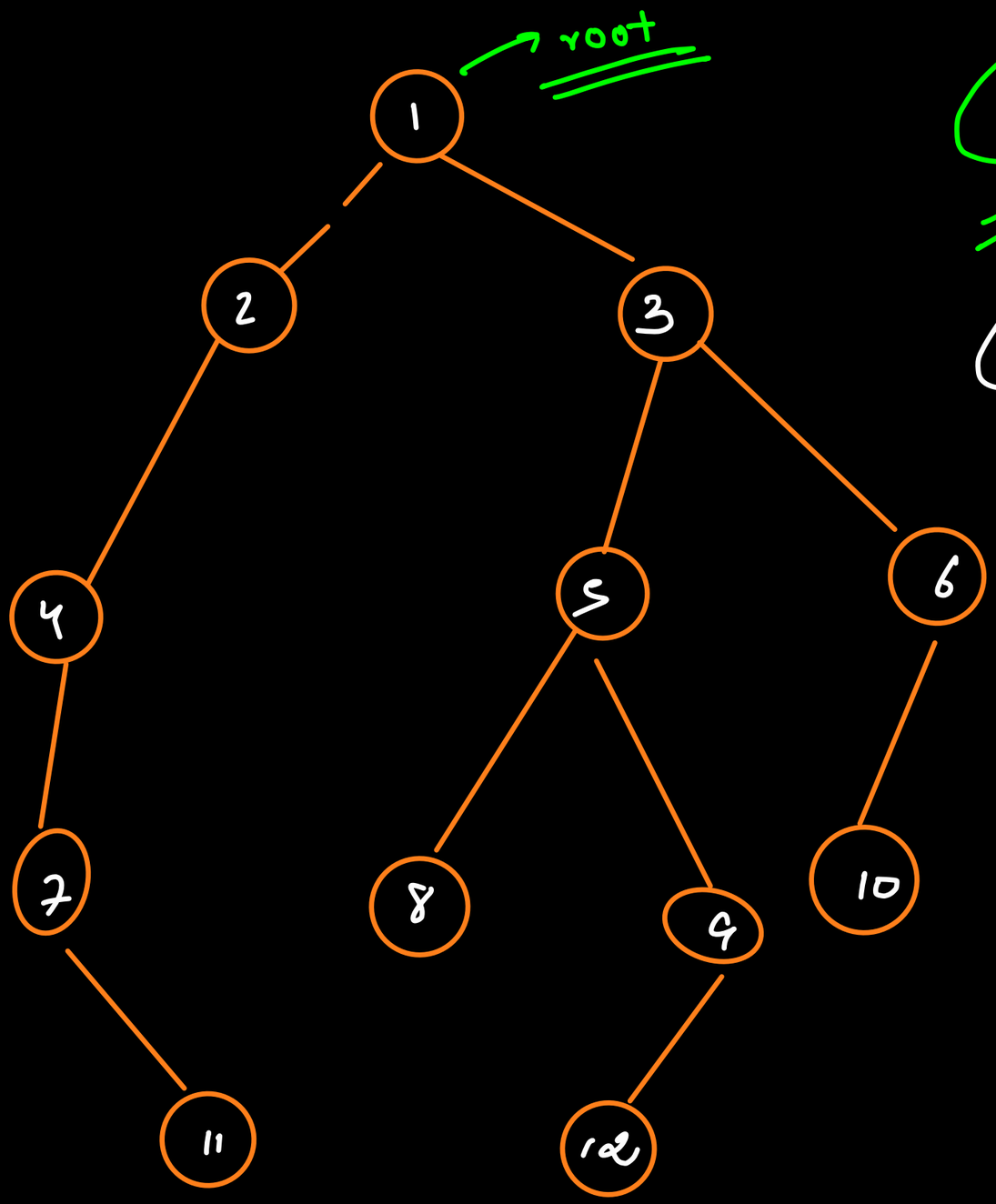


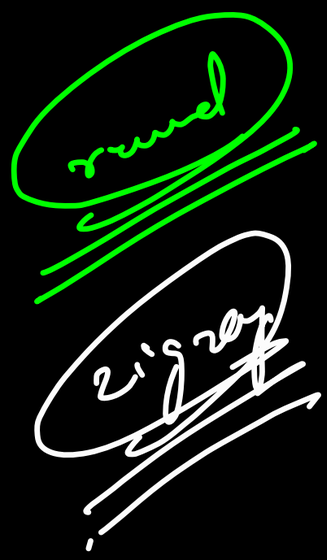


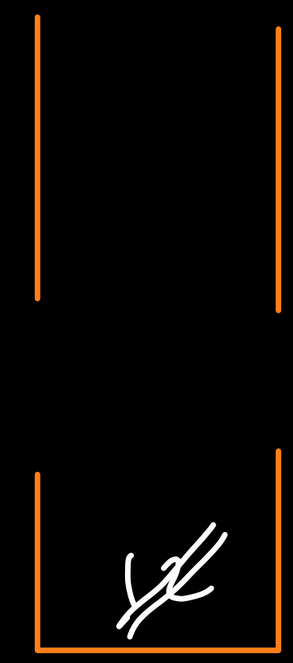
1 null 2 3 null 4 5 6 null
7 8 9 10 null 11 12 null

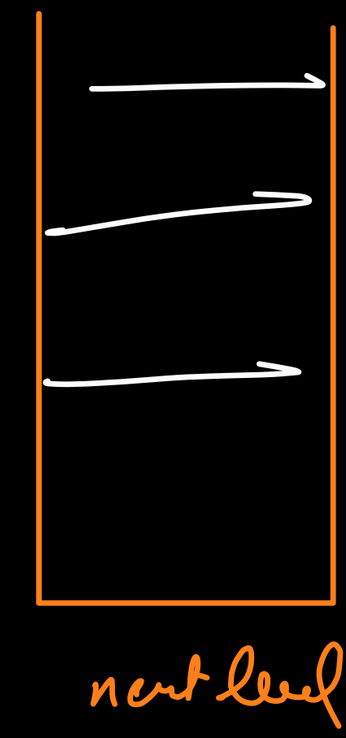
everytime when you remove an null, we only have elements of next level.

JOIN THE DARKSIDE









O(n)

O(n)

