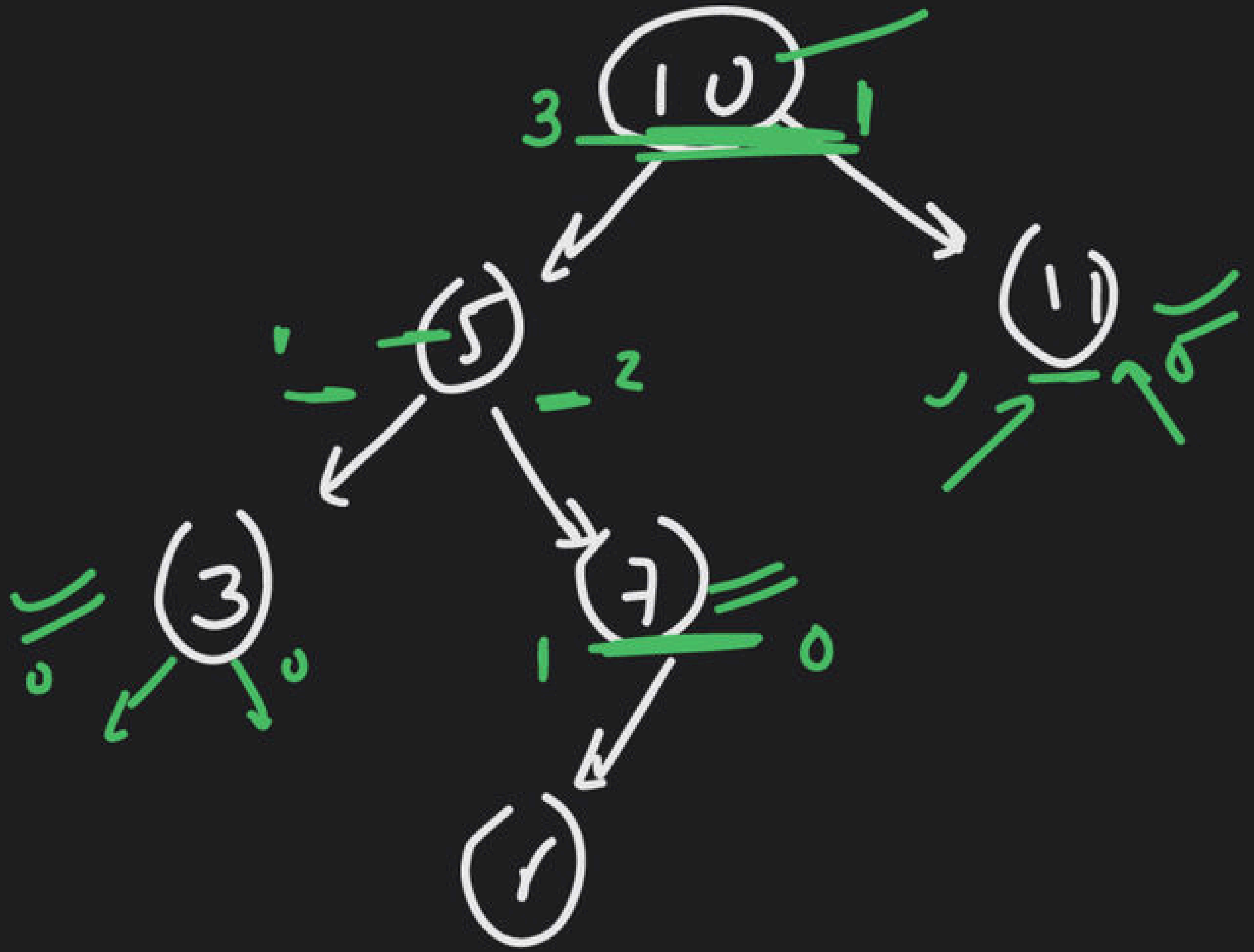
Binary Tree - I

Foundation Course on Data Structures & Algorithms - Part II

Love Babbar • Lesson 7 • July 9, 2022

→ Balanced tree or not



Balanced tree

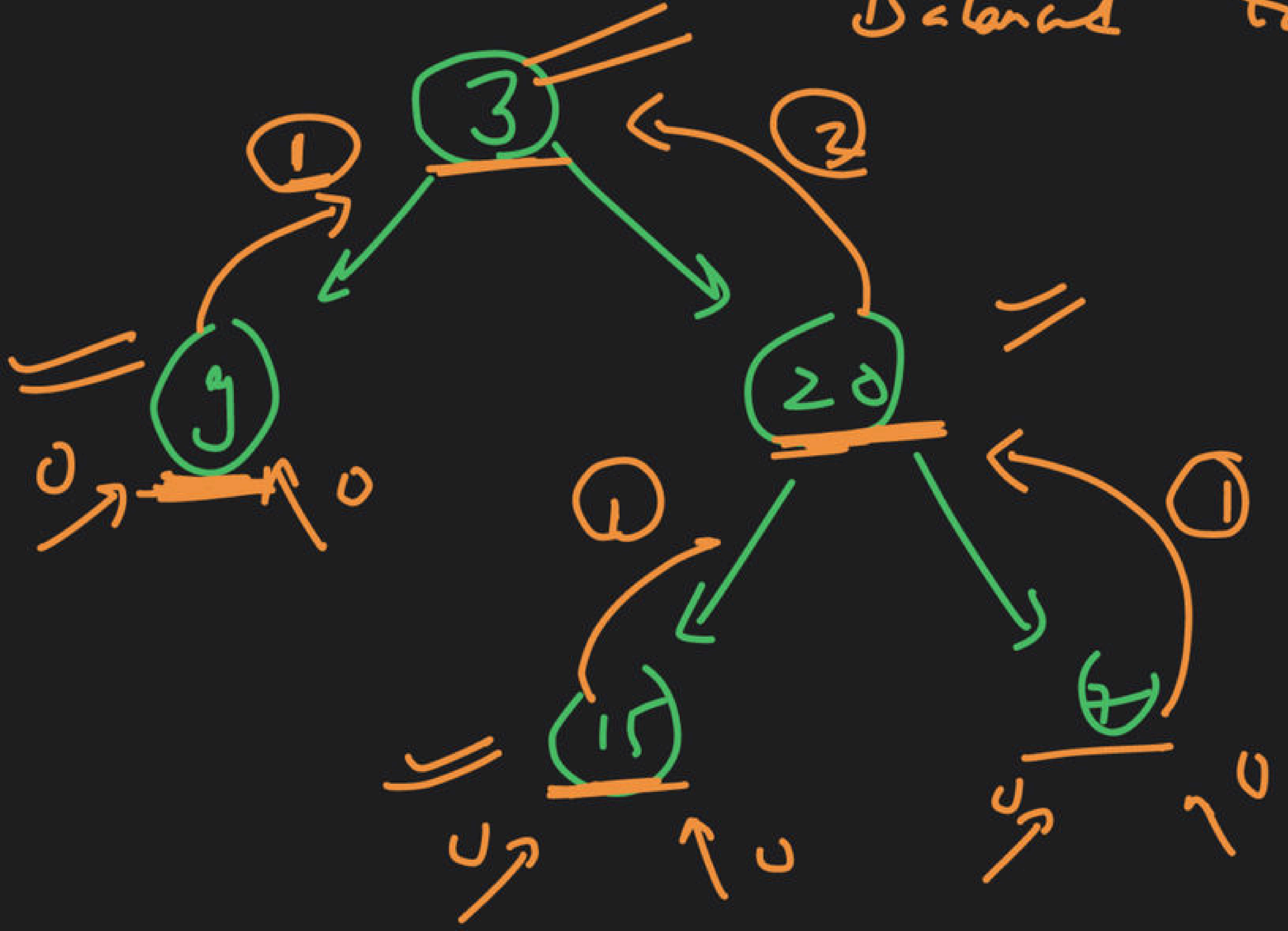
$$\boxed{abs(lh - rh) <= 1}$$

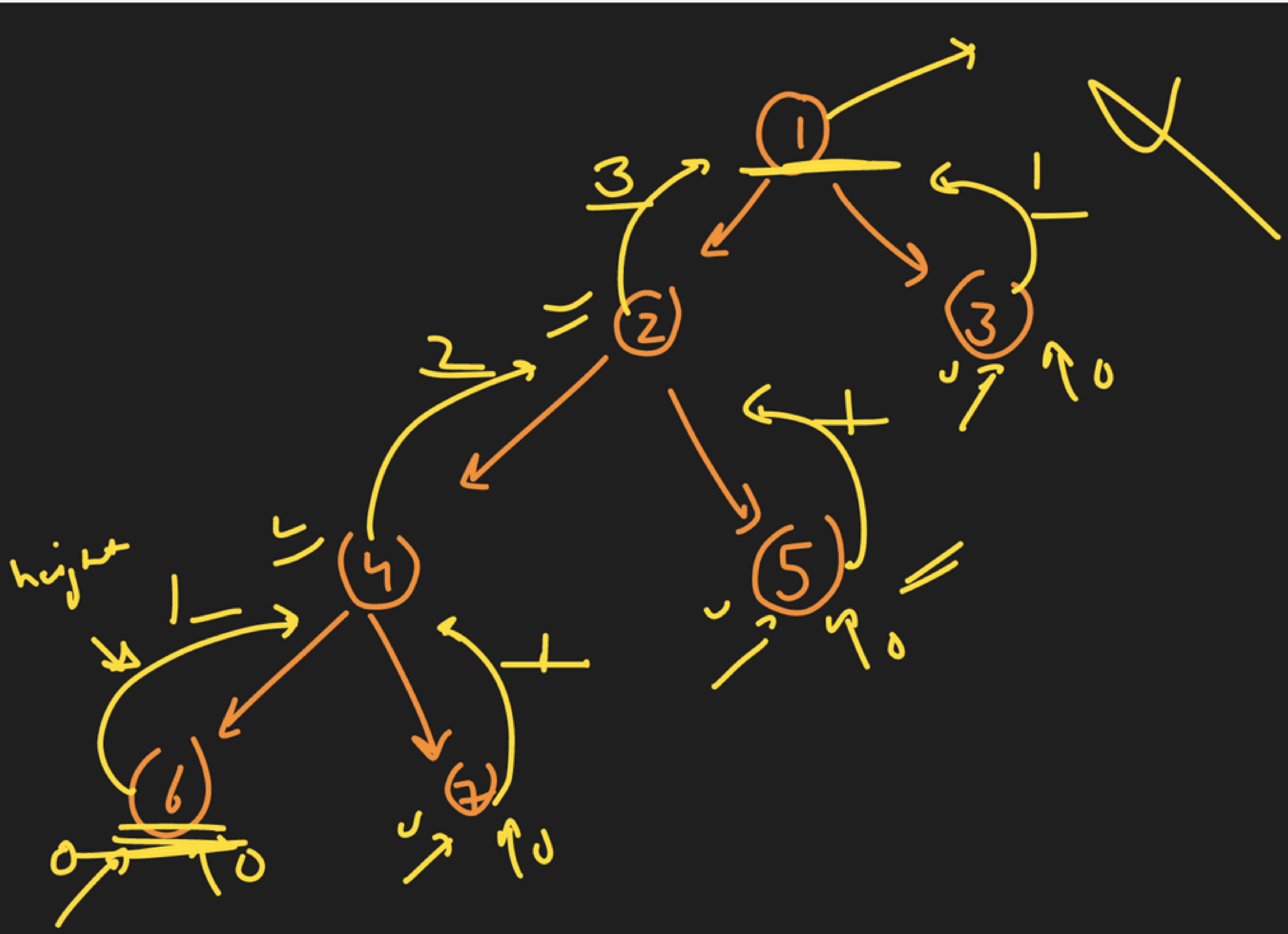
for all nodes

lh → Left sub tree heigh

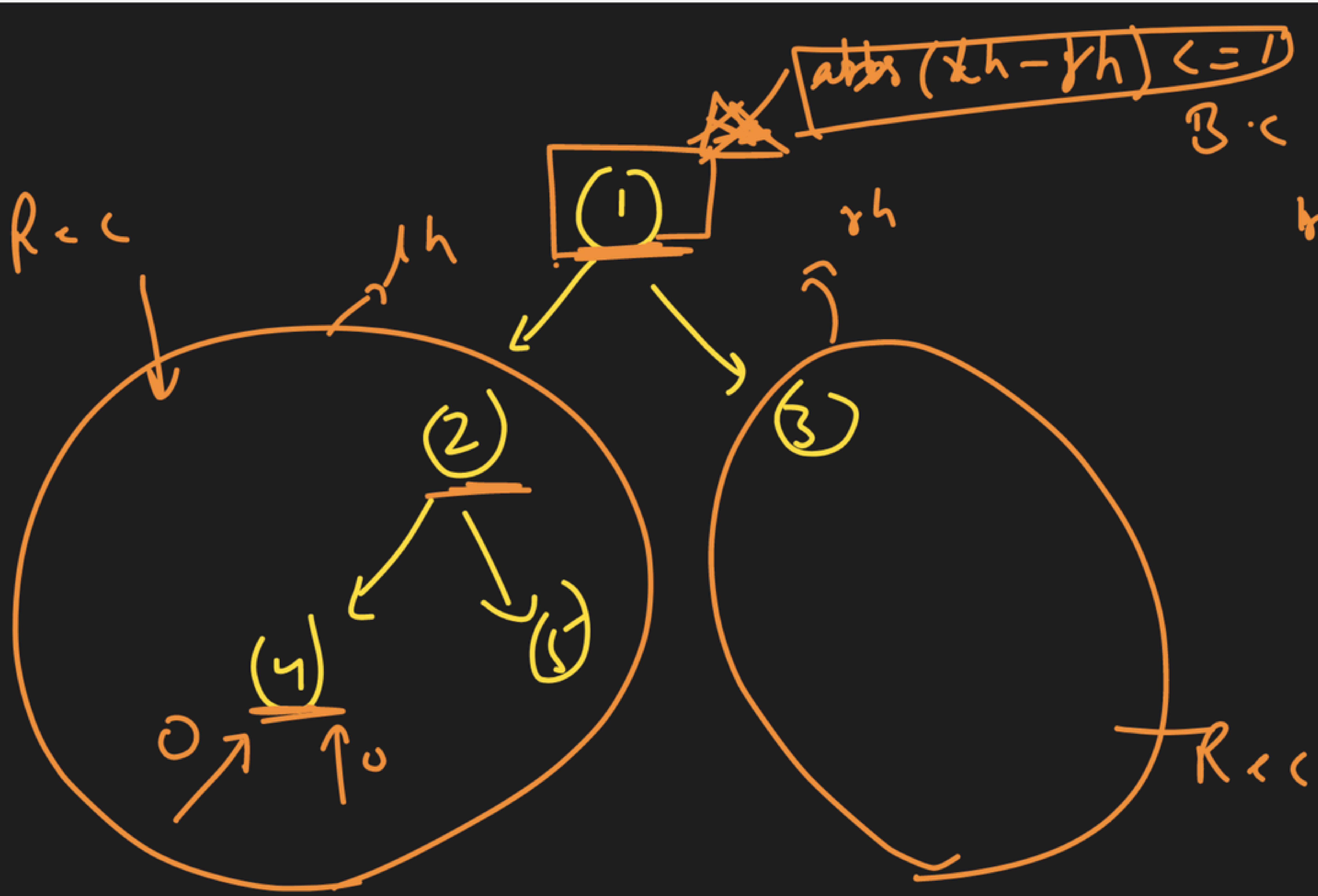
