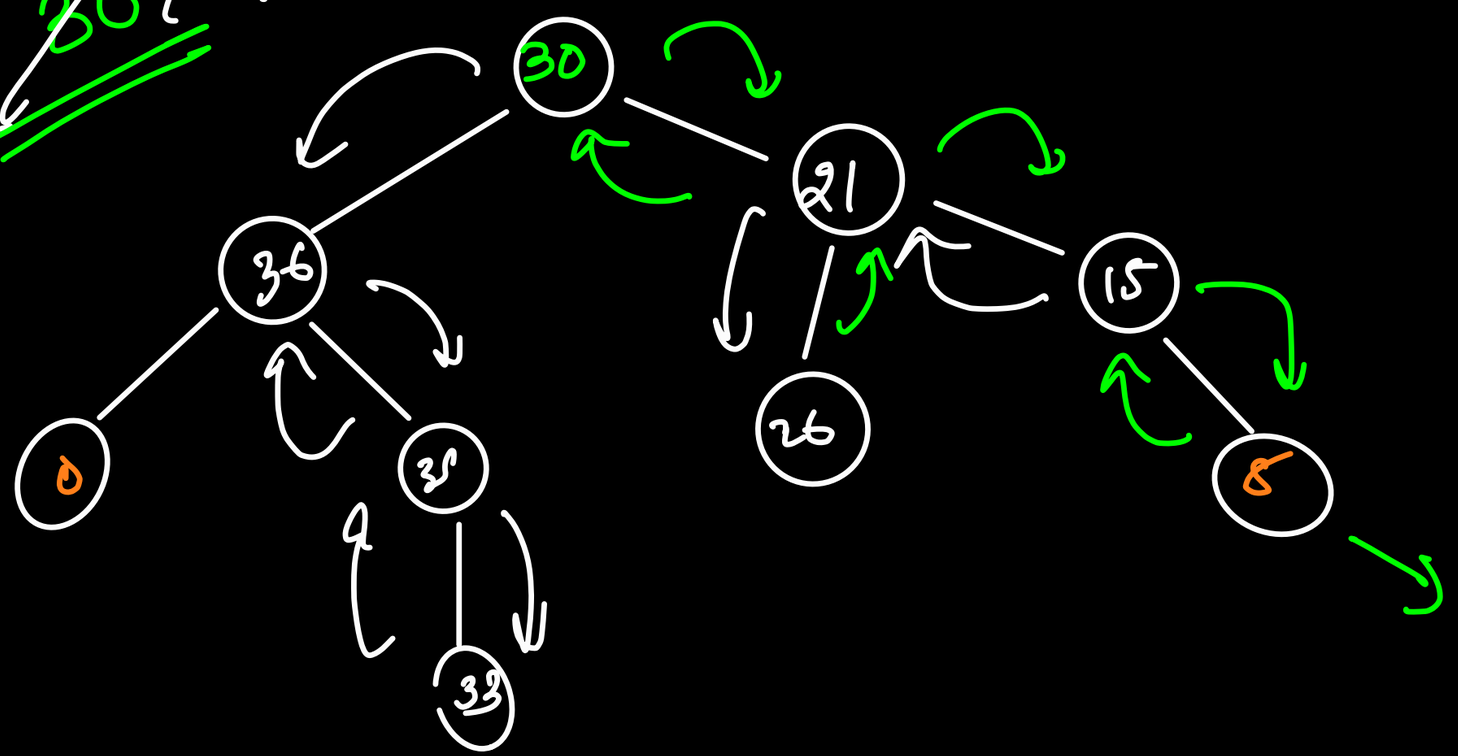
global sum = 30 63 35



Reverse Inorder

RST
Root
LST

JOIN THE DARKSIDE

f(root) = f(root.right)