next level

curr

count - 1 2 3 4 5

right → left

then left → right

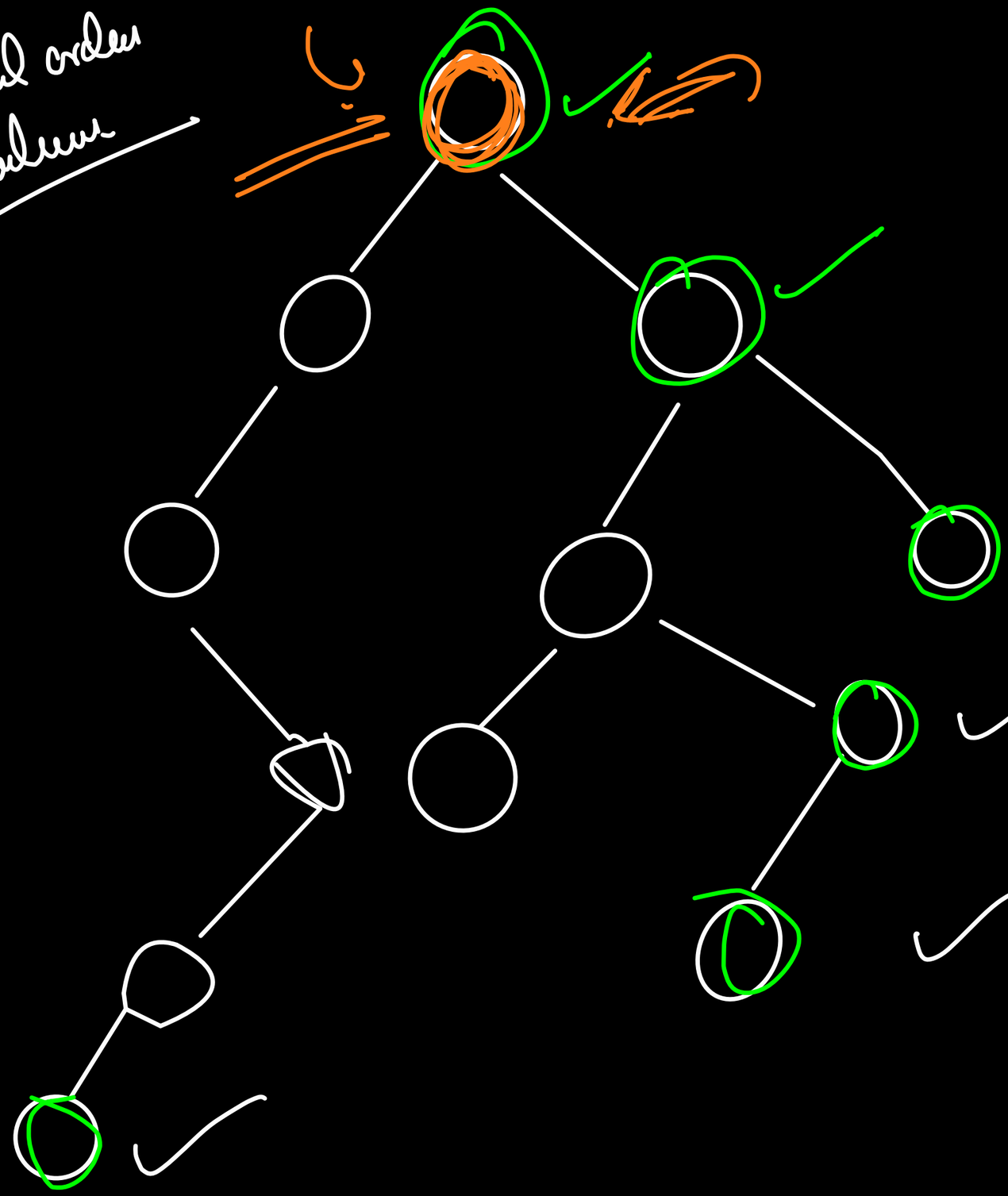
left → right