rh → right sub-tree height

Balanced tree

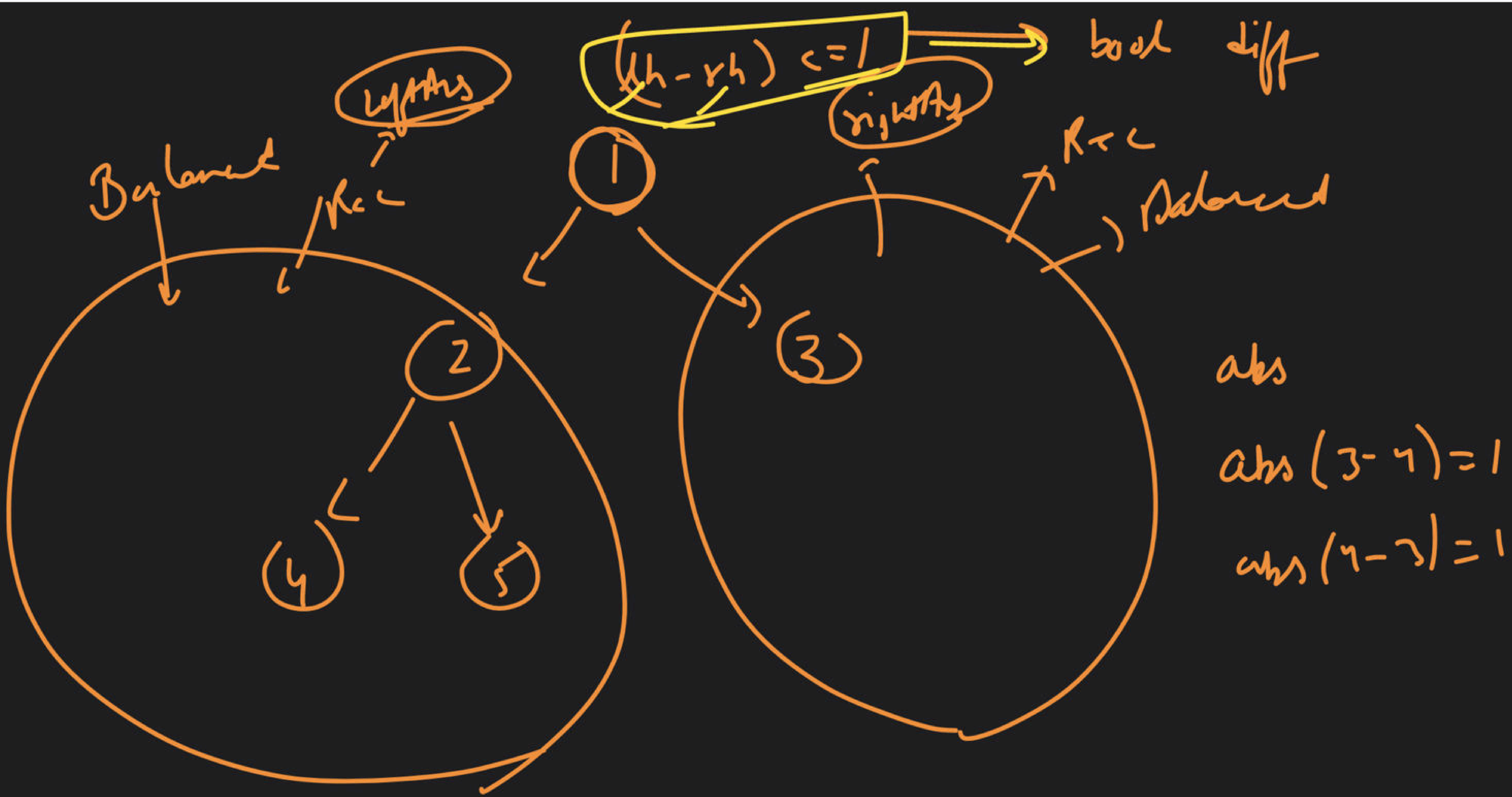




1
2
3
4
5
6
7
3
2
1
1
1
1
height
0
0
0
0
0
0
0
0
0
0



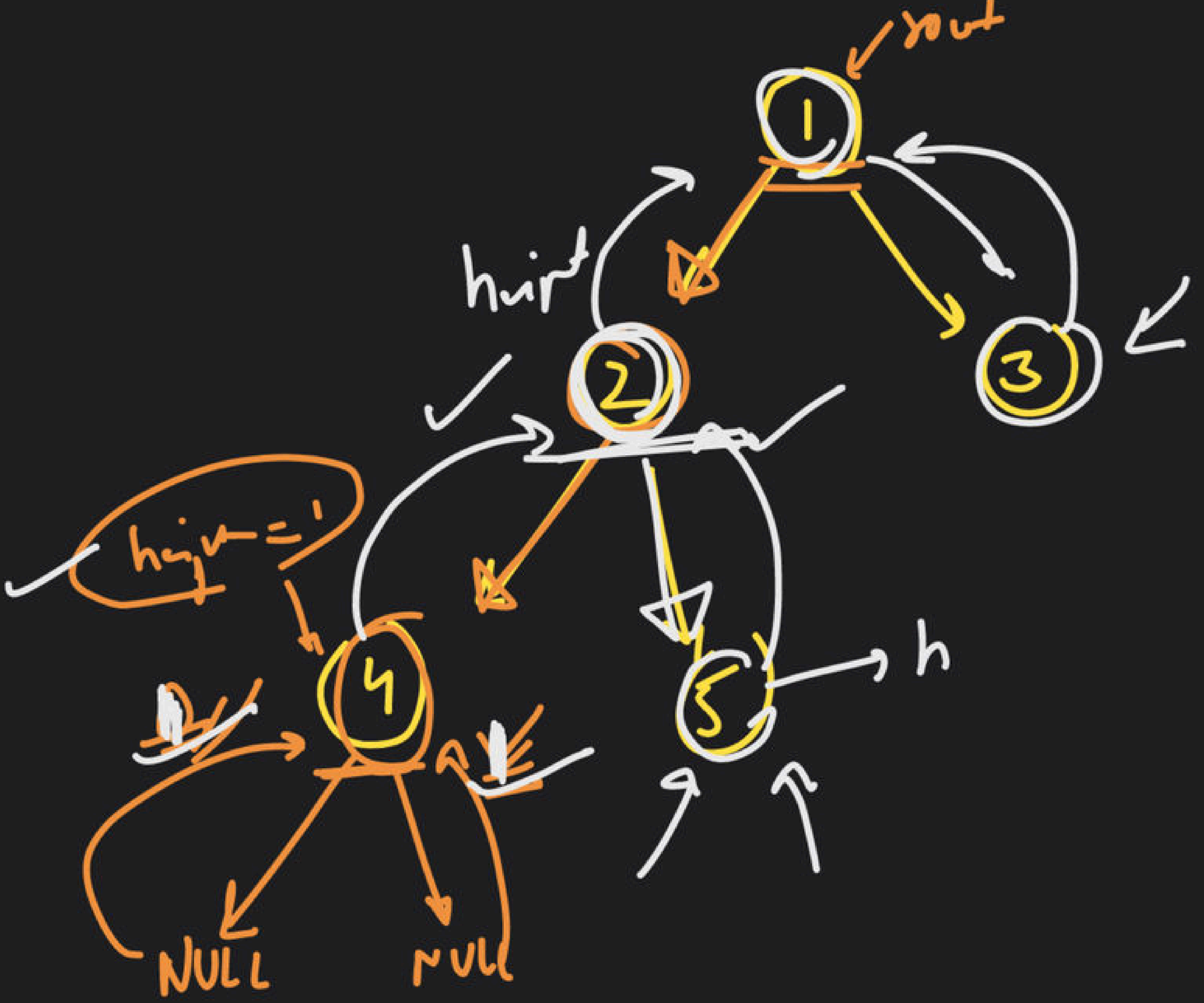
root == NULL
return 0



abs

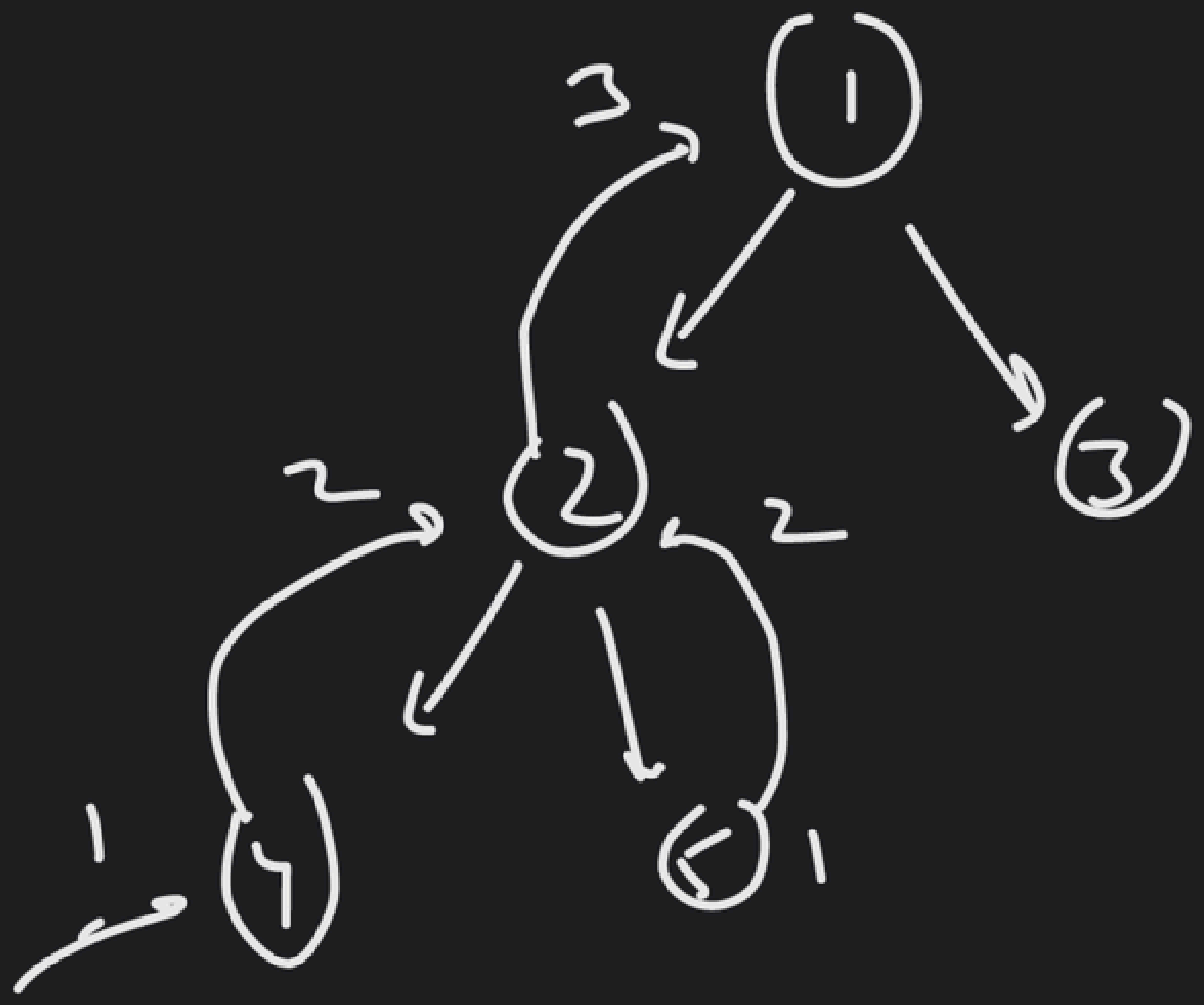
$abs(3-4) = 1$

$abs(4-3) = 1$



T.C ⇒ n^2

α

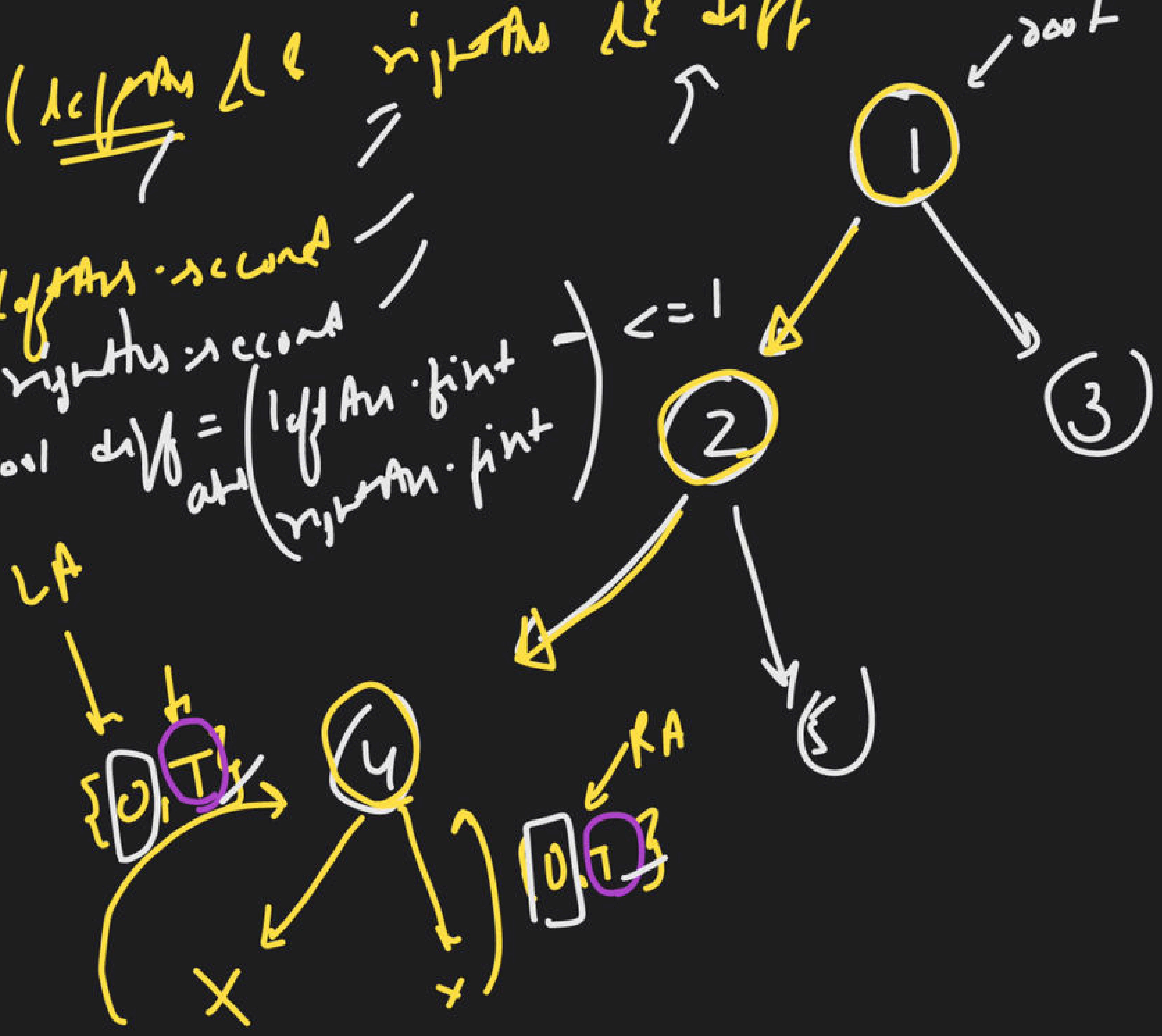

1
2
3
4
5
3
2
2
1
1

if (leftAns && rightAns && diff

