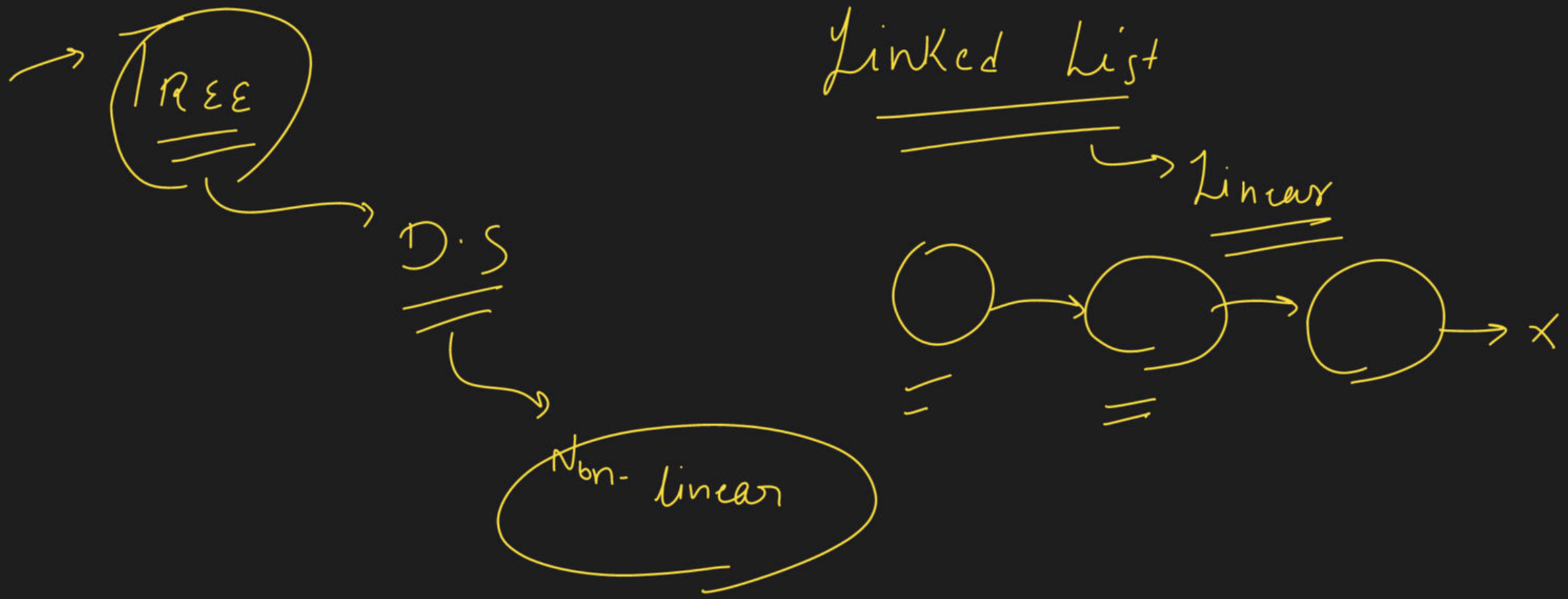
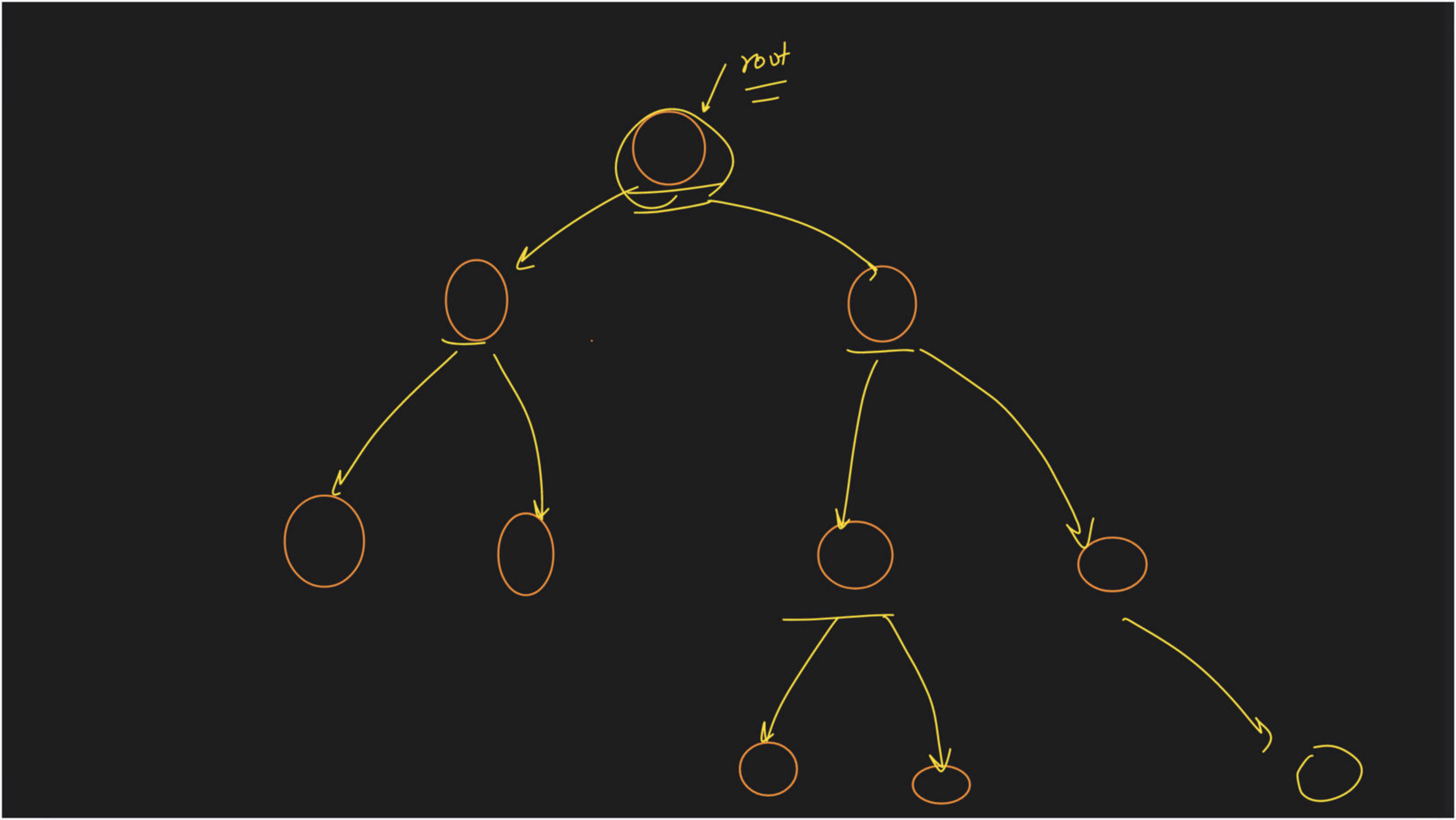
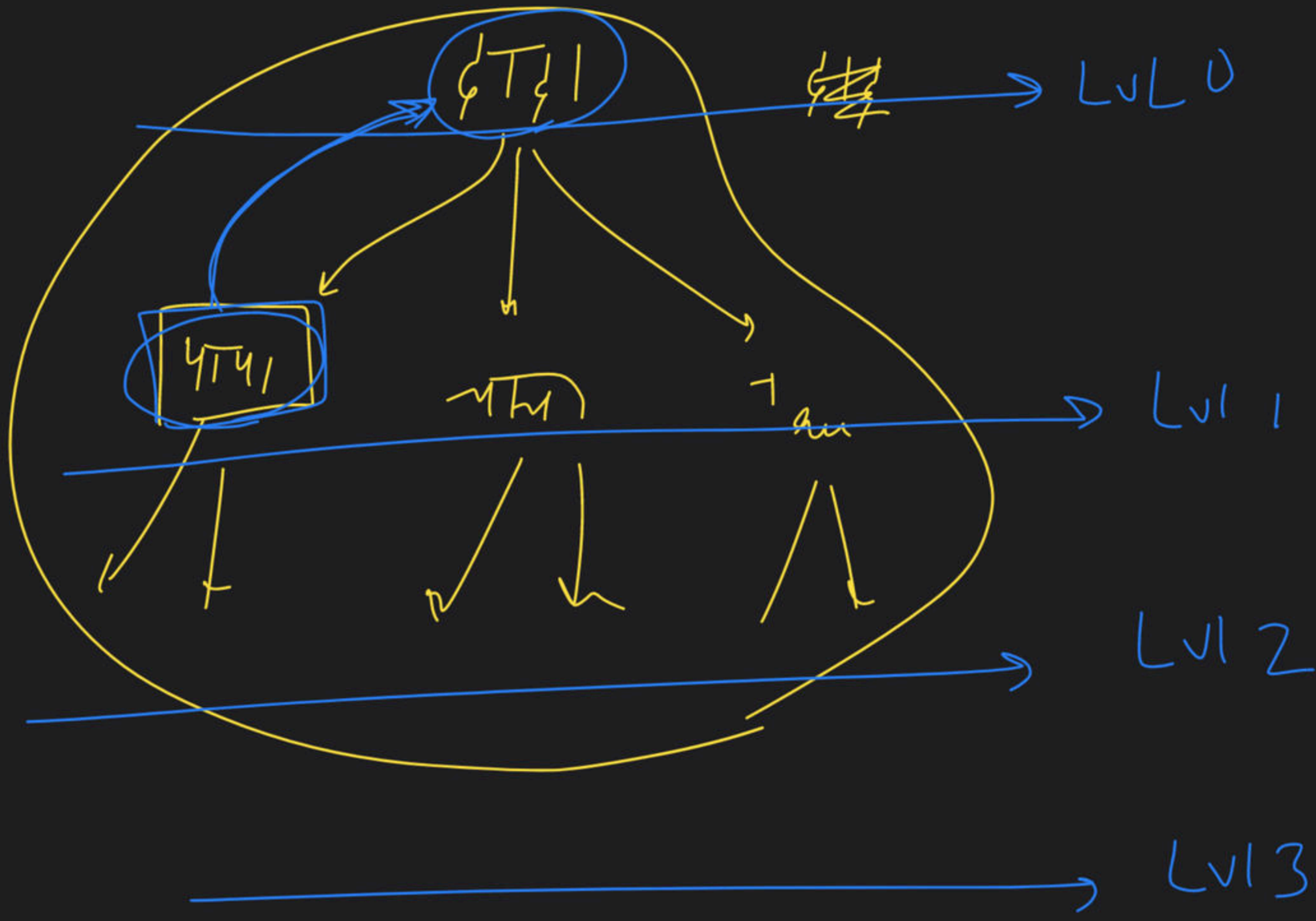
# Trees Class - 1