↳ recursive

root.val += globalSum

globalSum = root.val

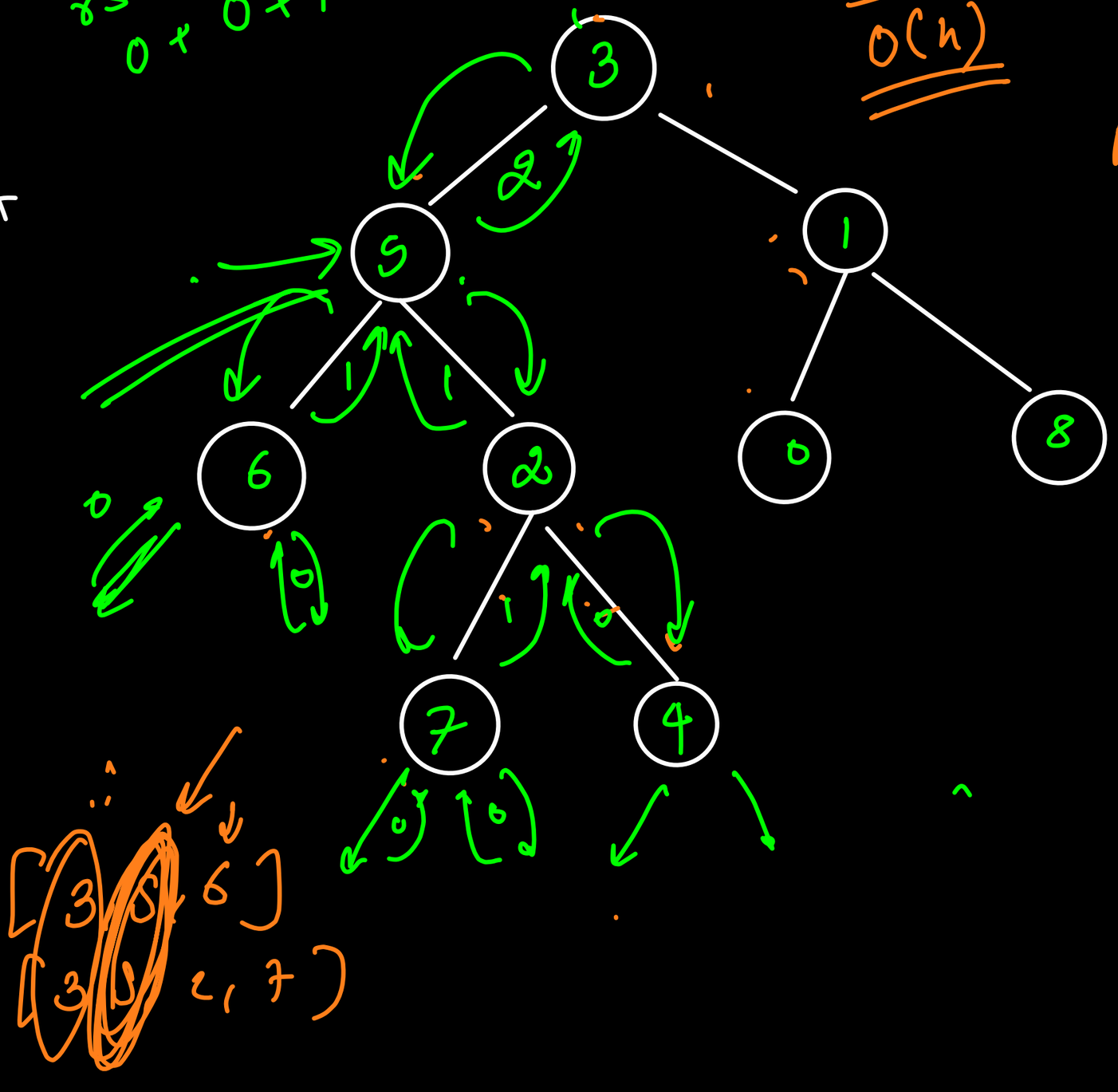
f(root.left)

JOIN THE DARKSIDE

LCA

rst + lst + self
0 + 0 + 1

path from root



O(N)
O(h)