leftAns.second

rightAns.second

bool diff = abs(leftAns.first - rightAns.first) <= 1



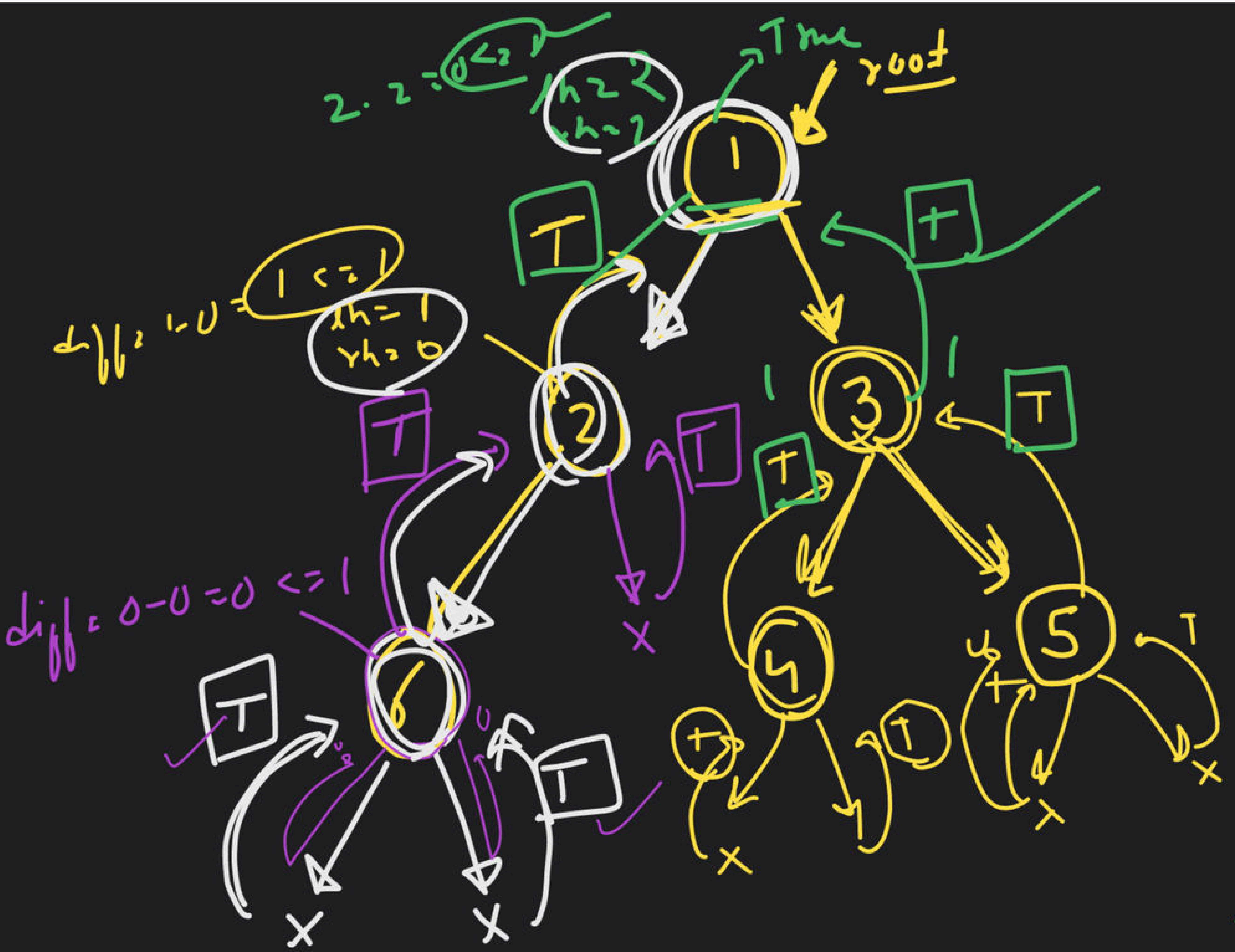
int bool

{ 9 , b }

height balanced or not

pair <int, bool>

p = make_pair(' , ');



3 cond

① Left SubTree Balanced

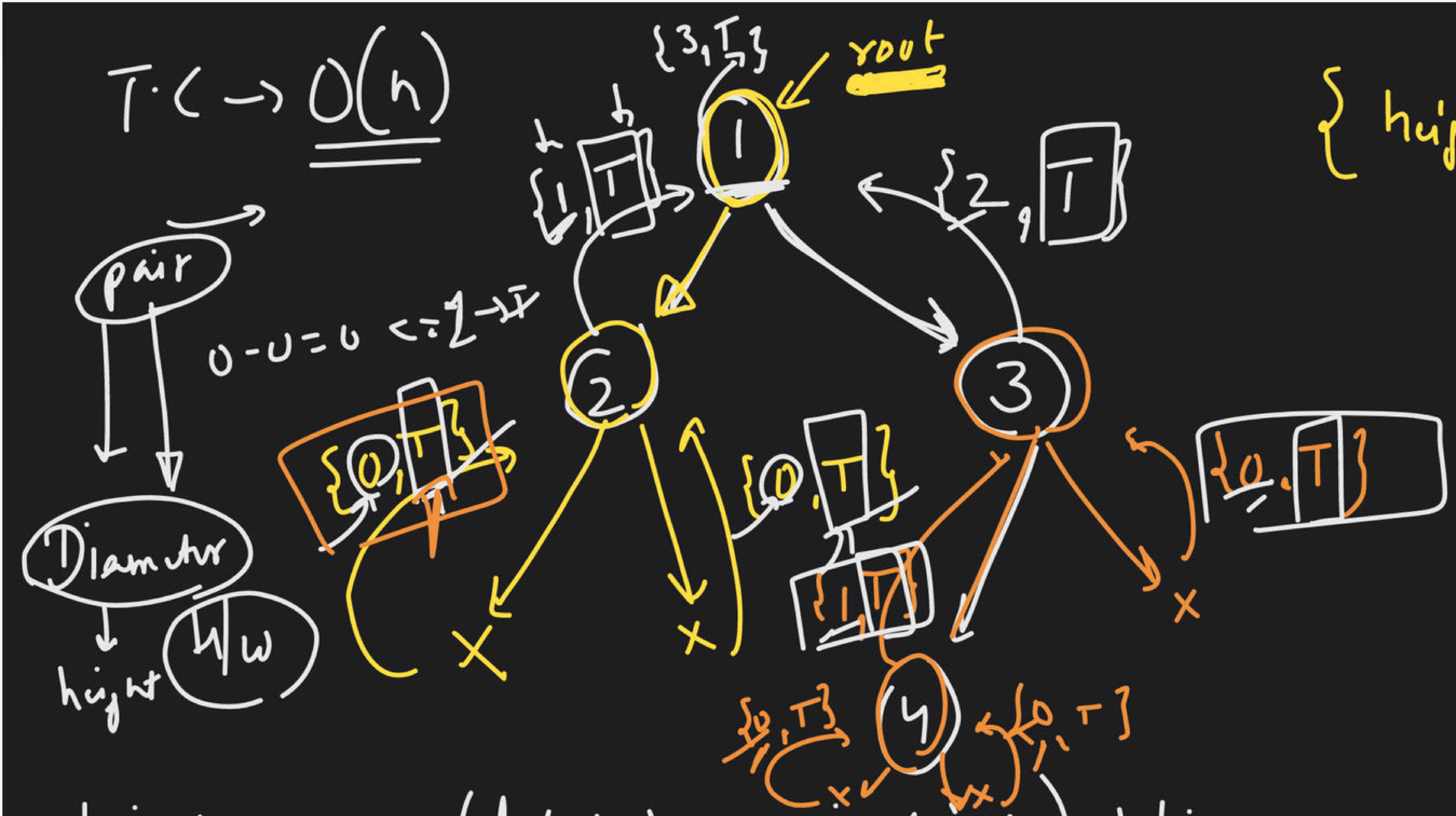
② Right Subtree Balanced

③ abs (lh - rh) <= 1

Balanced tree or not

T·C -> O(n)