Special class

**Love Babbar** • May 6, 2023


TREE
D.S
Non- linear
Linked List
Linear
X

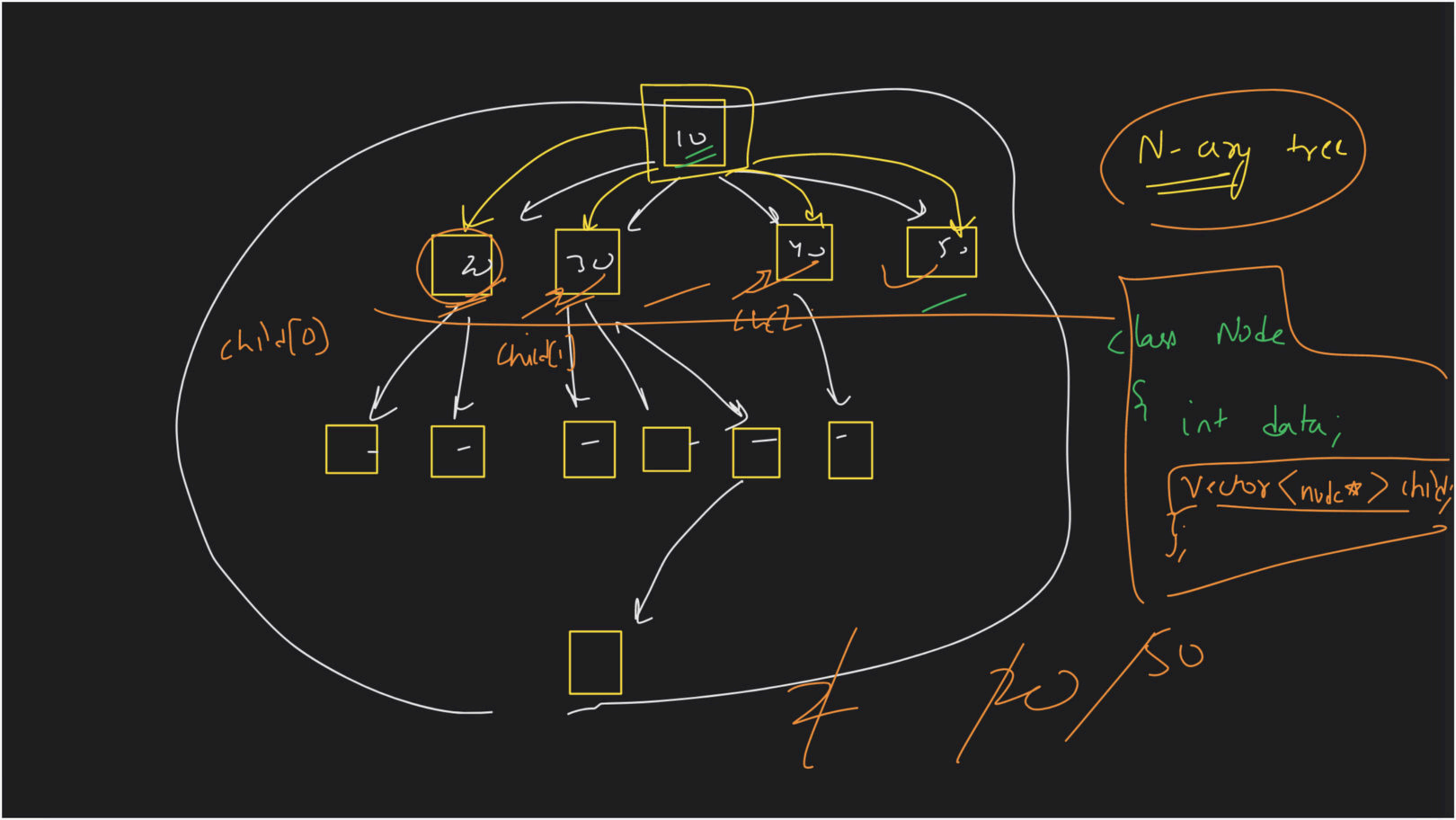

root

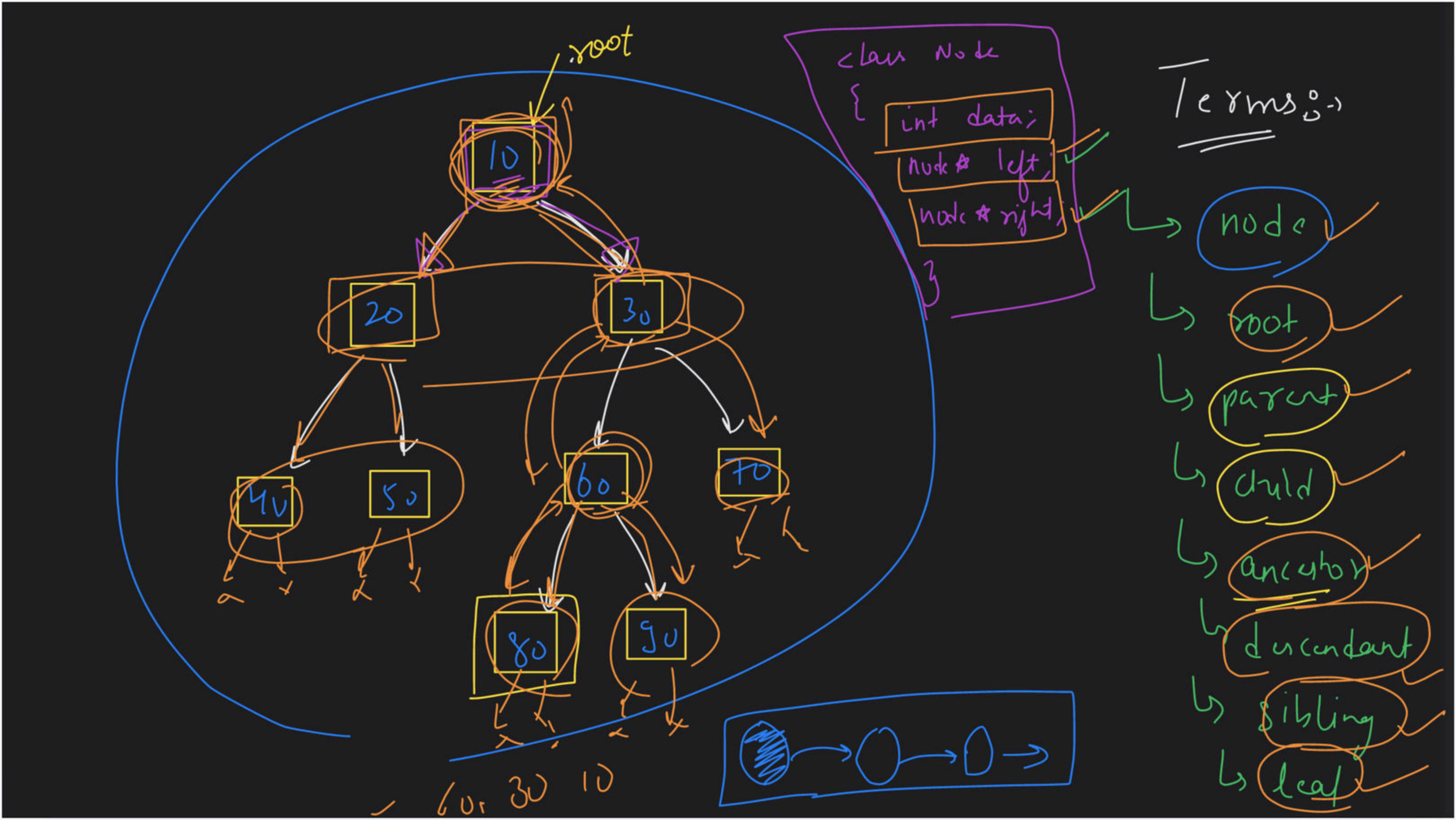

LvL 0
Lvl 1
Lvl 2
Lvl 3

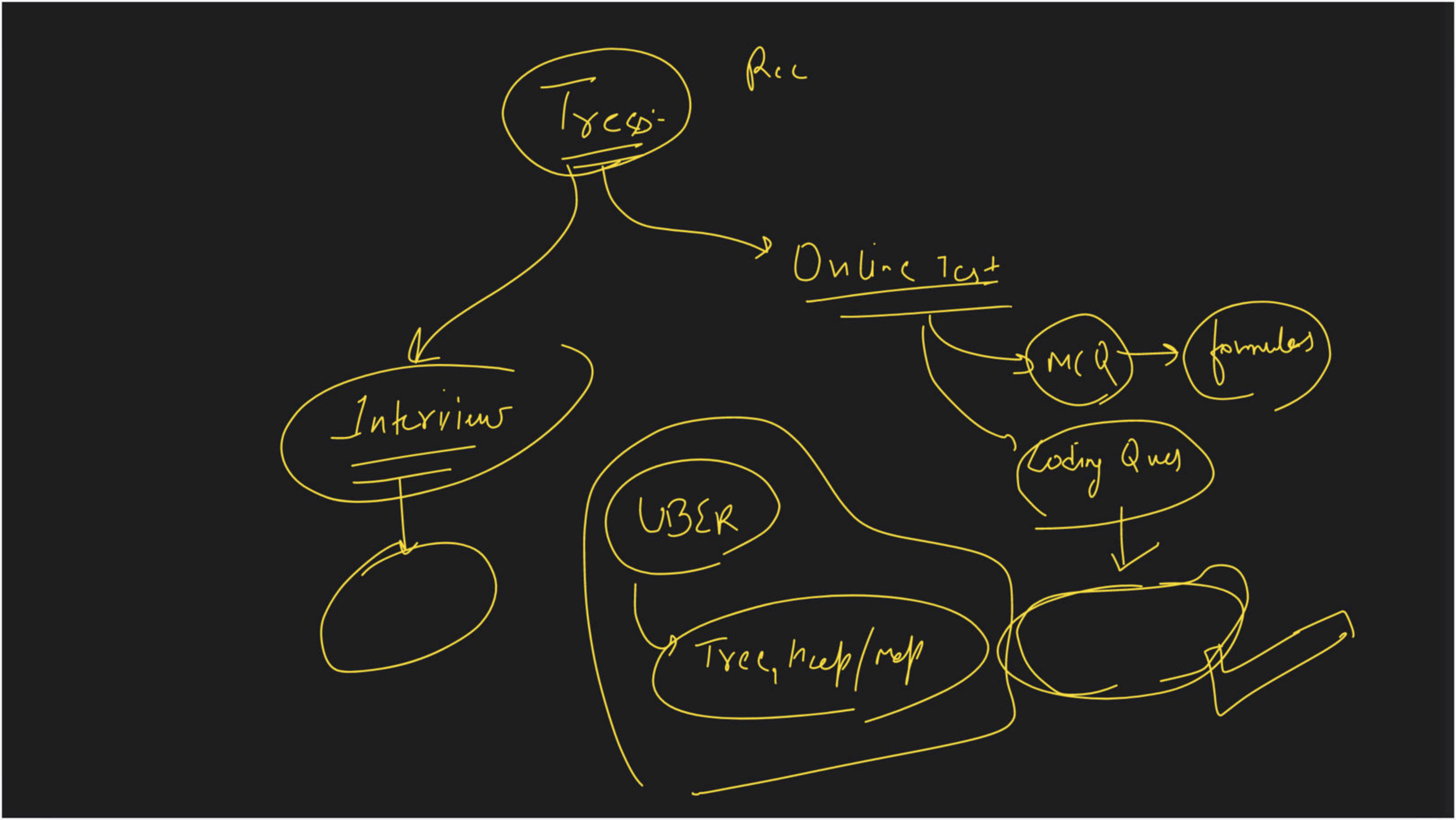
→ Tree → Non-linear DS


Child → 0 / 1 / 2

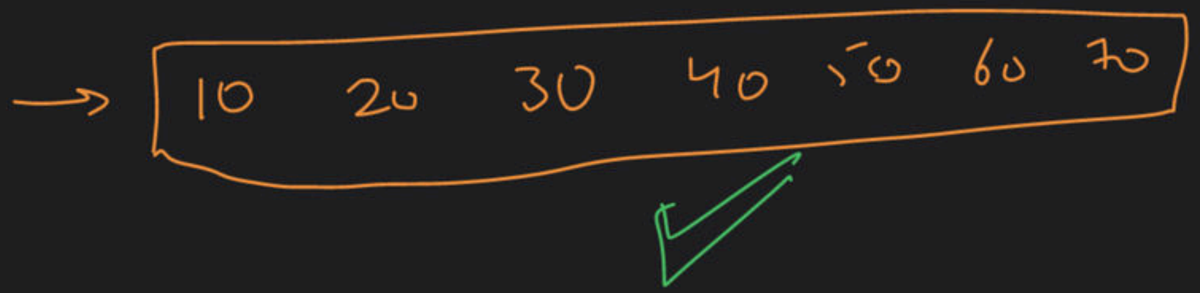
Binary Tree

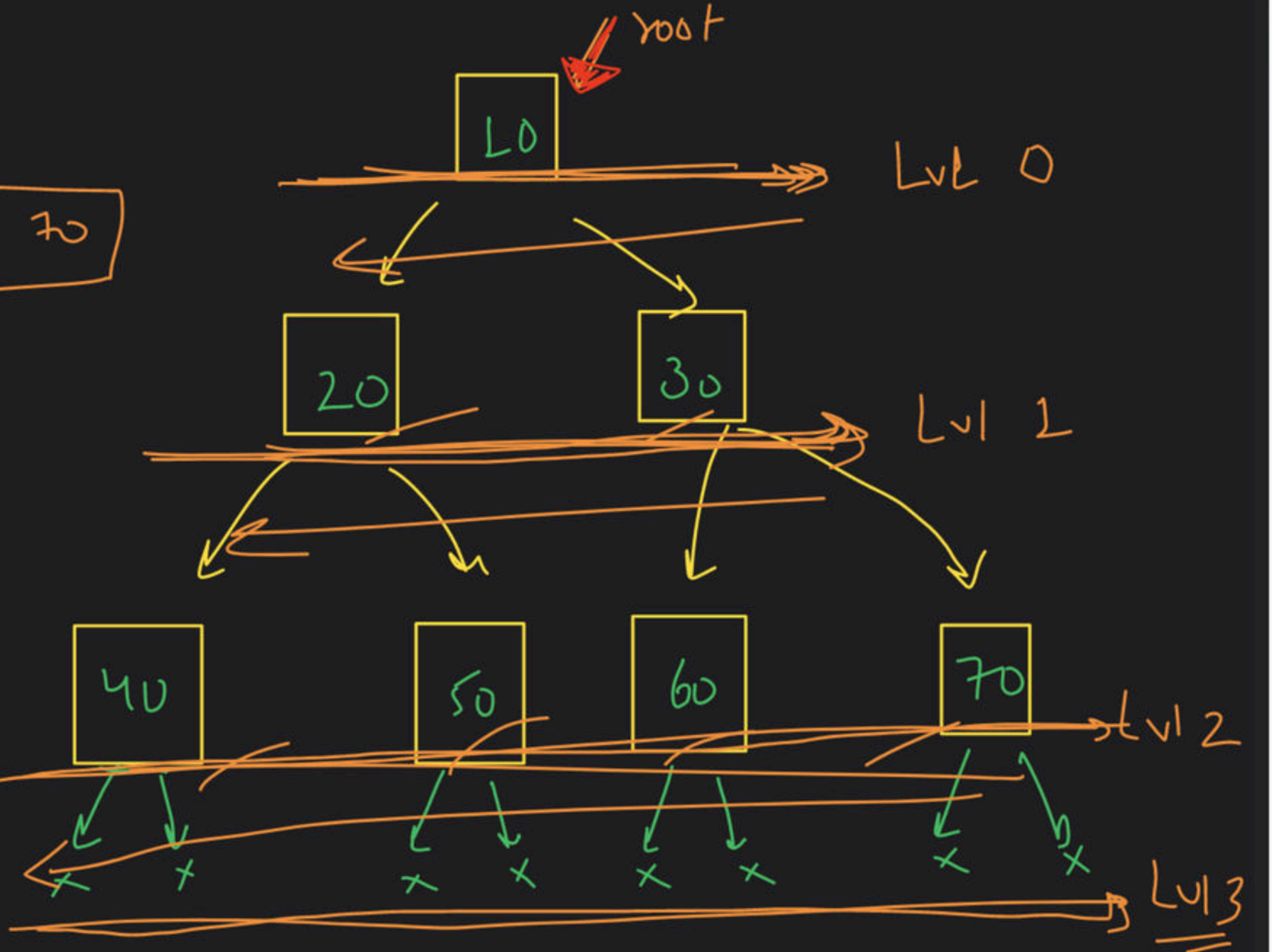

N-ary tree
10
20
30
40
50
child[0]
child[1]
child[2]
class Node
{
int data;
vector<node*> child;
};


root
10
20
30
40
50
60
70
80
90
class Node
{
int data;
node* left;
node* right;
}
Terms :-
node
root
parent
child
ancestor
descendant
sibling
leaf