then right → left

1 3 2 4 5 6 10 9 8 7

11 12

level order traversal



level order

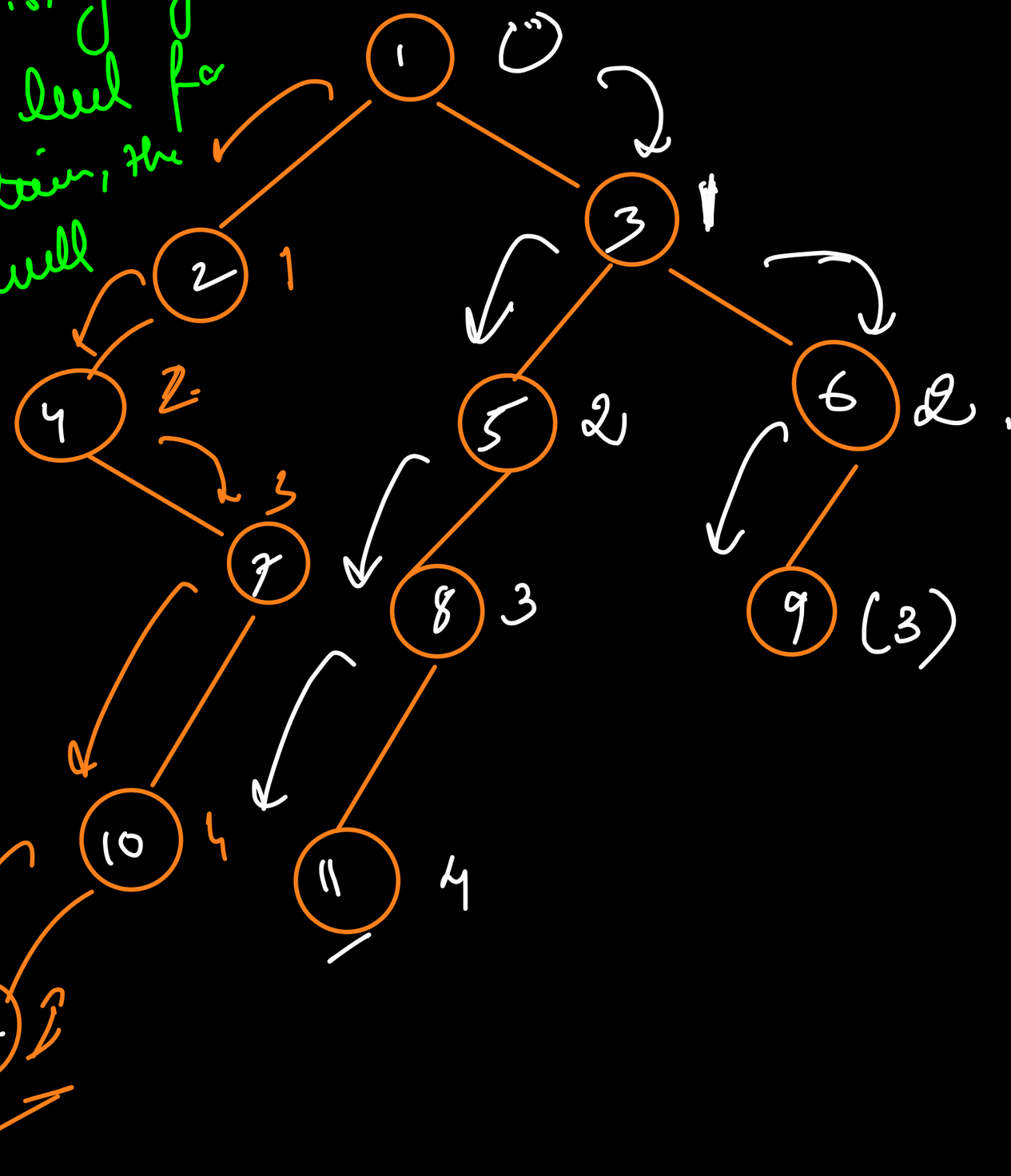
O(n)