{height, balance)



height = max (leftHeight, rightHeight) + 1;

balance -> LA.second && rightAns.second && abs (lh - rh) <= 1;

$$\boxed{lh - rh}$$

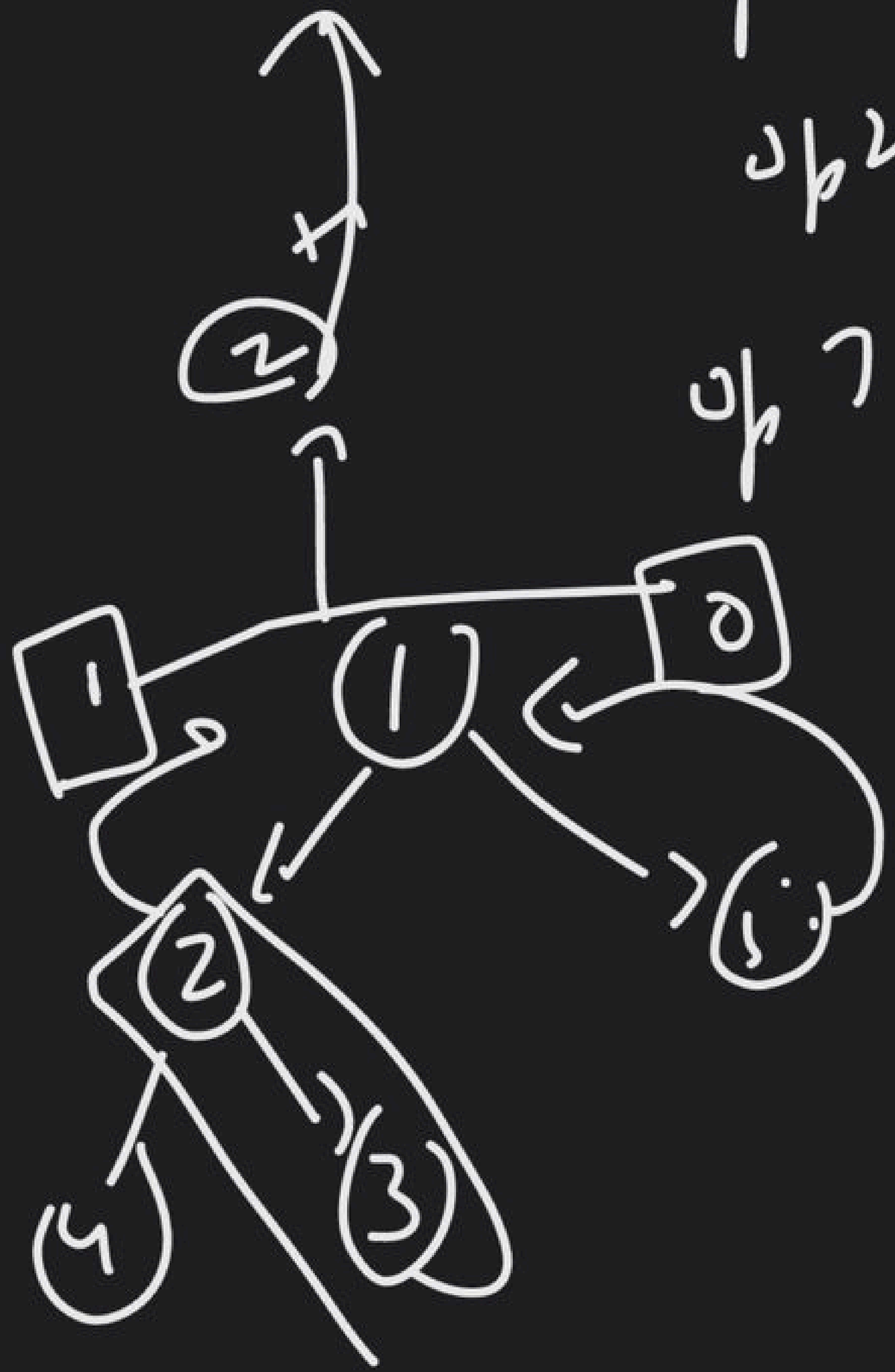
$$\underline{3} \qquad 4 \longrightarrow \textcircled{1}$$

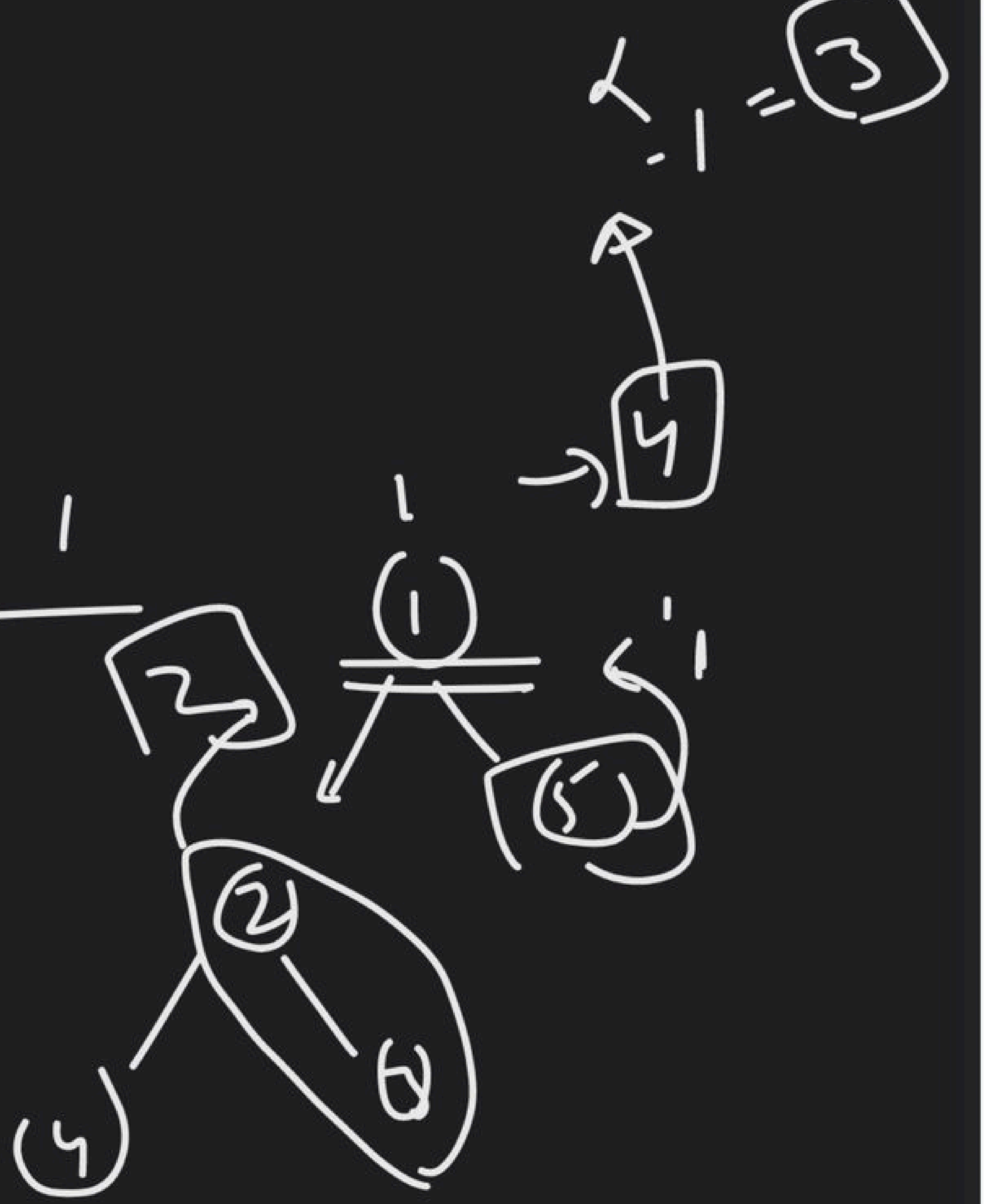
$$\underline{\underline{2}} \qquad 3 \longrightarrow \textcircled{1}$$