Rec
Tycs-
Online Test
MCQ
formulas
Interviews
Coding Ques
UBER
Tree, Heap/Map

# Level Order traversal



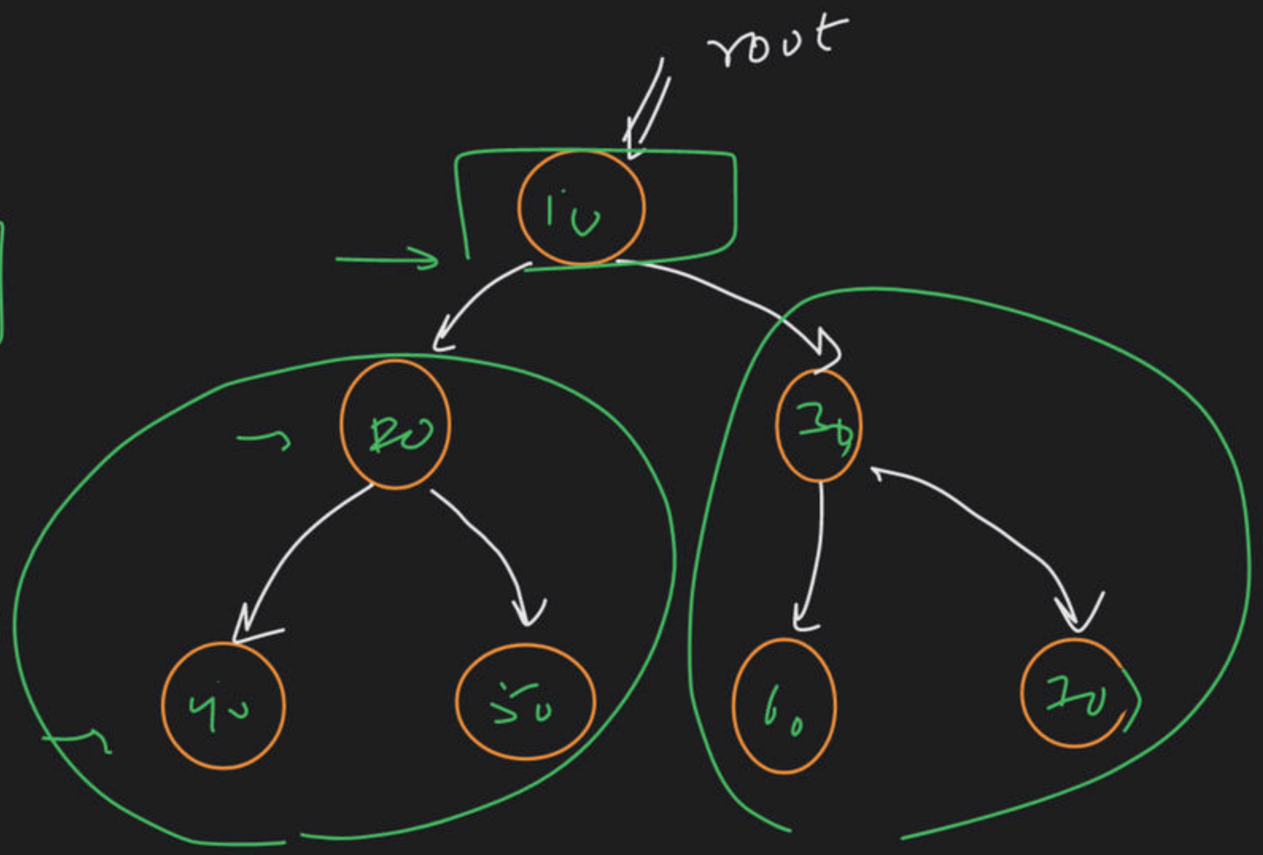

Level Order traversal

Inorder / PreOrder / PostOrder traversal

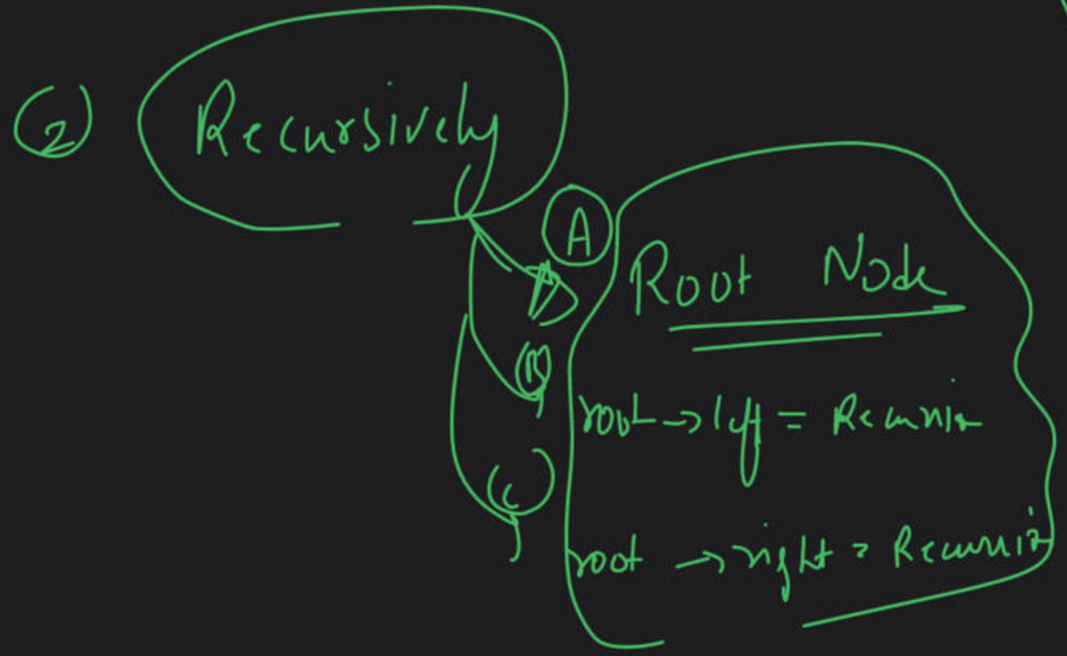
Morris traversal

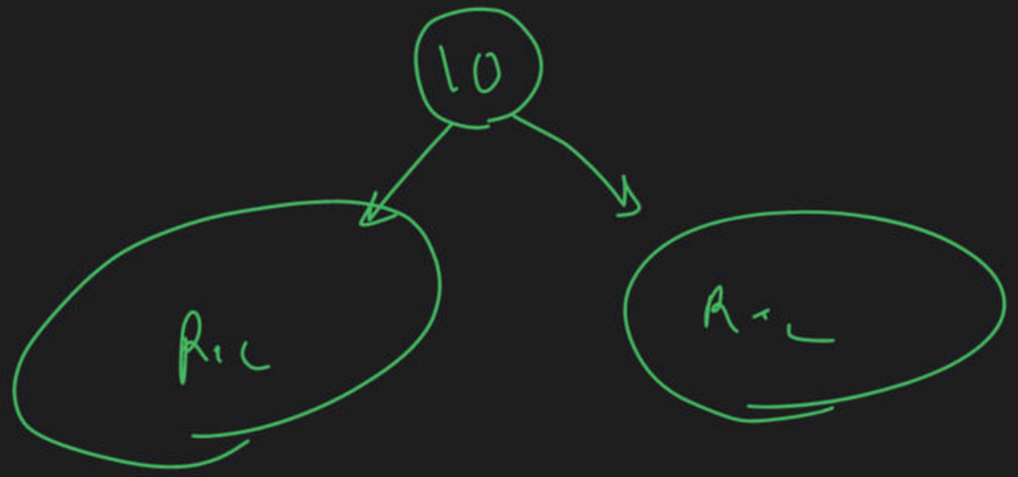
# Build Tree

① Create Using L·O·T




② Recursively

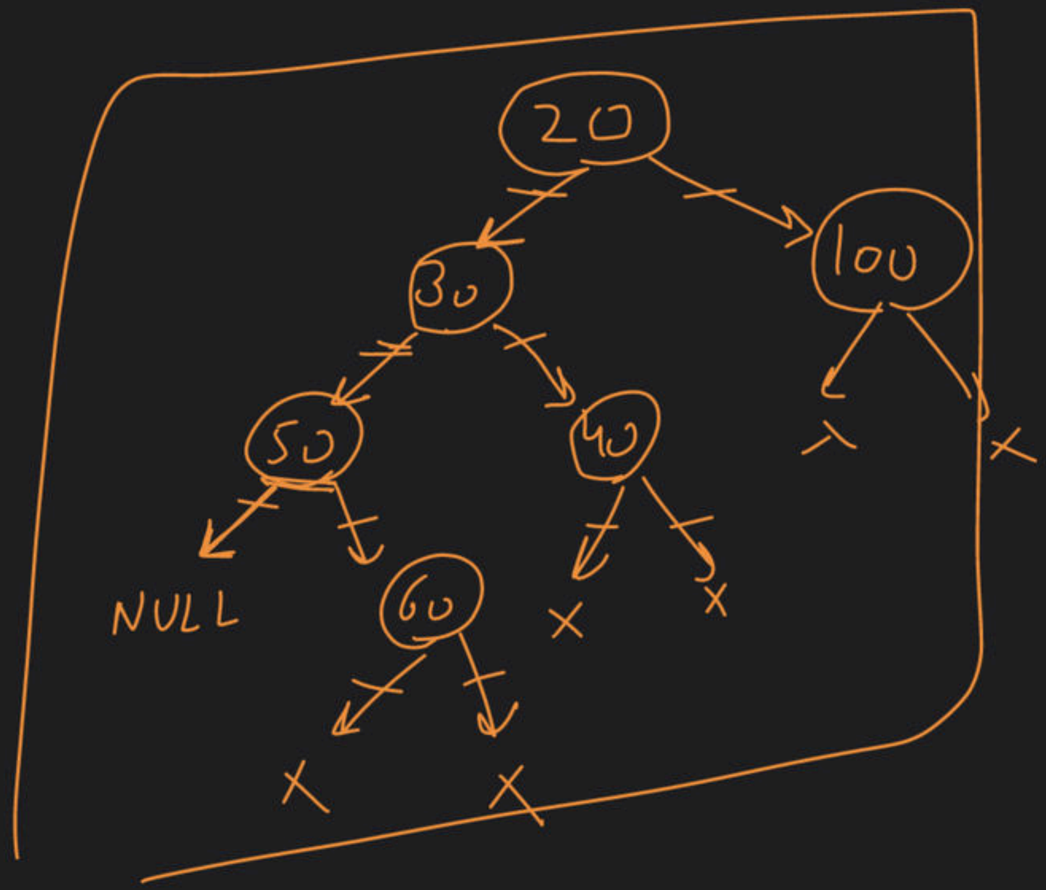


(A) Create Root Node

(B) root -> left = Recursion ()

(C) root -> right = Recursion ()