7 ← p, q → 6

path from root to p
↳ path p →

path q →

path [7, 2, 5, 3]

(3, 5, 2, 7)

[3 5 6]
[3 5 2, 7]

[0, 1, 3] (3, 1, 4)

JOIN THE DARKSIDE

0 \
1 → f(root, p, q)
2 /

↓

This fun^ returns 0
if none of p or q are present in
subtree of root,
it returns 1 if any one of p or q
is present in subtree, & it
returns 2. if both of them are
present in root's subtree

JOIN THE DARKSIDE

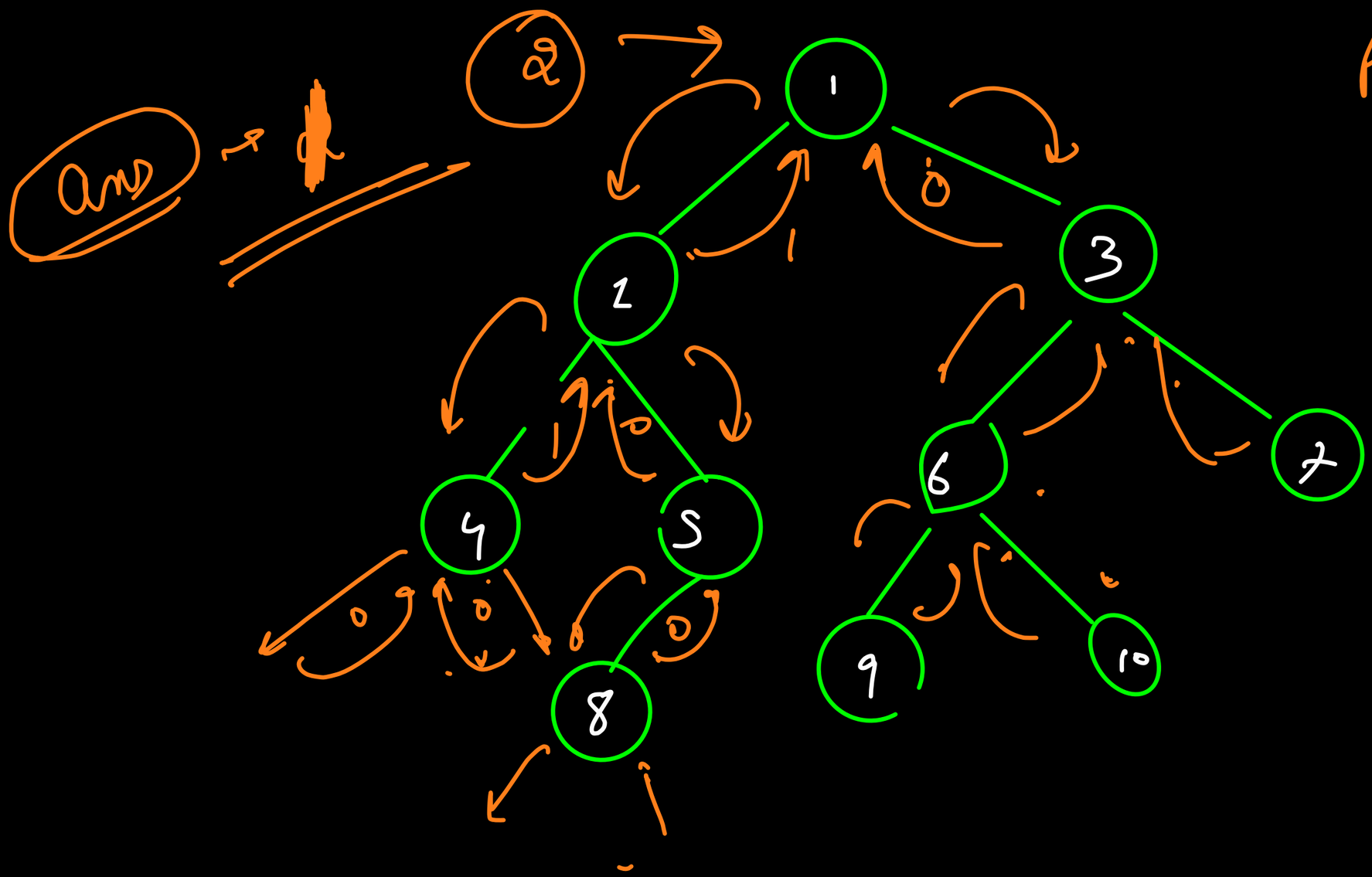
$f(root, p)$ =

↓

boolean if a path is found from root → p

```
if (root->val == p) {
    path.push(root->val);
    return true;
}
lans = f(root.left, p)
if (lans == true) {
    path.push(root.val)
    return true;
}
rans = f(root.right, p)
if (rans == true)
    path.push(root.val)
    return true
}
return false;
```