op1 = left diam

op2 = right dia

op3 = $\boxed{lh} + \boxed{rh} + 1$



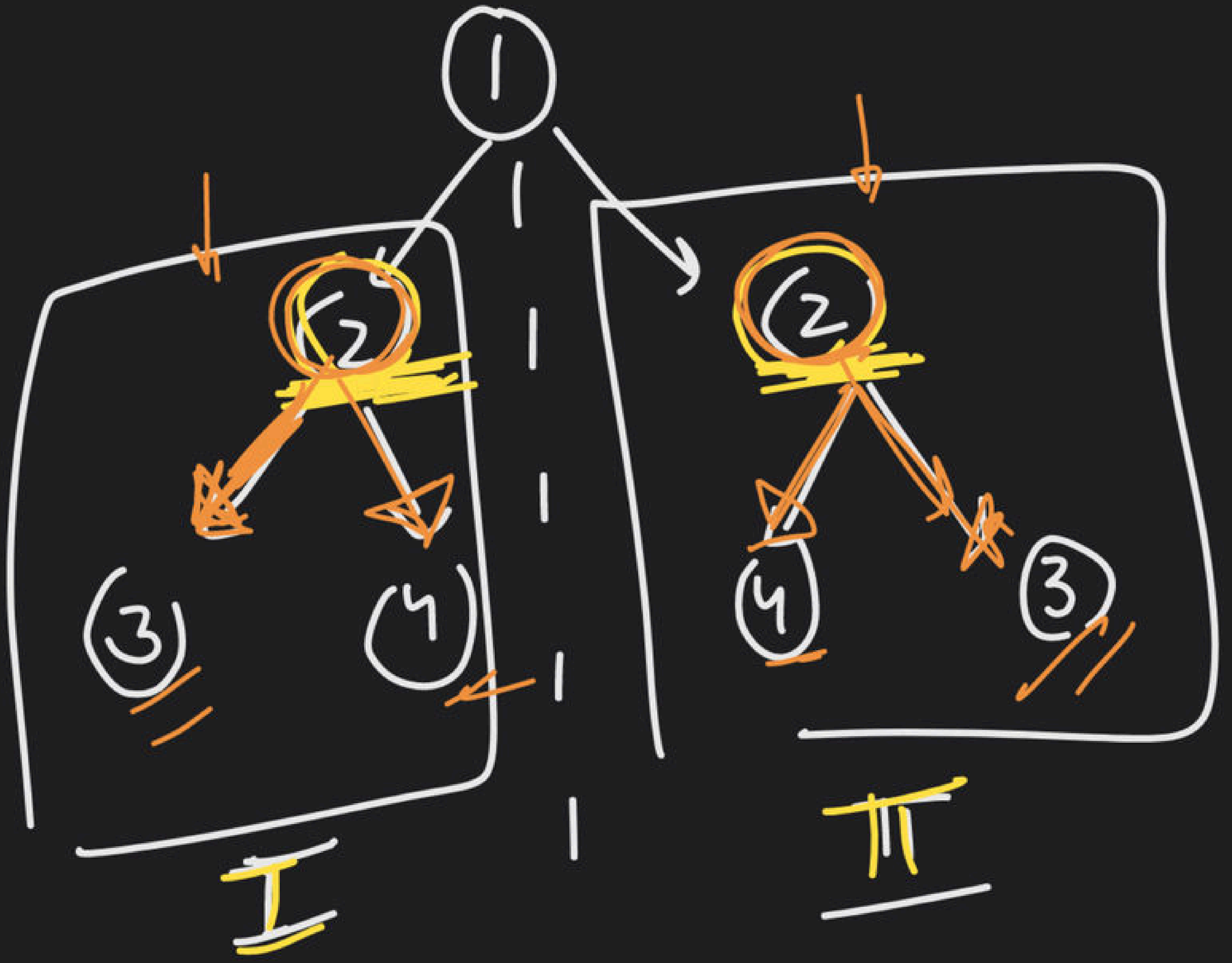


→ Lateral Inversion

Dora the Explorer

Recursion

a



!NULL && = NULL

false

=N && !N

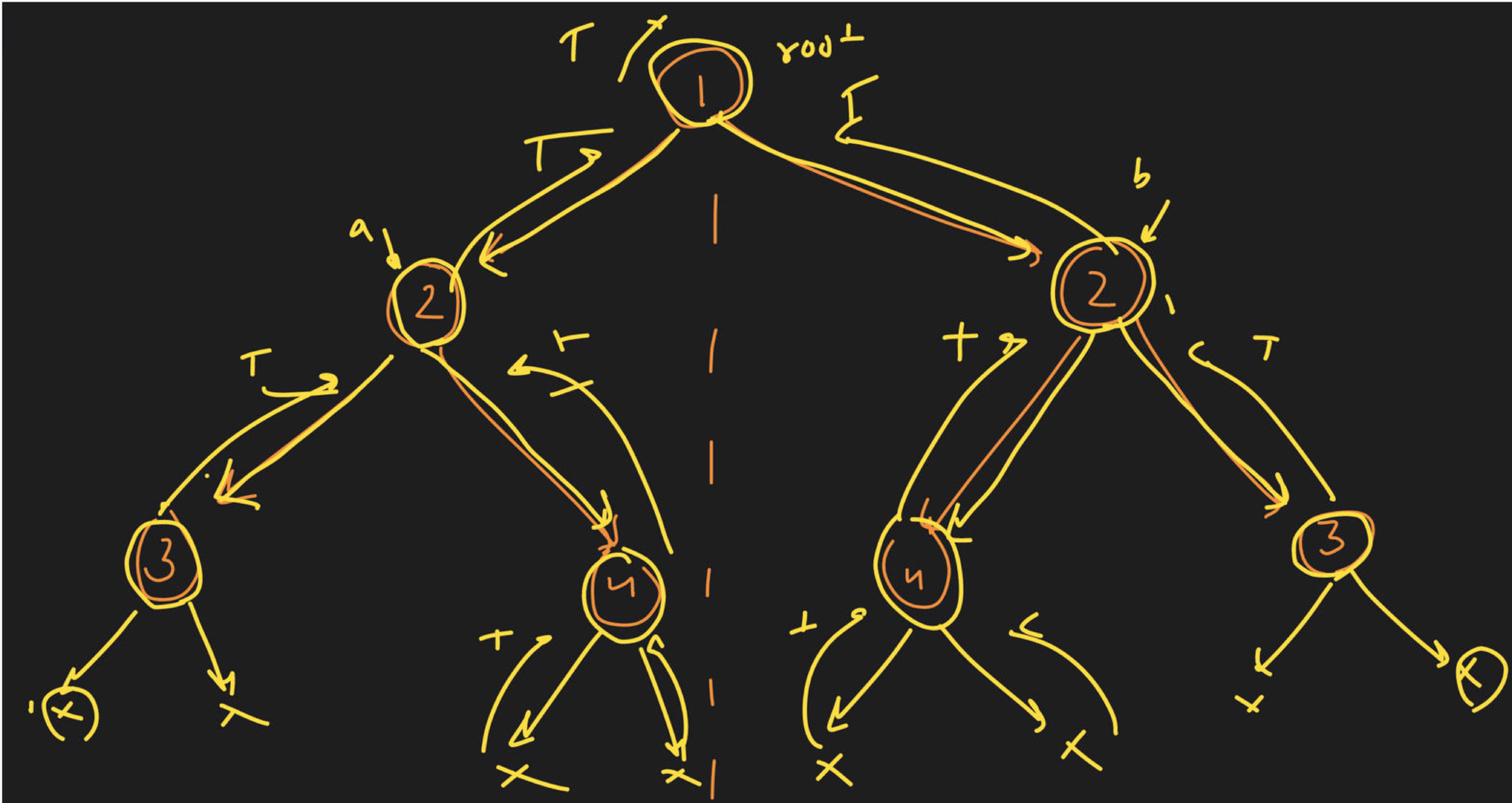
F

=N && =N

T

→ val != val

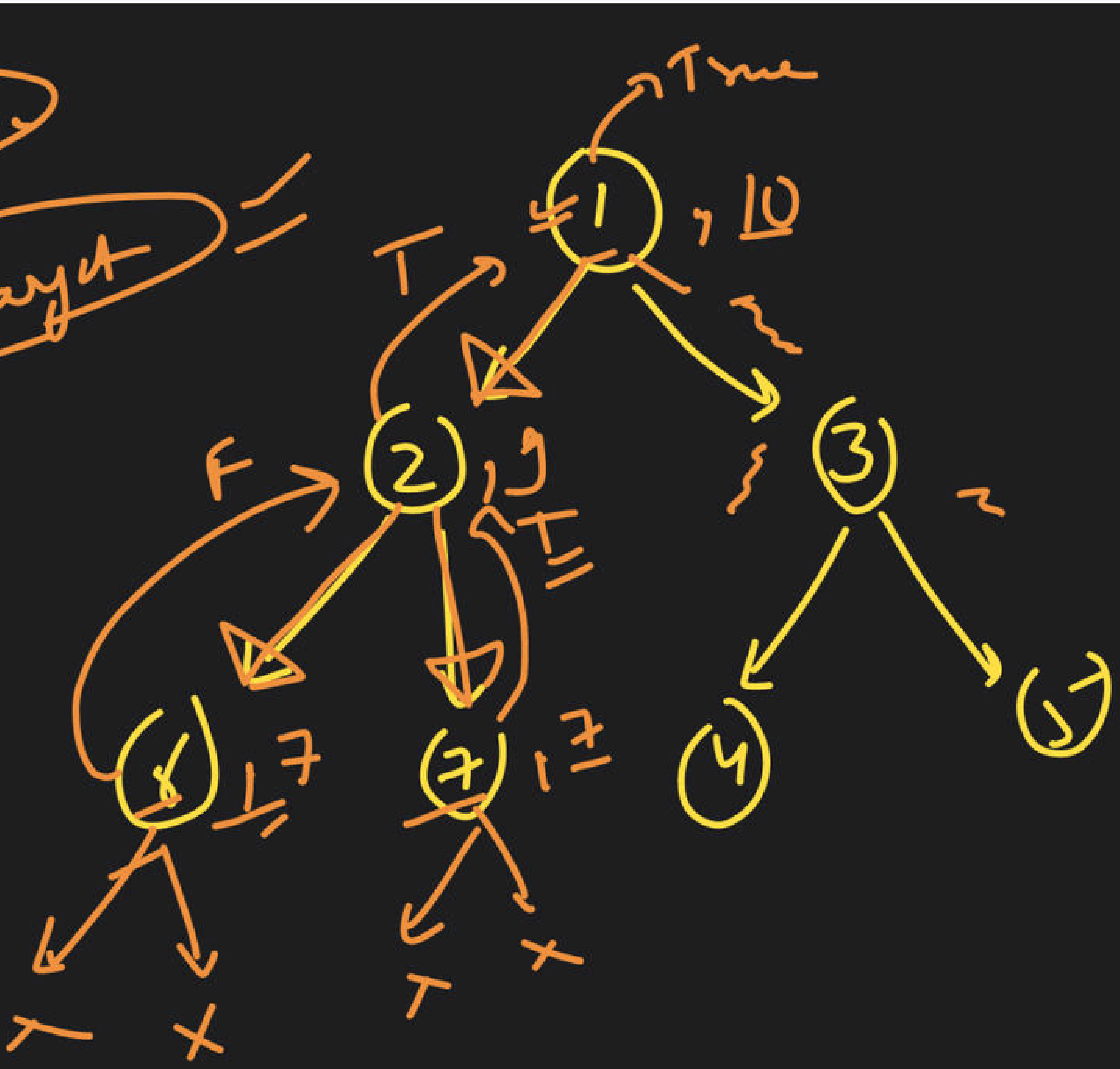
False


T
1
root
T
a
2
b
2
T
T
3
4
4
3
T
T

Leaf node

Val == Target

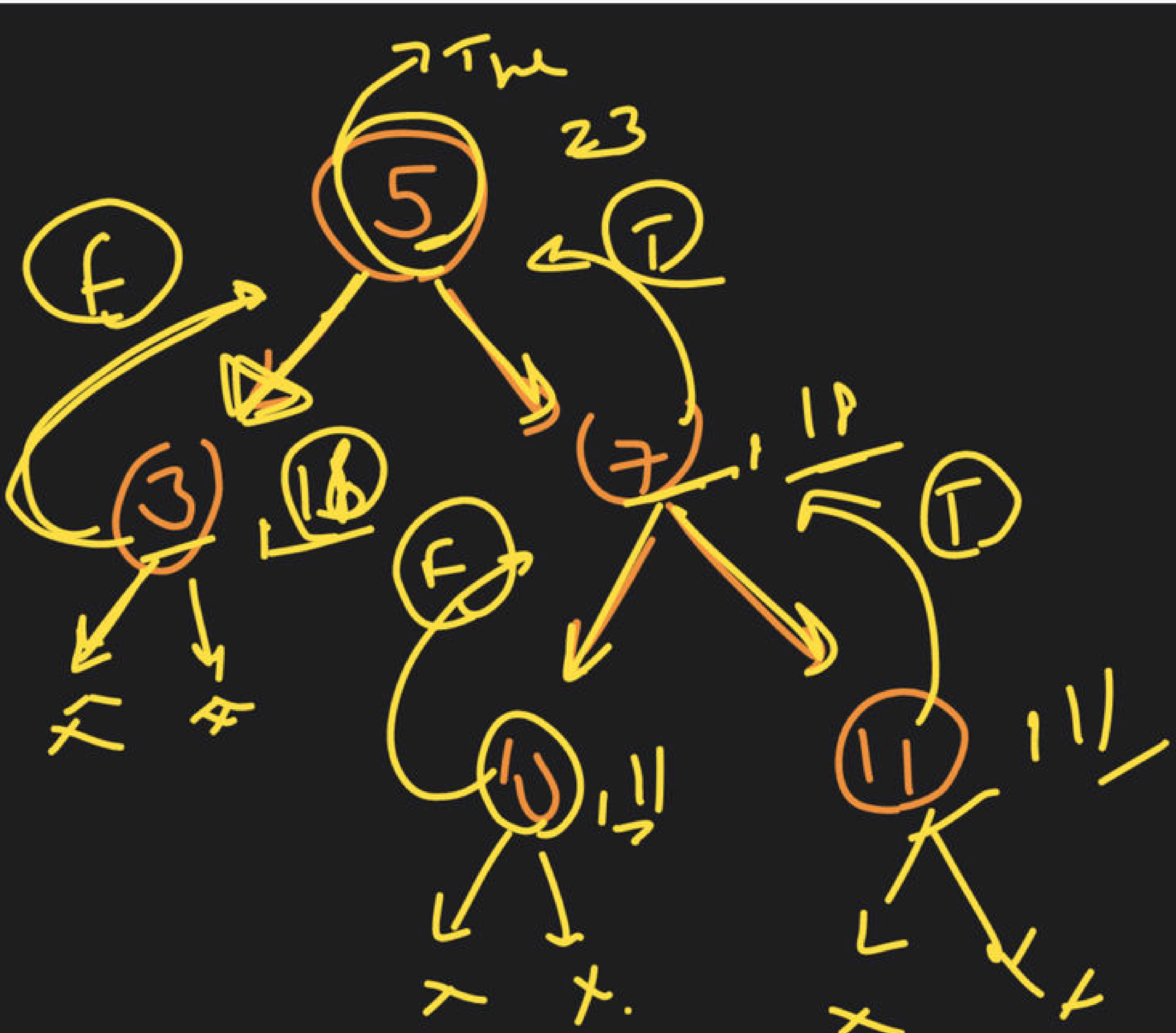
fch



T = 9

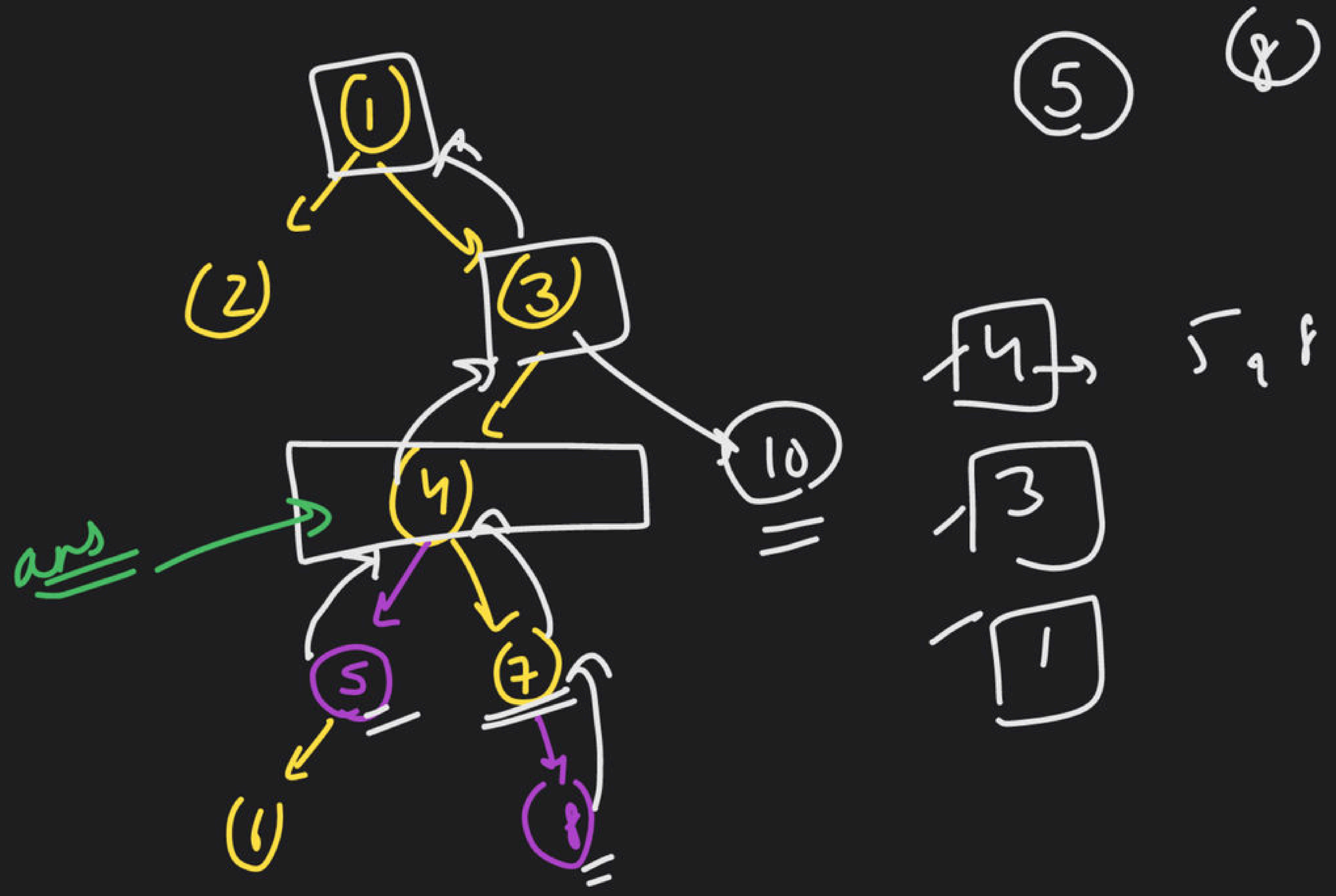
L.N

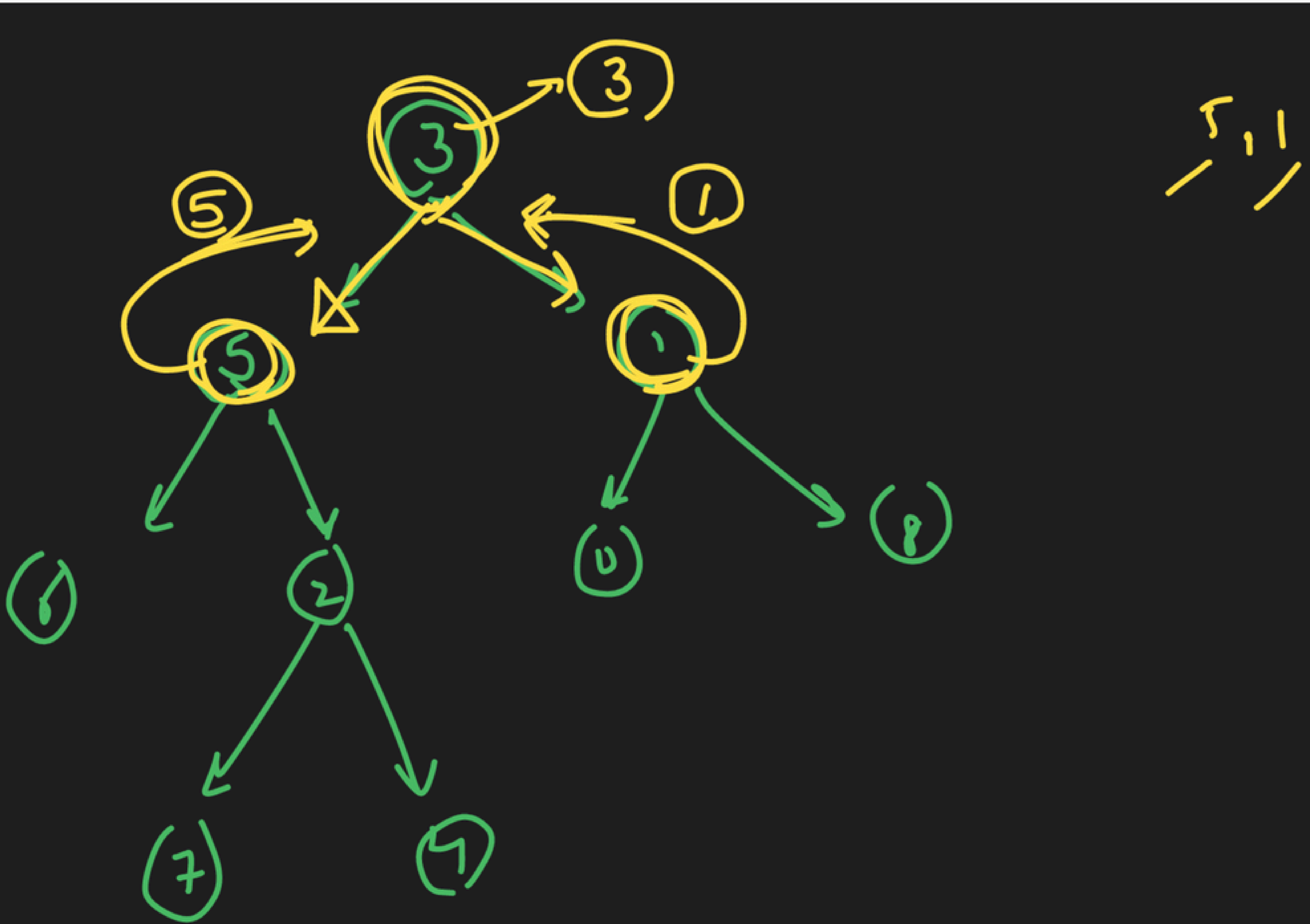
↳ unp

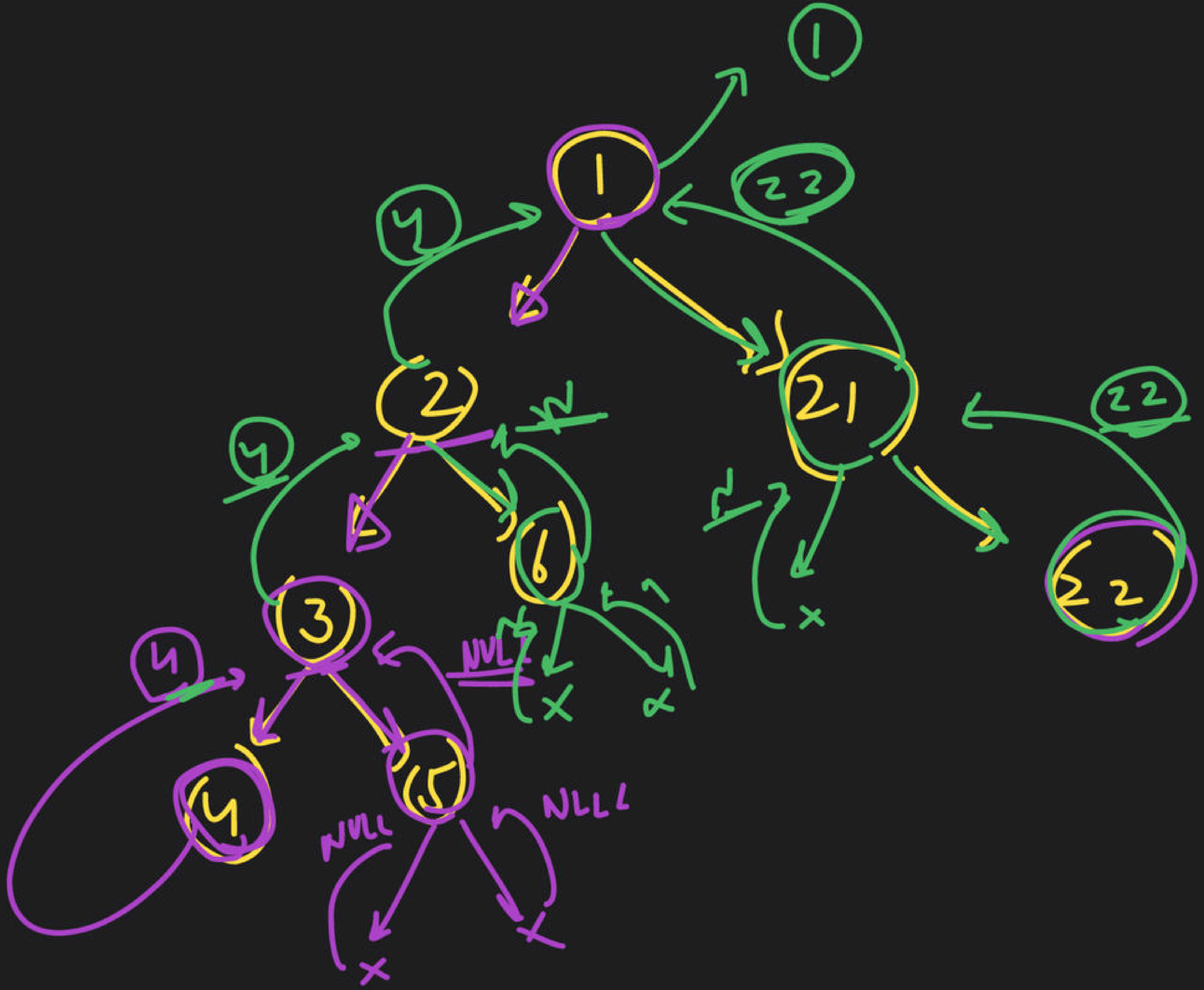


T=23

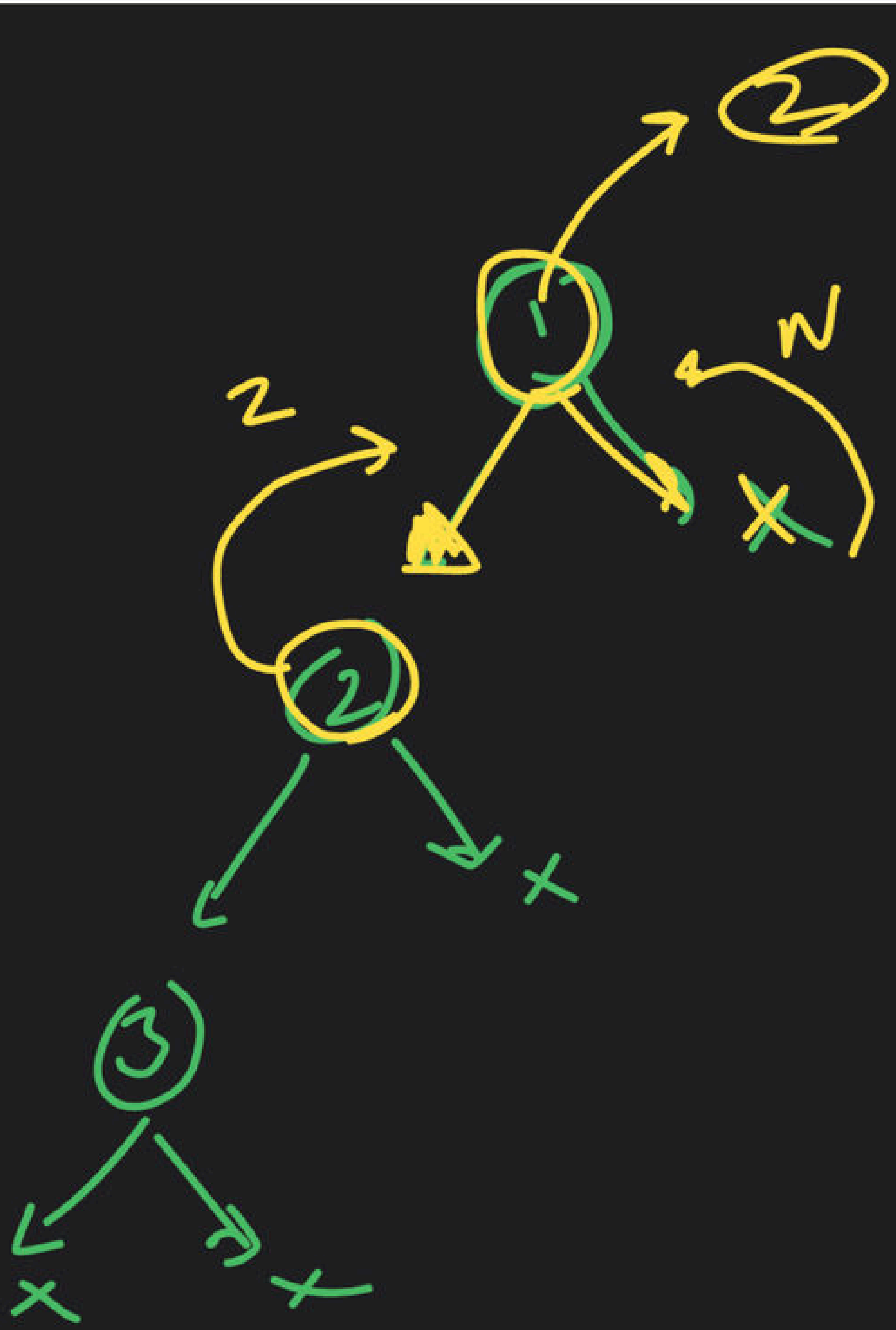