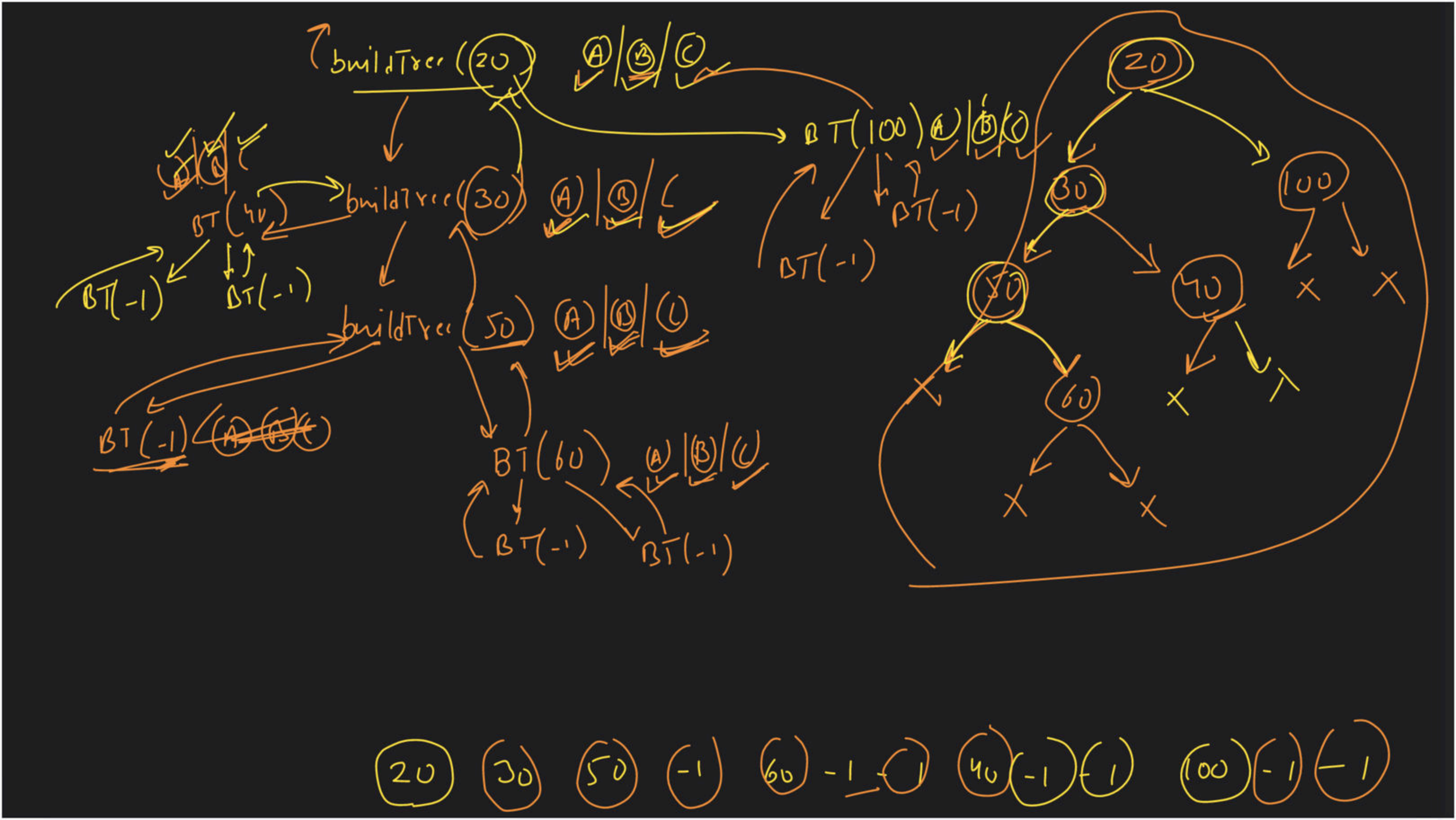

assum

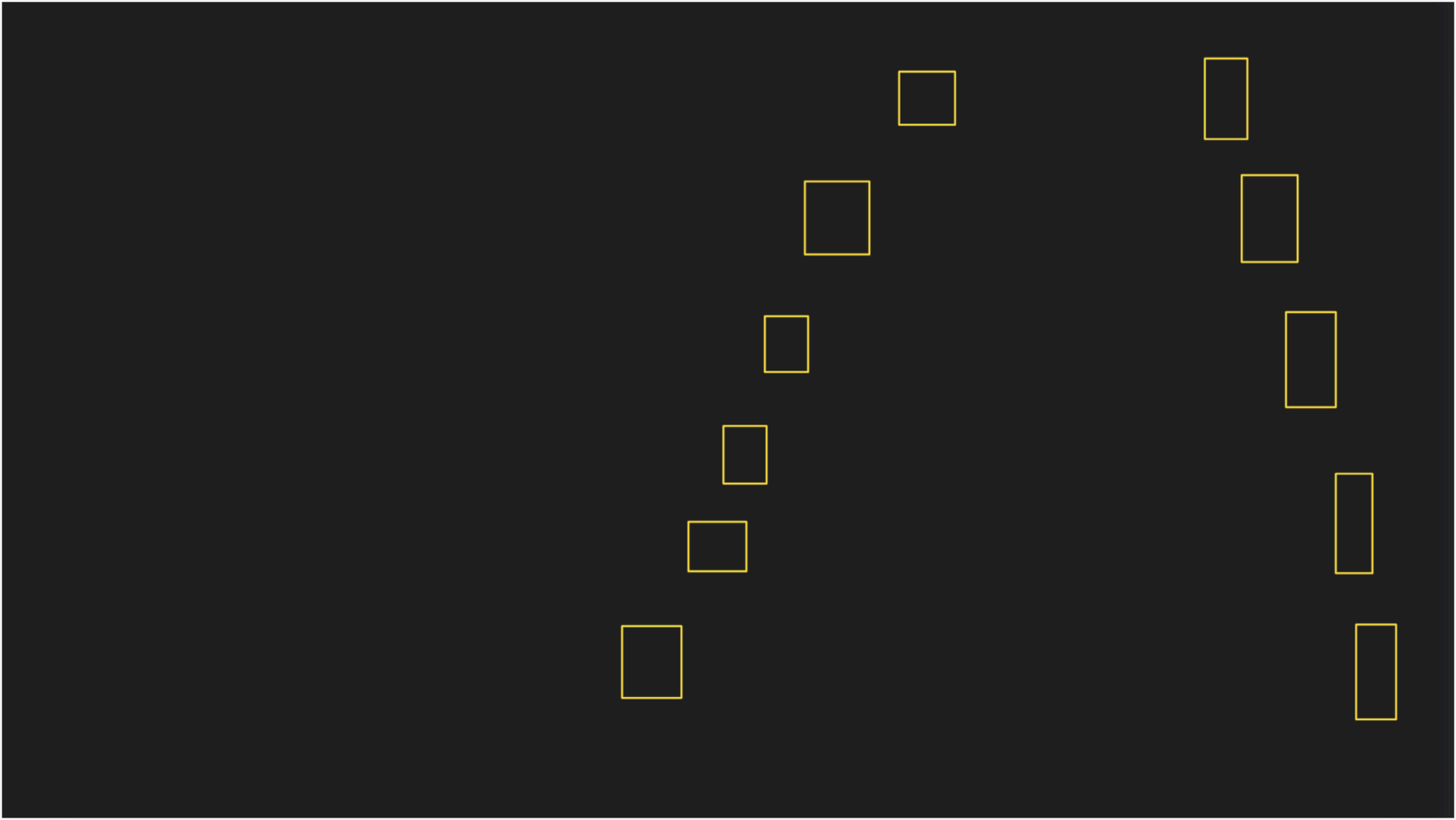
i/p

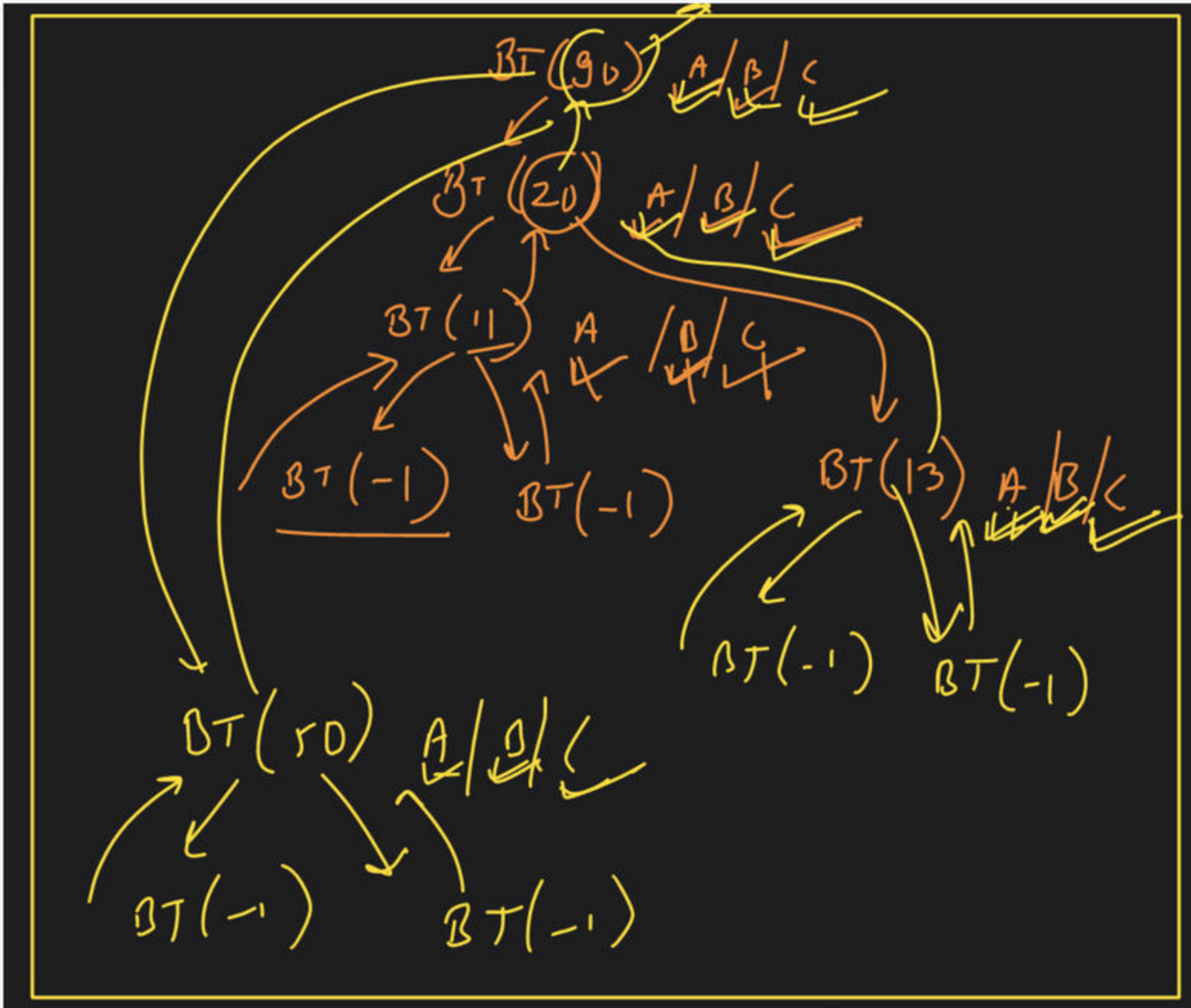
| = -1 -> Valid data

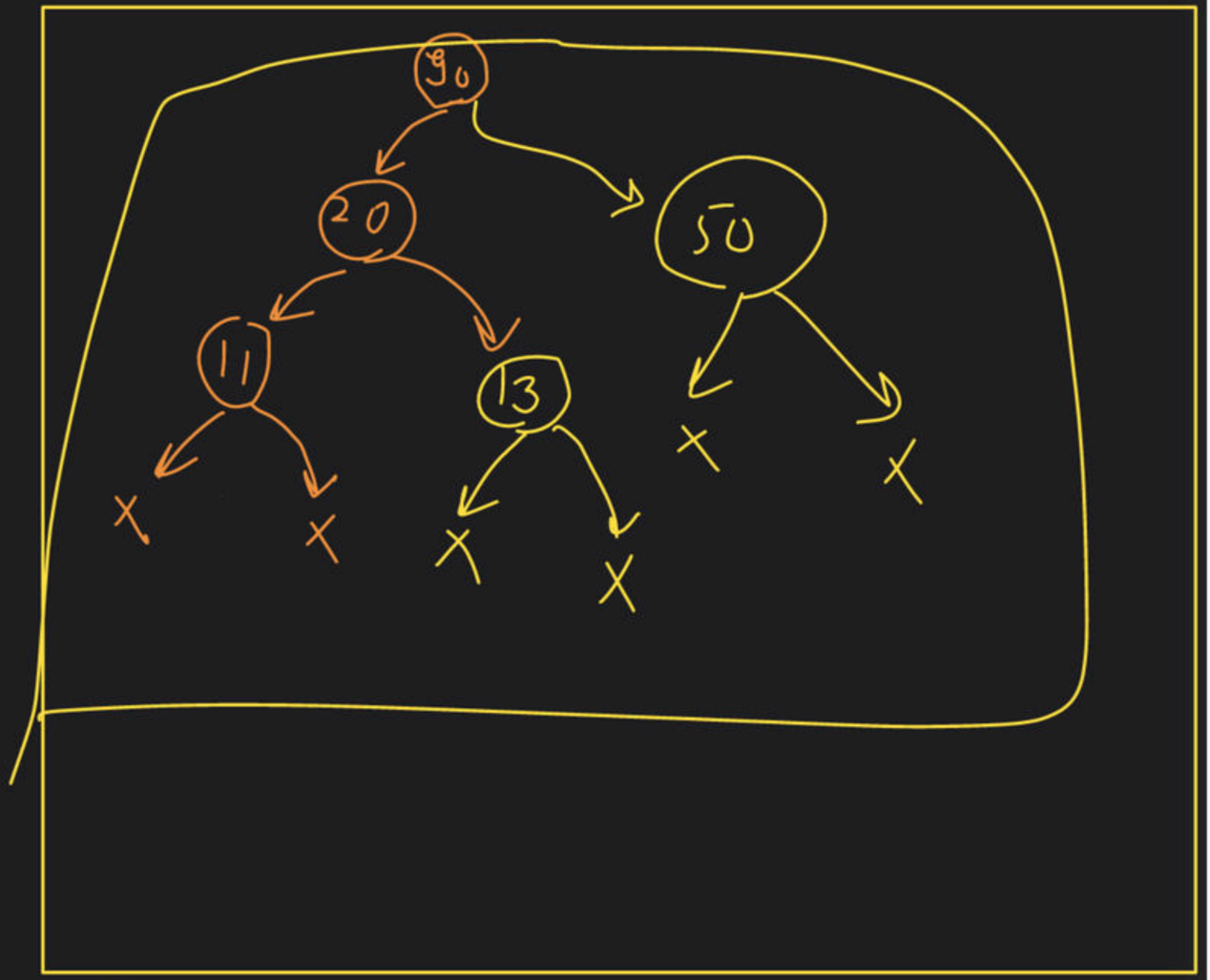
= = -1 -> NULL return

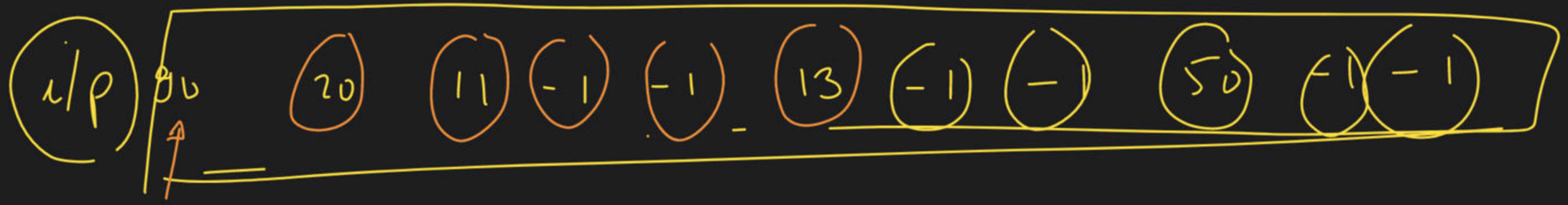
20 30 50 -1 60 -1 -1 40 -1 -1 100 -1 -1


buildTree(20)
A/B/C
BT(100)
A/B/C
BT(-1)
buildTree(30)
A/B/C
BT(40)
BT(-1)
BT(-1)
buildTree(50)