JOIN THE DARKSIDE



JOIN THE DARKSIDE

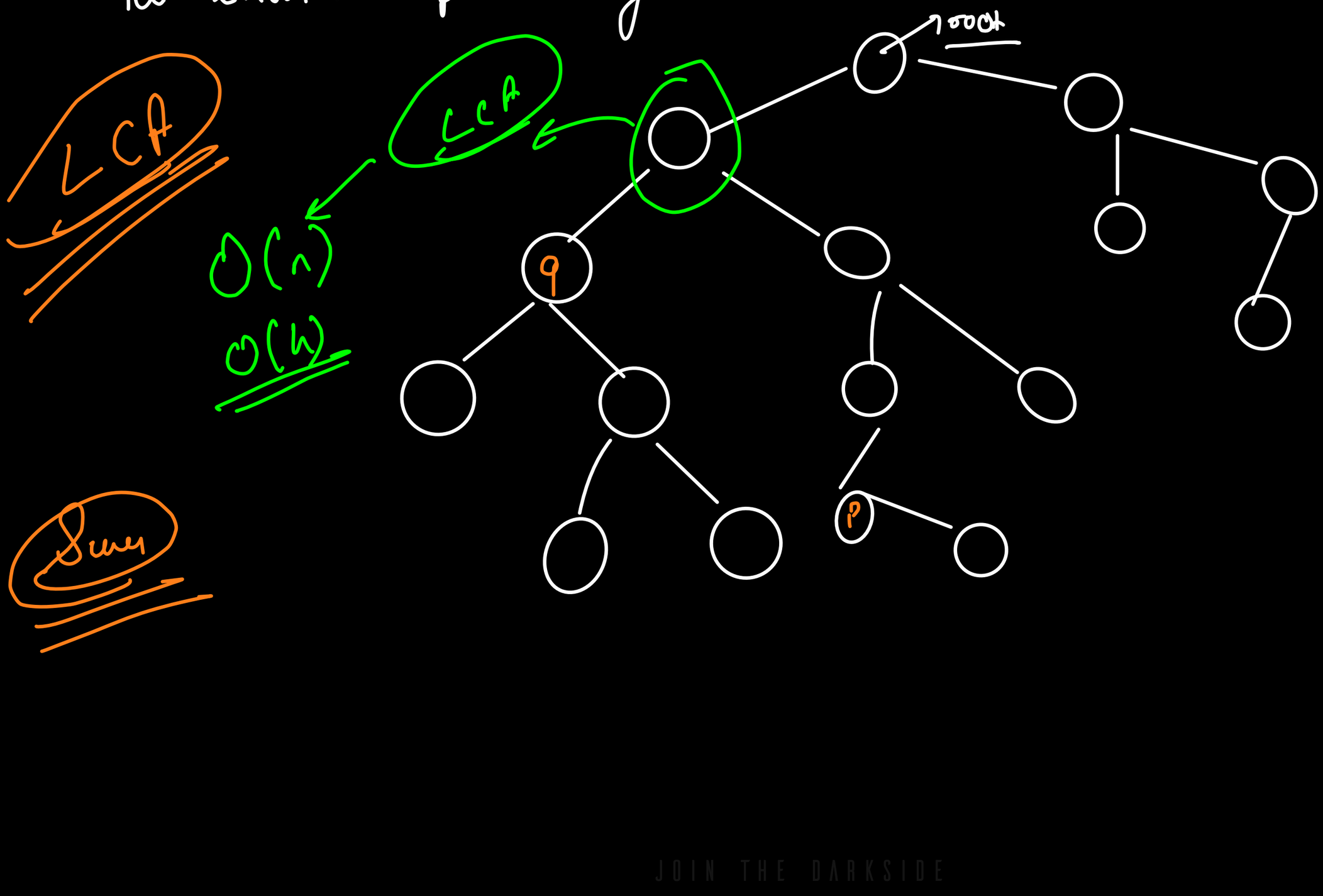
ans = -1

f(root, p, q) ⇒

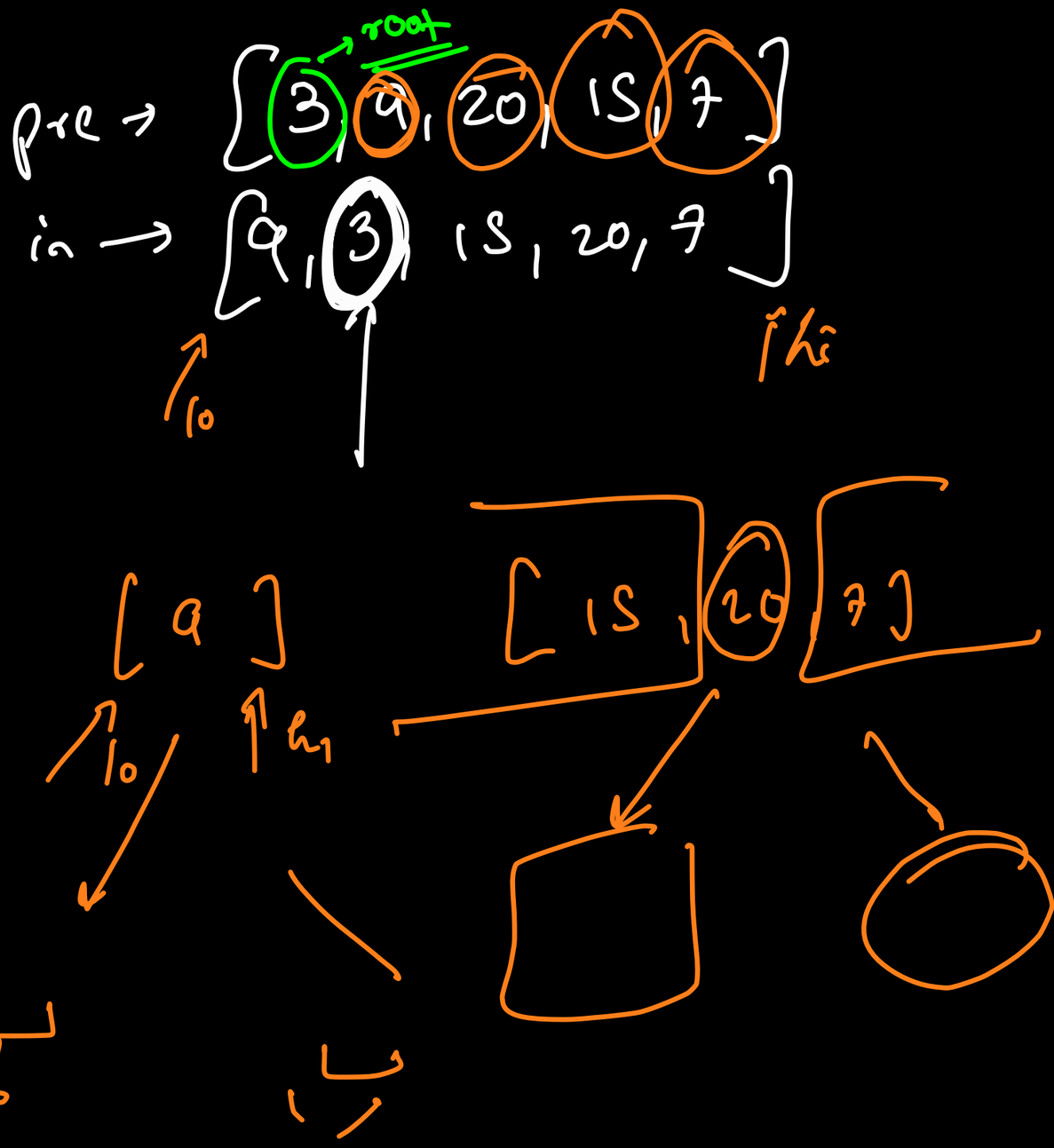
```
if(ans != -1) return 0
lans = f(root.left, p, q)
rans = f(root.right, p, q)
myans = (root.val == p or root.val == q)
if (lans + rans + myans > 0)
        ans = root.val
return lans + rans + myans;
```