→ Lowest Common Ancestor :-

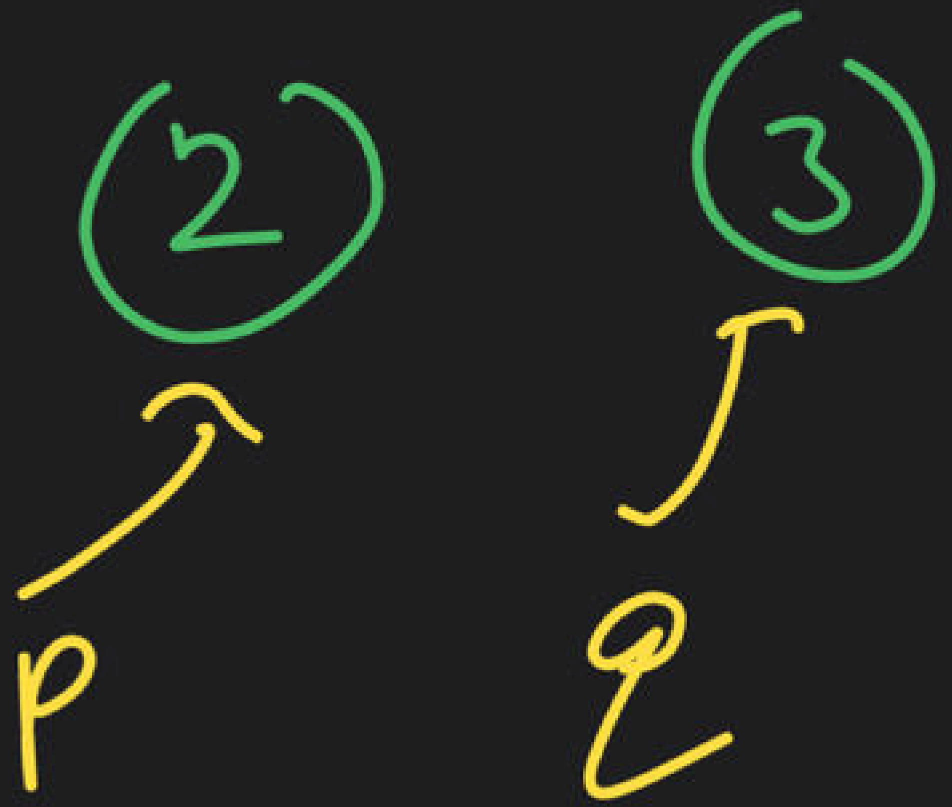


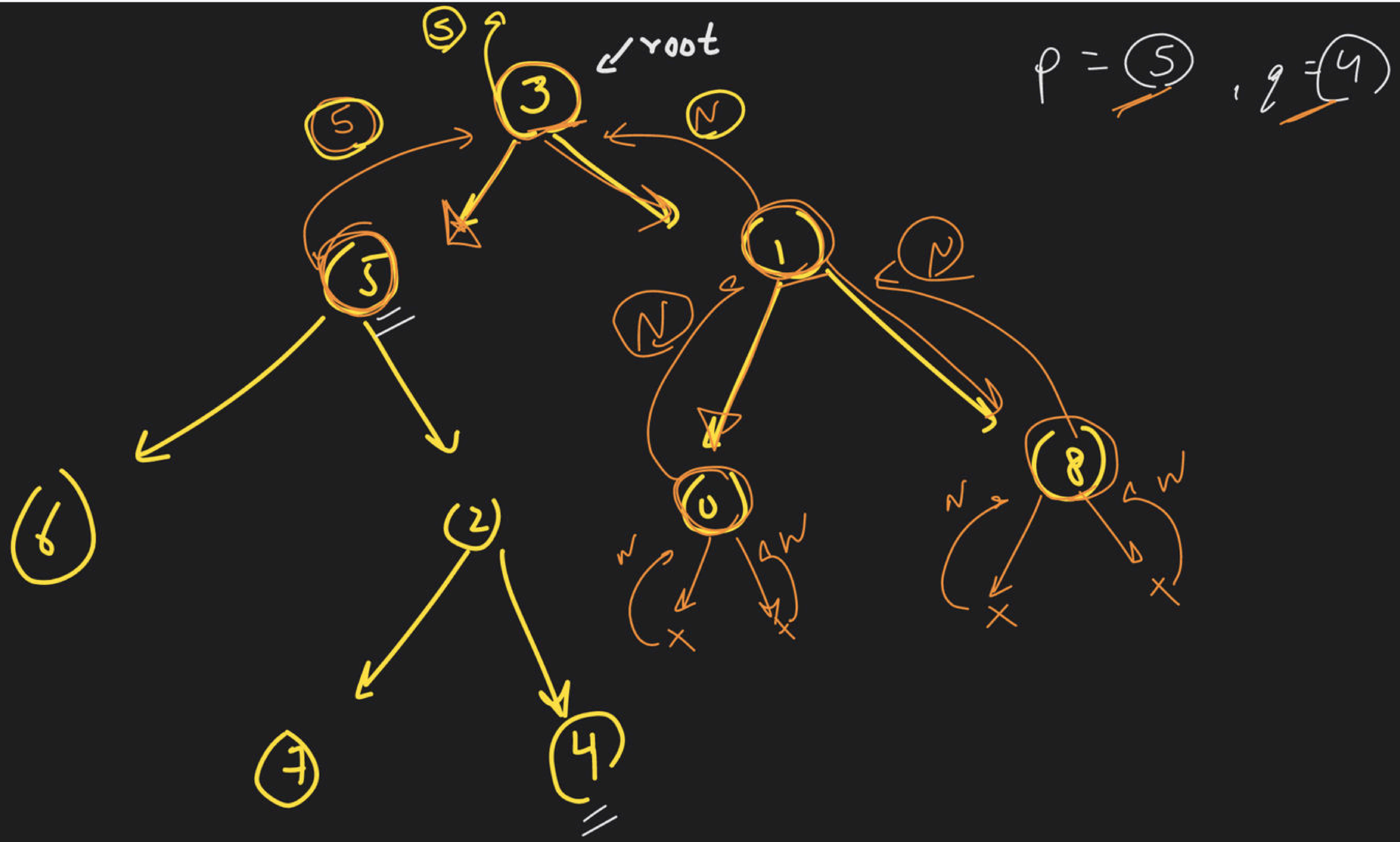

3
3
5
5
1
1
6
2
0
8
7
7
5, 1


1
1
22
4
2
21
4
22
6
3
4
NULL
4
5
NULL
NULL
x
x
x
x
x

4
22

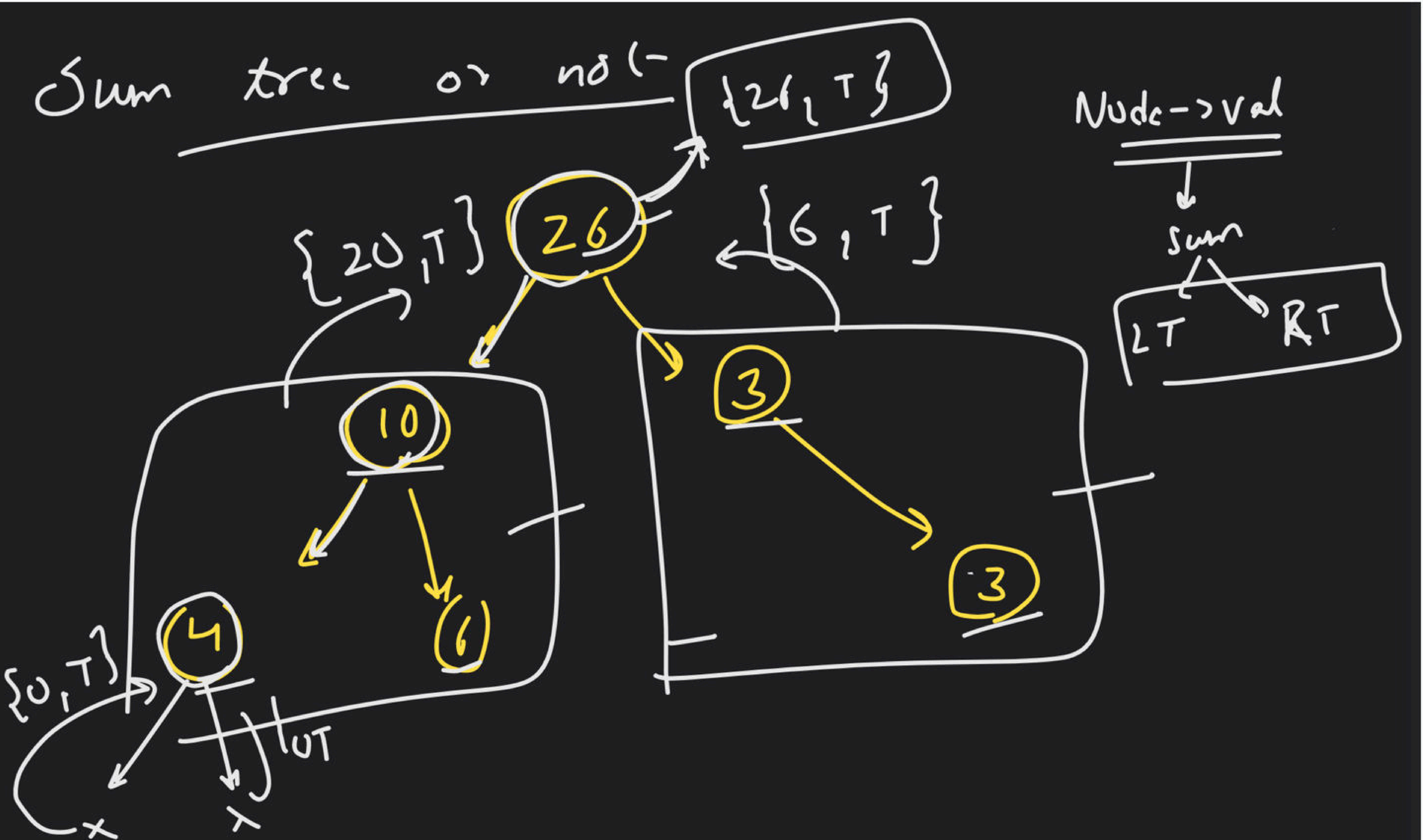

2
1
2
N
2
3


2
3
p
q

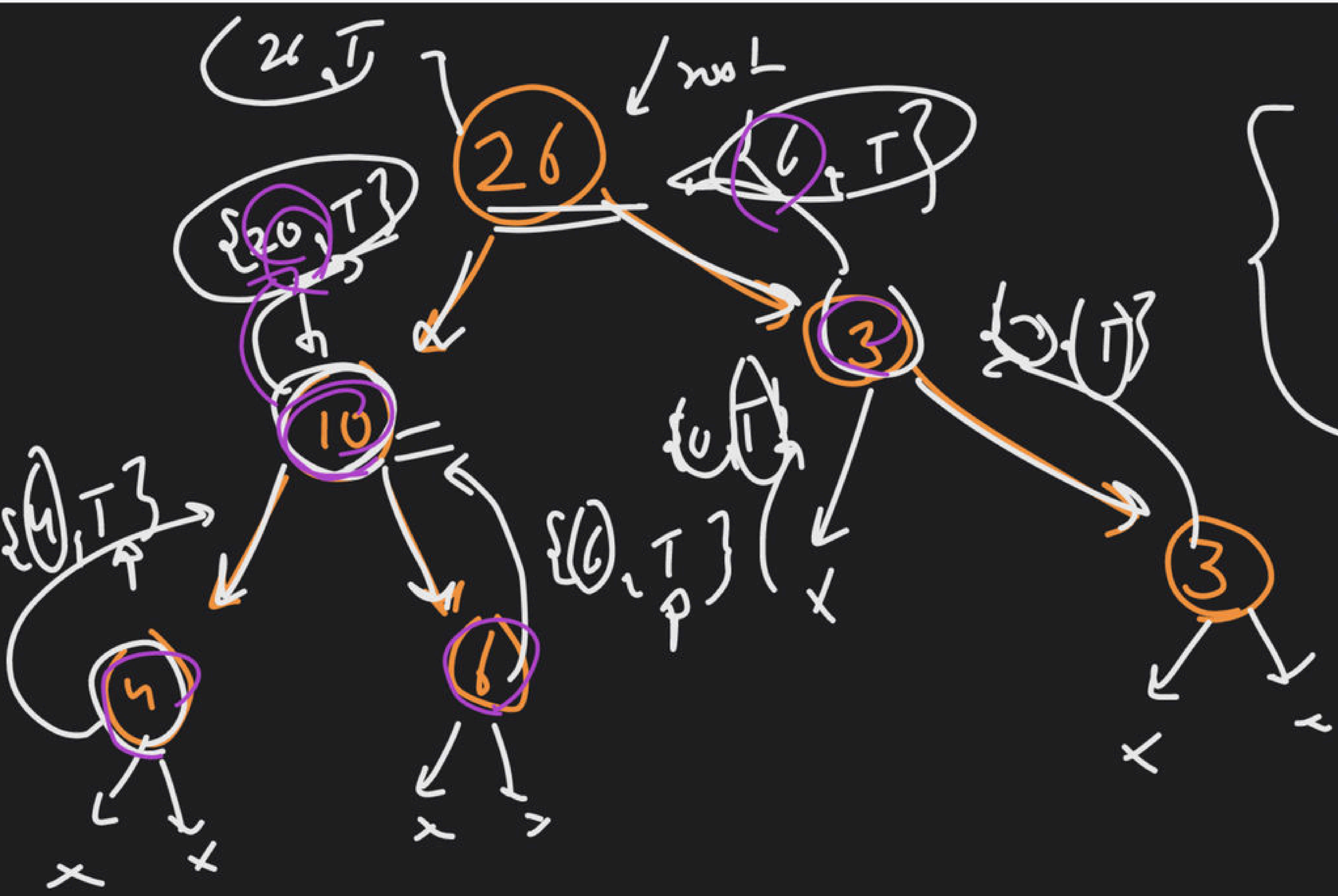


p = 5 , q = 4
root
5
3
5
N
5
1
N
N
6
2
0
8
7
4
N
W
N
W
×
×
×
×

→ Sum tree or not



L·F
↓
{ val , T }



Left sum Tree

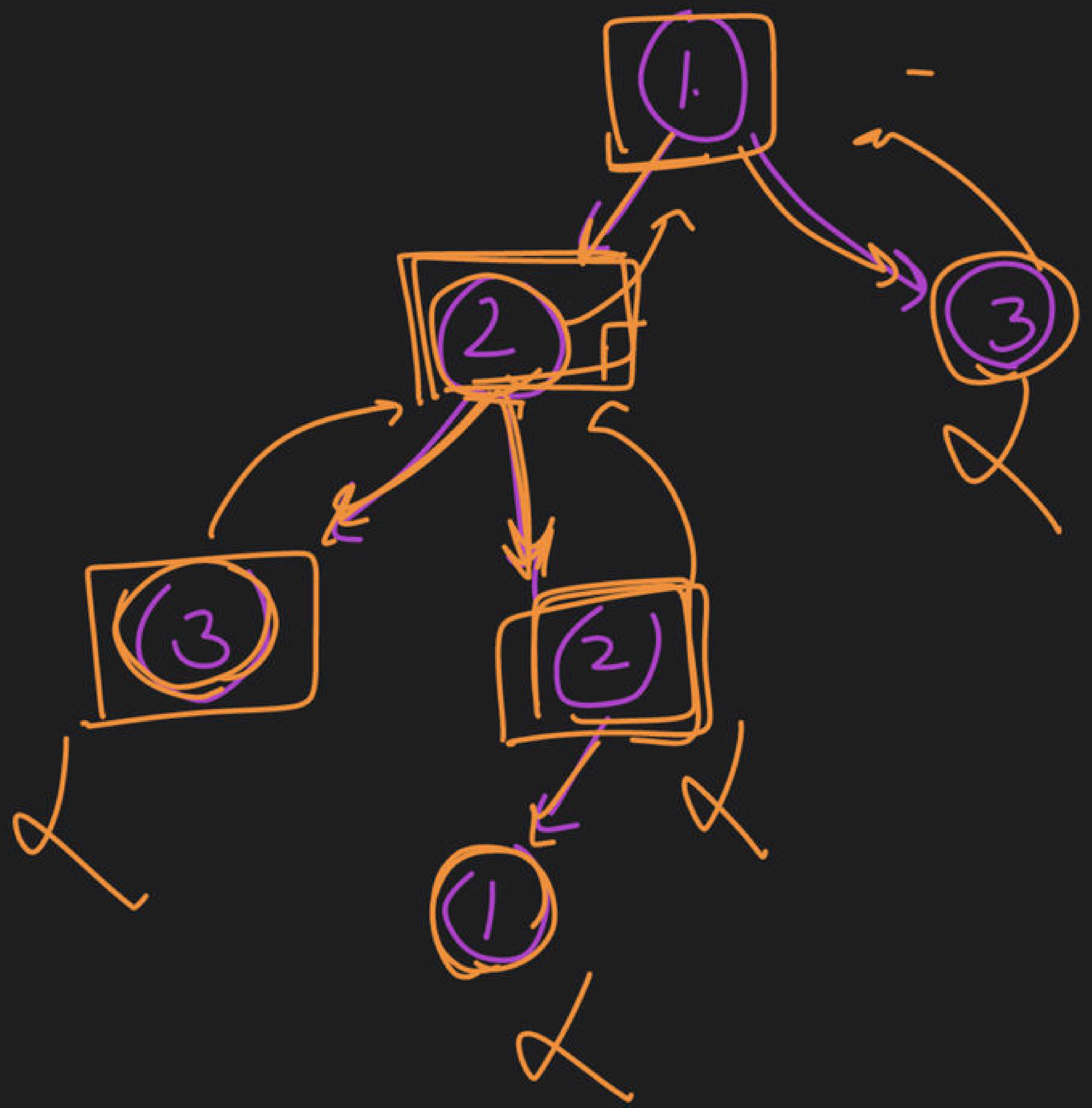
RightSumTree

val = left sum + right_

```
pair<int, bool> solve (Node* root)                {0, T};
{
    //B.C  if (root == NULL)