A/B/C
BT(-1)
A
BT(60)
A/B/C
BT(-1)
BT(-1)
20
30
100
50
40
60
20 30 50 -1 60 -1 -1 40 -1 -1 100 -1 -1



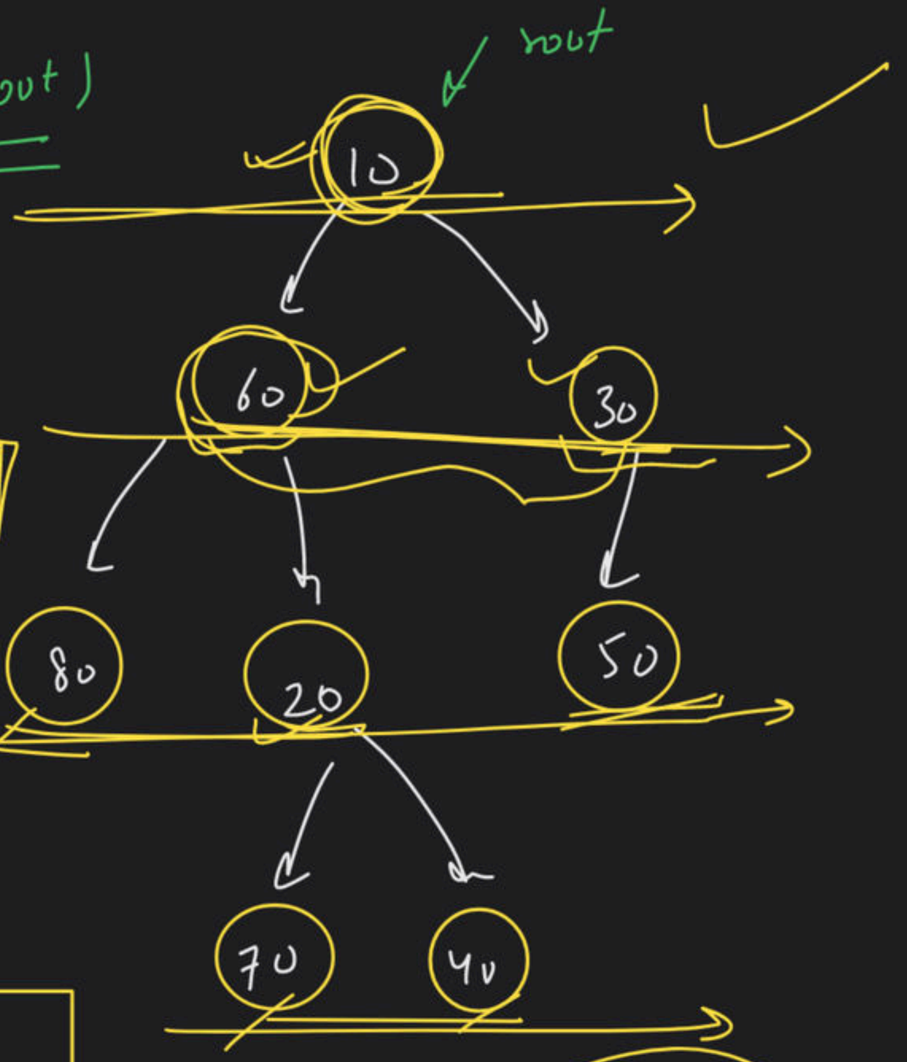

BT(90)
A/B/C
BT(20)
A/B/C
BT(11)
A/B/C
BT(-1)
BT(-1)
BT(13)
A/B/C
BT(-1)
BT(-1)
BT(50)
A/B/C
BT(-1)
BT(-1)


90
20
50
11
13
X
X
X
X
X
X


i/p
90
20
11
-1
-1
13
-1
-
50
-1
-1

# Level Order Traversal

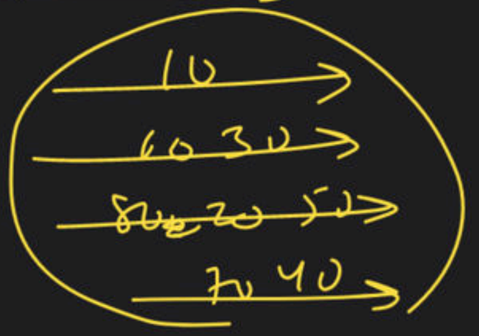
q.push(root)