JOIN THE DARKSIDE

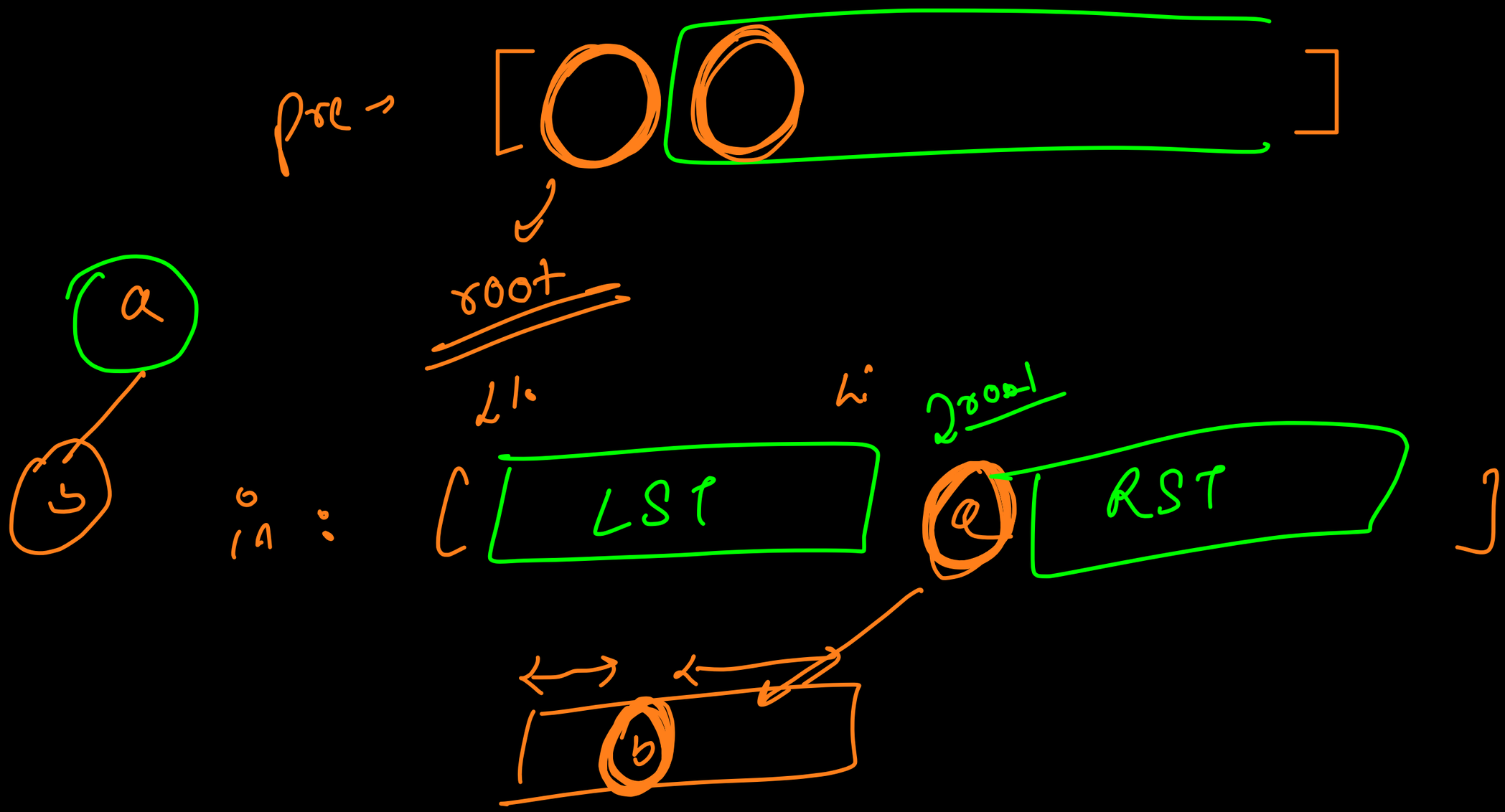
Q. Given a B.T, and 2 nodes p and q, we need to find the shortest path length between the 2 nodes p and q.