O(n)

null

left

if we are visiting any particular level for the first time, the curr node will be part of



Right

Recursive code.

1, 3, 6, 9, 11,

12

Right

Left

max_level = 0 1 2 3 4 5

Global

curr_level = 0

recursive

(cl, ml)

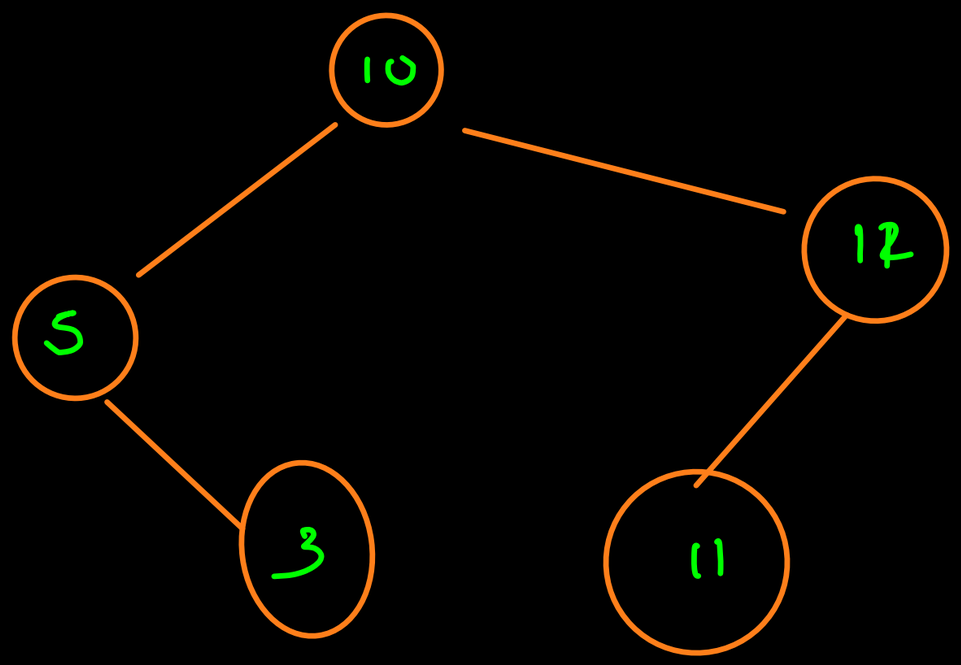
$f(root, cl) =$

```
if (cl > ml)  {
    ml = cl
    rout.push(root.val)
}
f(root.right, cl+1)
f(root.left, cl+1)
```