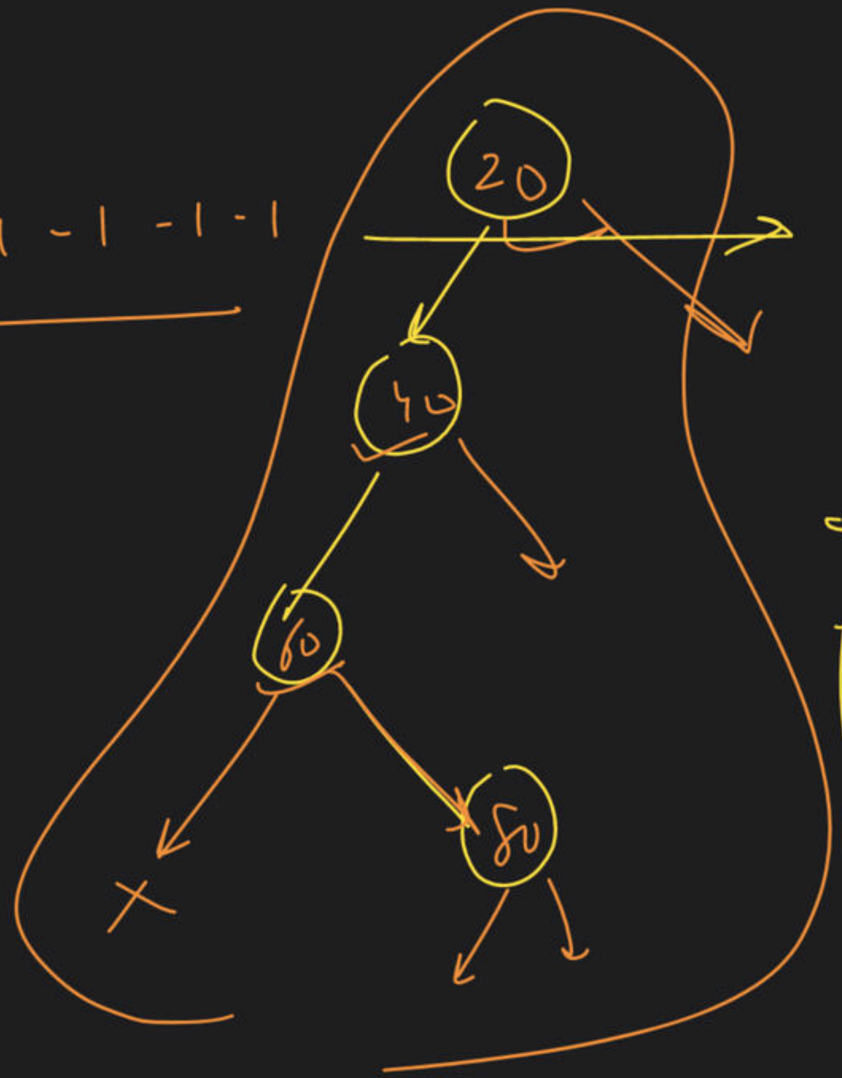

ans → 10 60 30 80 20 50 70 40

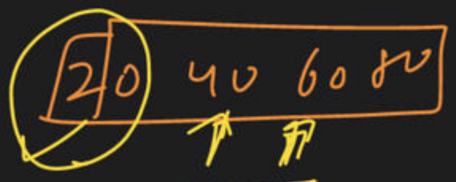
Logic

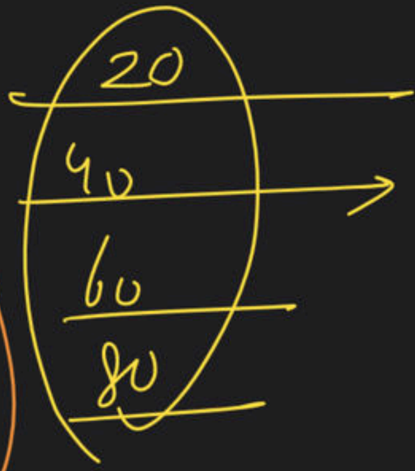
(4) temp = q.front()
(3) child insert
(2) q.pop()
(1) print ans



20 40 60 -1 80 -1 -1 -1 -1



NULL → (1) Purana Level complete ho chuka hai

↳ cout << endl;