JOIN THE DARKSIDE



JOIN THE DARKSIDE



JOIN THE DARKSIDE

pre → [3 9 20 15 7]
in → [9 3 15 20 7]

O(n)

if (l > r) return null;

optimise search el → ind

9	0
3	1
15	2
20	3
7	4

mapping

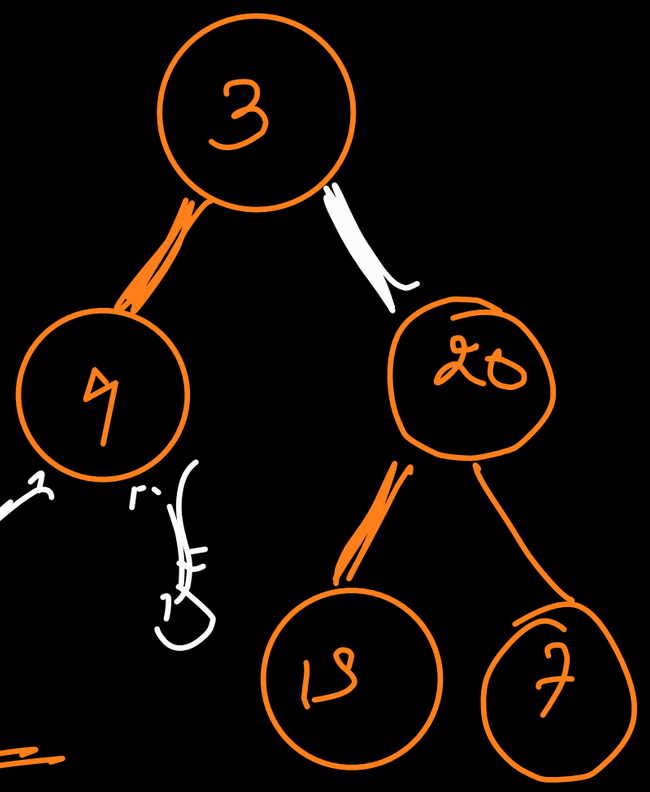
f(0, 4)

f(0, 0)

f(2, 4)

f(2, 2)

f(4, 4)



JOIN THE DARKSIDE

$O(n)$

```
f(root, l, r) {
    if (l == r)
        return new Node(pre[l])

    i++;
    n = new Node(pre[i])
    m = mp(pre[i])
    n.left = f(root, l, m-1);
    n.right = f(root, m+1, r);
    return root
}
```

JOIN THE DARKSIDE