                 return make_pair(0, True);

    //B.C  if (root->left == NULL && root->right == NULL)
                          return make_pair(root->val, True);

    pair<int, bool> left = solve(root->left);
     pair<int, bool> right = solve (root->right);
  if( left.second && right.second && (root->val == left.first
                                                       +
                                                    right.first))
              return make_pair(2 * root->val , True);
 } else return make_pair( ______ , false);
                                         root->val + left.first + right
                                                                  .first
```



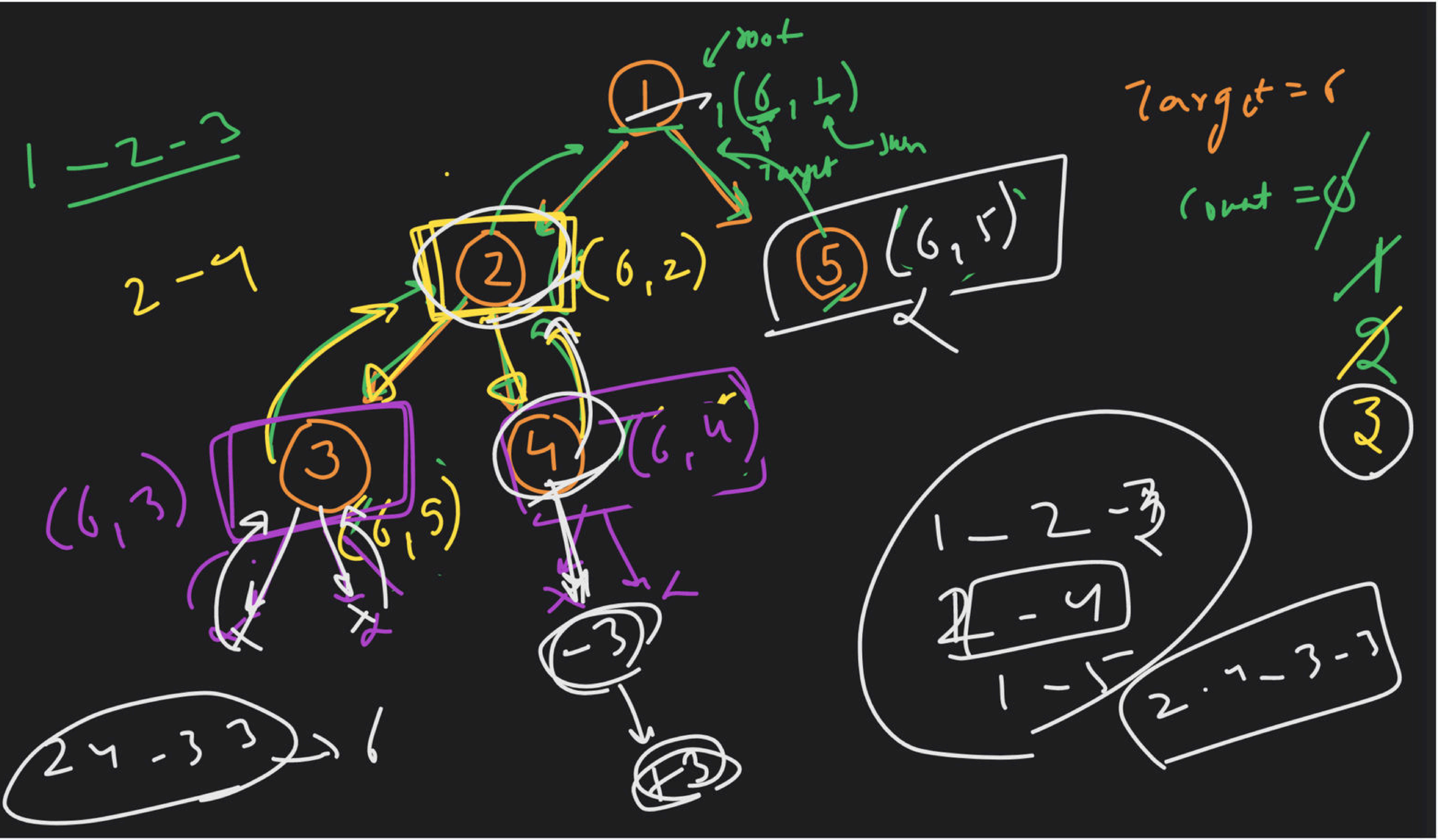
5

1 – 2 – 2

2 – 3

2 – 2 – 1

T.C → $O(n^2)$


root
(6, 1)
sum
target
Target = 6
count = 0
1
2
3
1 - 2 - 3
2 - 4
1
2
5
3
4
(6, 2)
(6, 5)
(6, 3)
(6, 4)
(6, 5)
2
-3
3
2 4 - 3 3 → 6
1 - 2 - 3
2 - 4
1 - 5
2 - 4 - 3 - 3

→ K-Sum path

H/W

→ Kth ancestor

→ Traversals:- Zig-Zag / Diagonal / Vertical

top / left / right / Bottom Views

→ Lowest Bloodline

→ Max Sum of non-adjacent nodes

→ Morris traversal → Burn the tree

→ Padhai start ho chuki hai?

→ On-campus placement ?