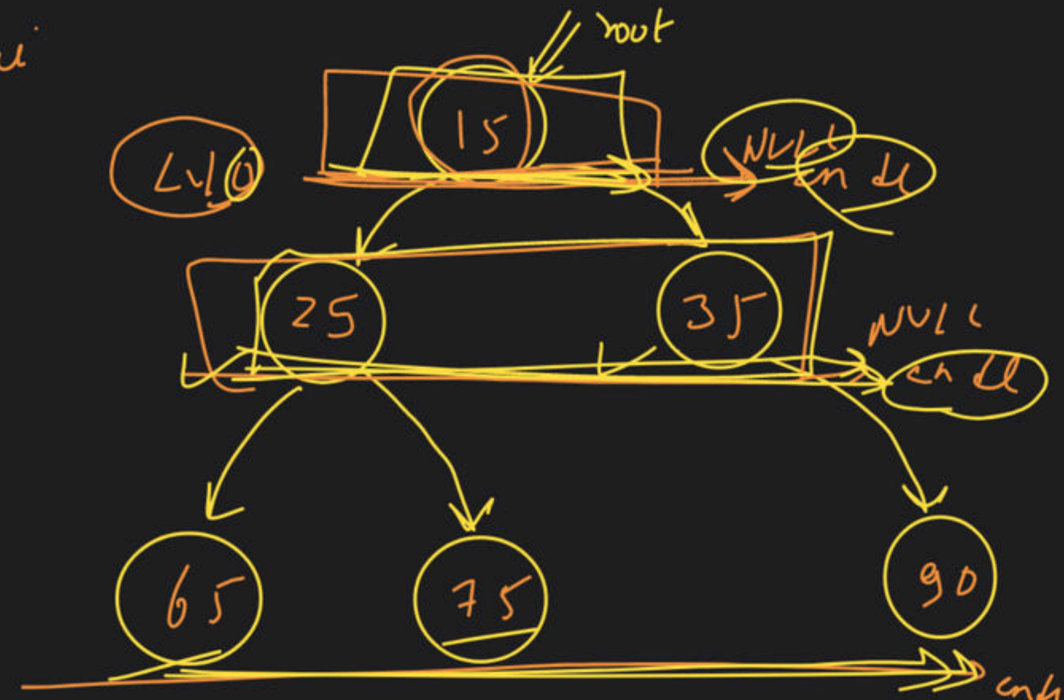
(2) Push NULL → condn

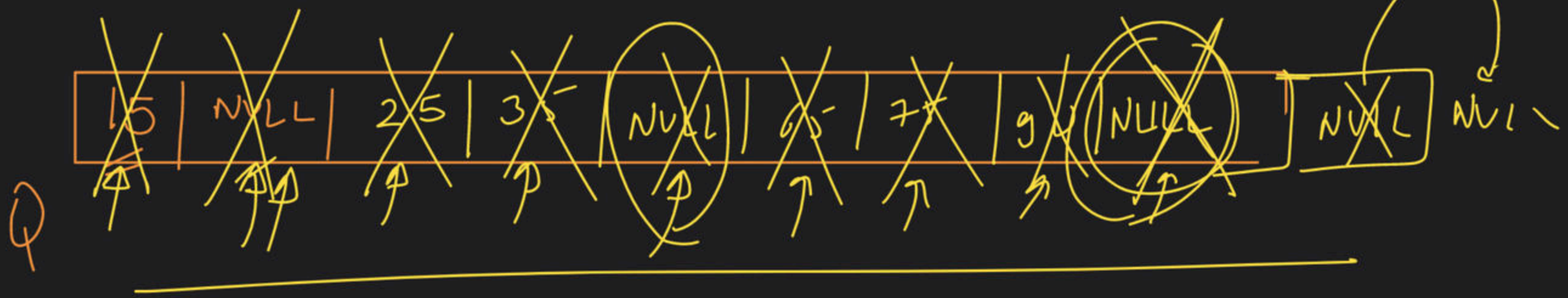
!q.empty

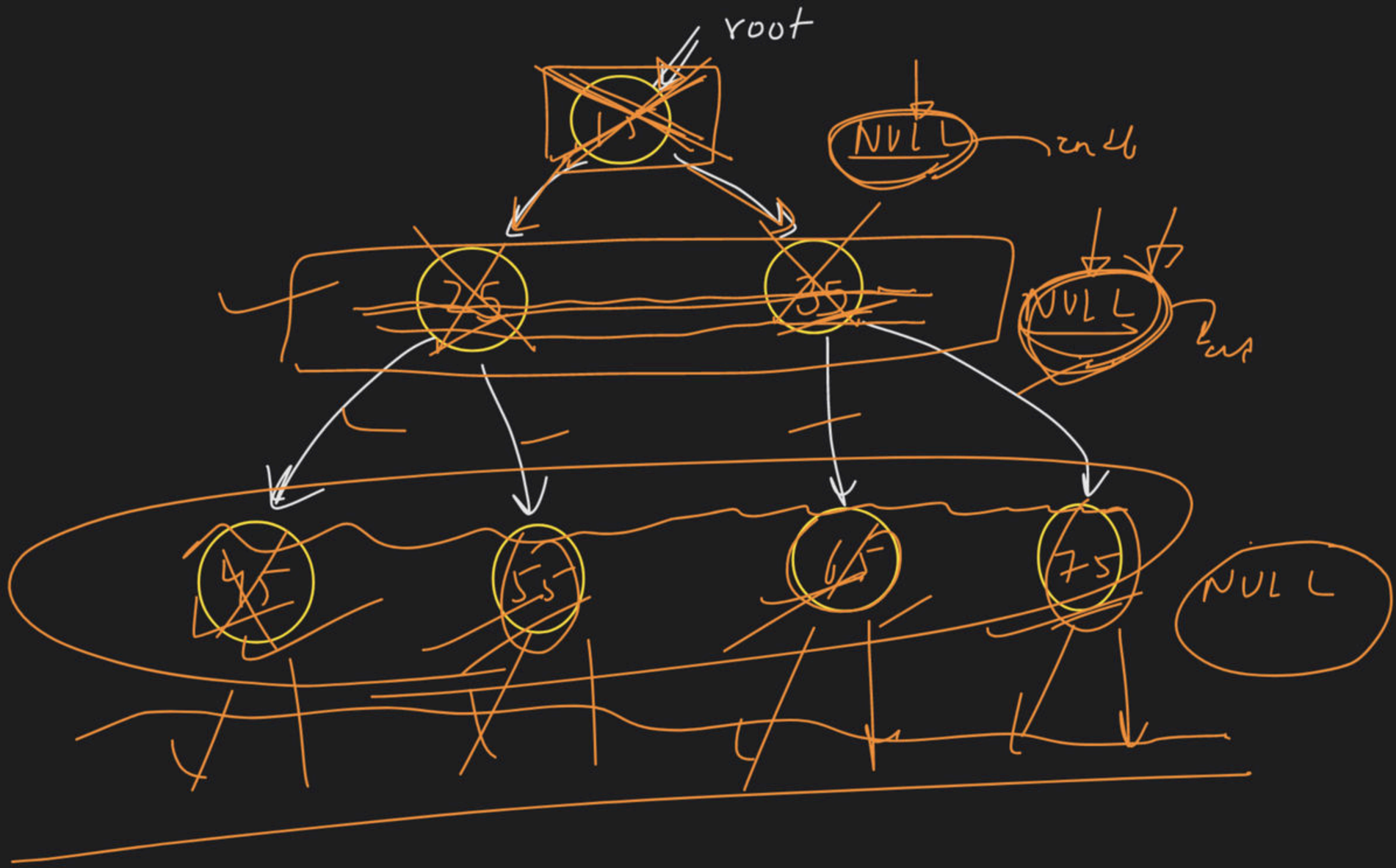