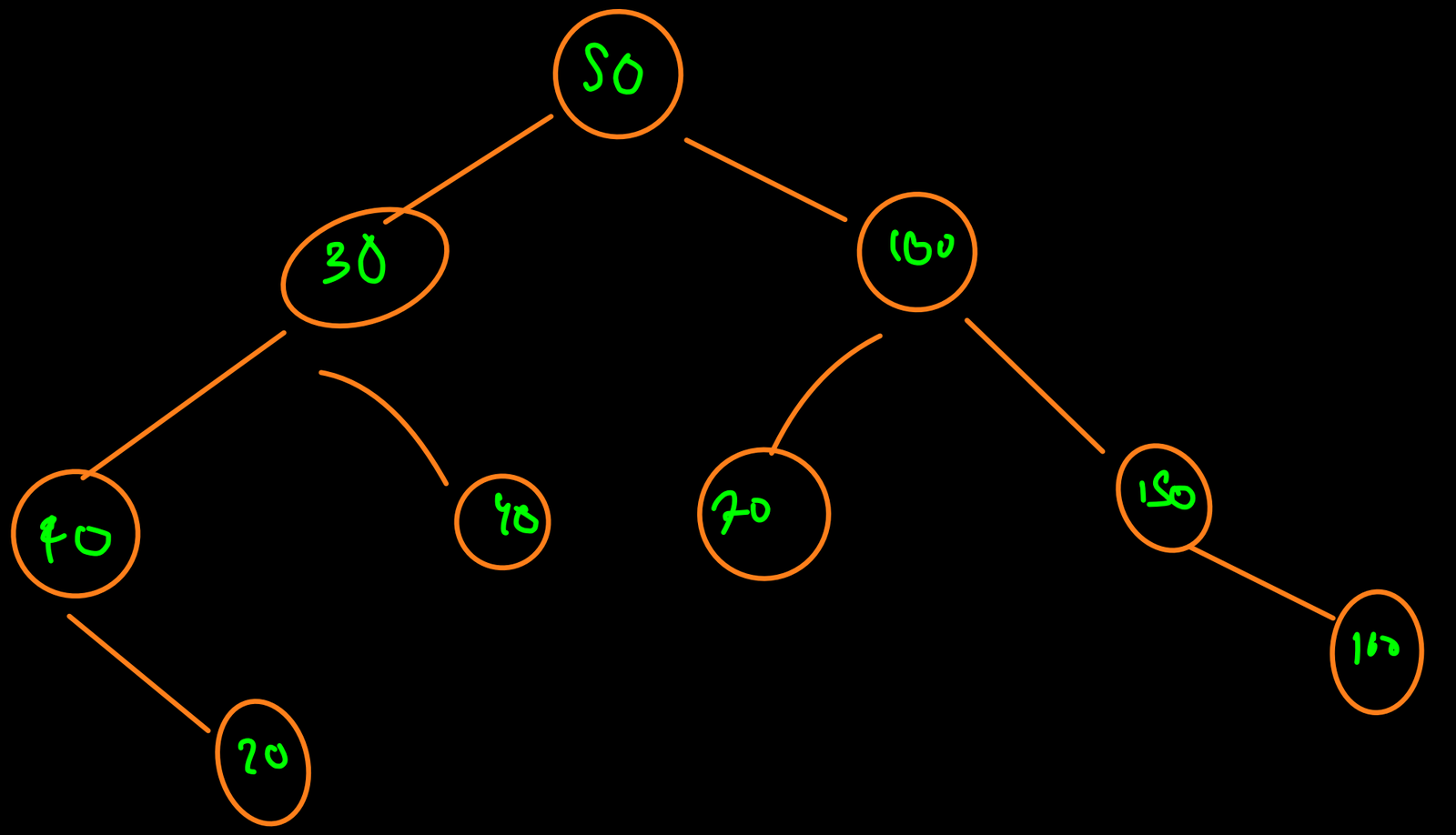
BST → Binary Tree

- all the elements of lst < root
- all the elements of rst > root

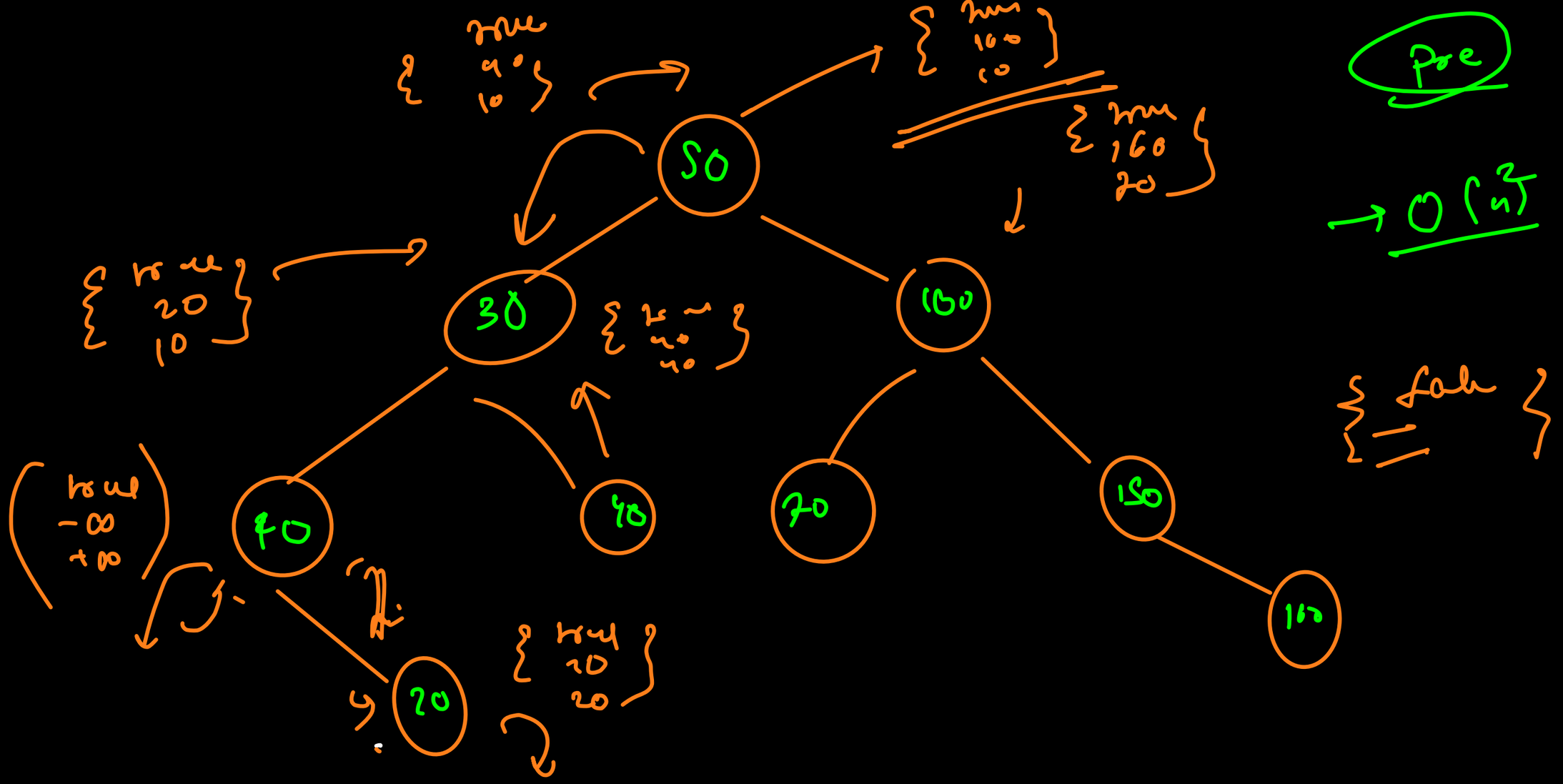
recursively



Property 2 → Inorder traversal of a BST is always sorted.



10 20 30 40 50 70 100 150 160


50
30
100
40
70
150
10
20
Pre
O(n)
O(n)
JOIN THE DARKSIDE

f(root).isBST = l - f(root.left)

r - f(root.right)

{

max

min

isBst

if (l.isBST and r.isBst and

root.val > l.max and root.val < r.min)

return { isBst : true

max : (l.max, r.max) root.val

min

}

}

root == null

→

{

isBST → true

max → −∞

min → +∞

}