15

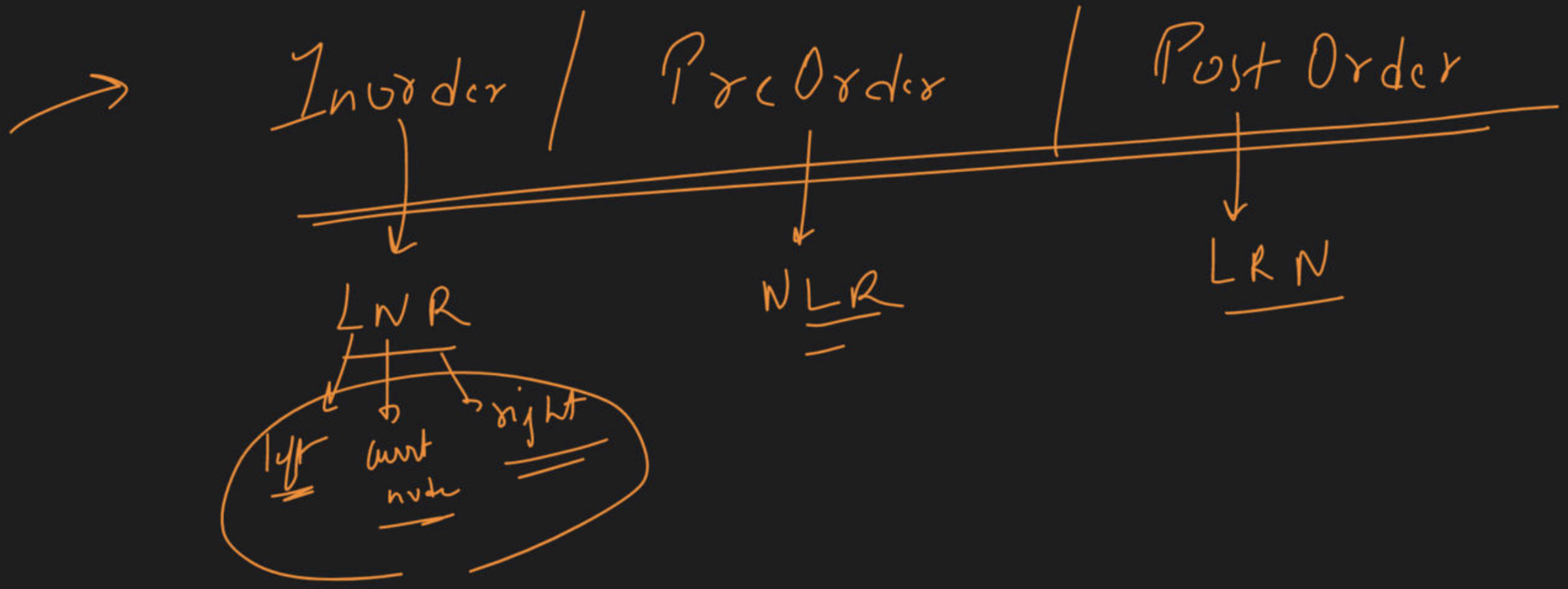
25 35

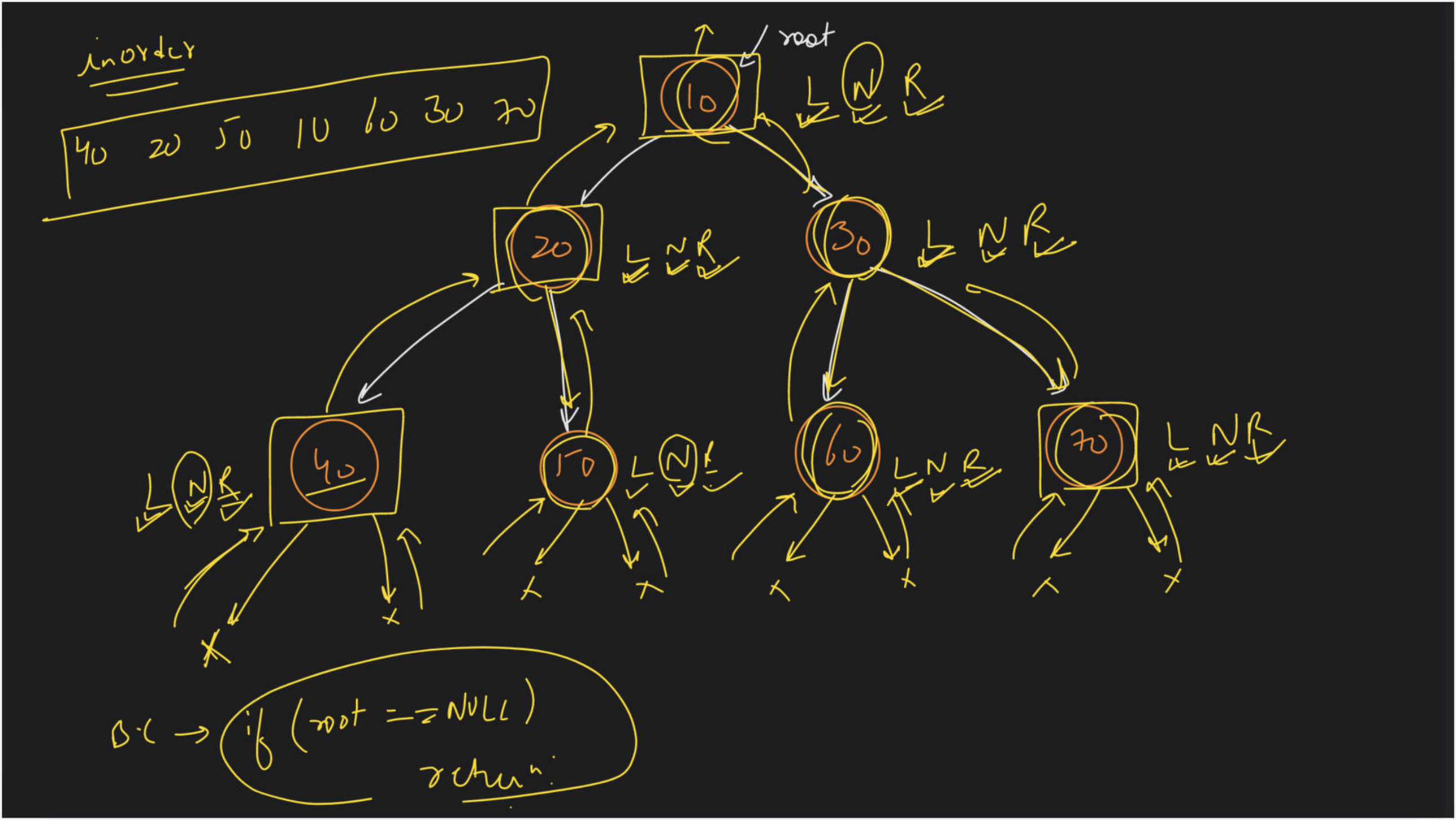
65 75 90

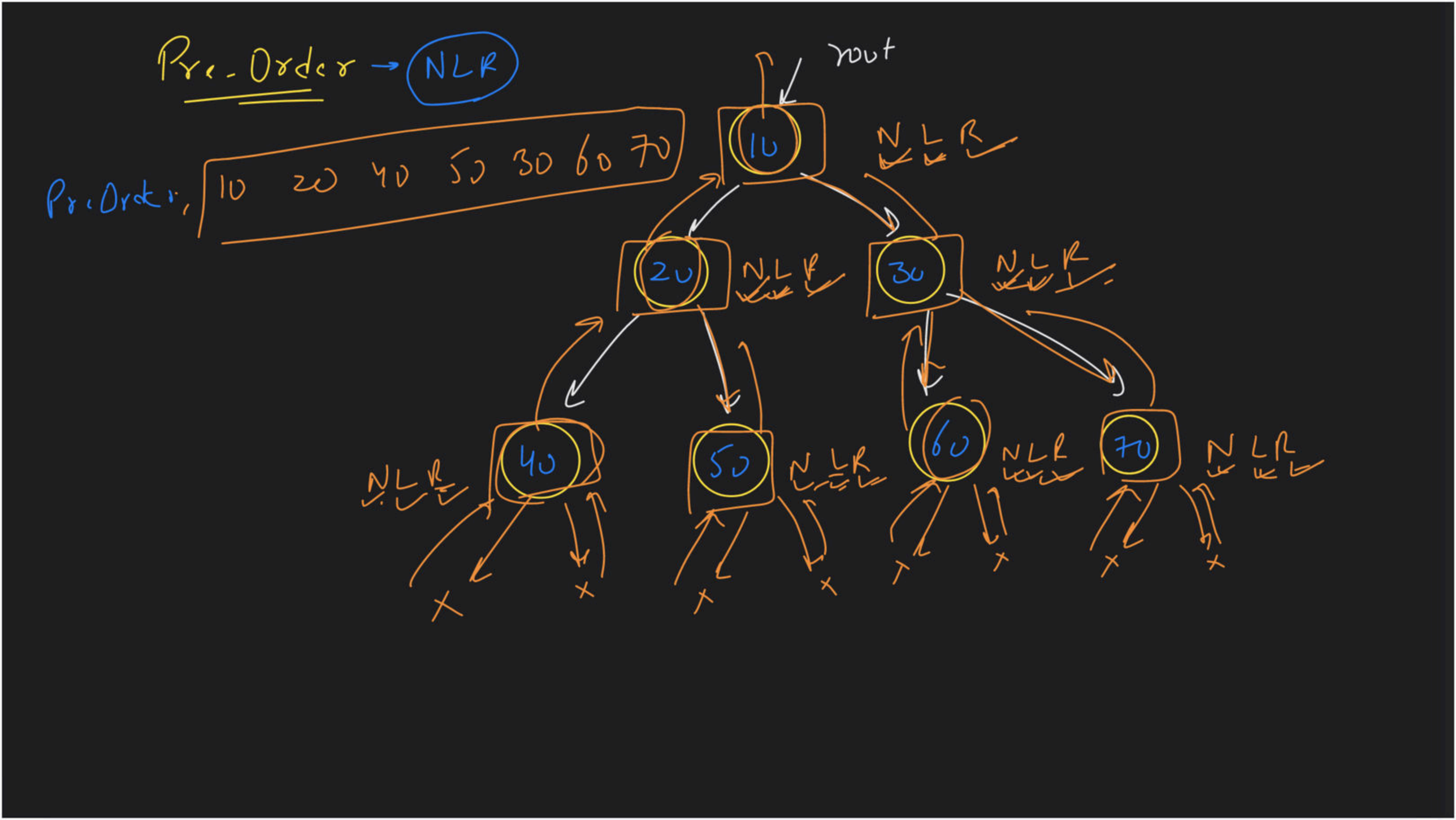


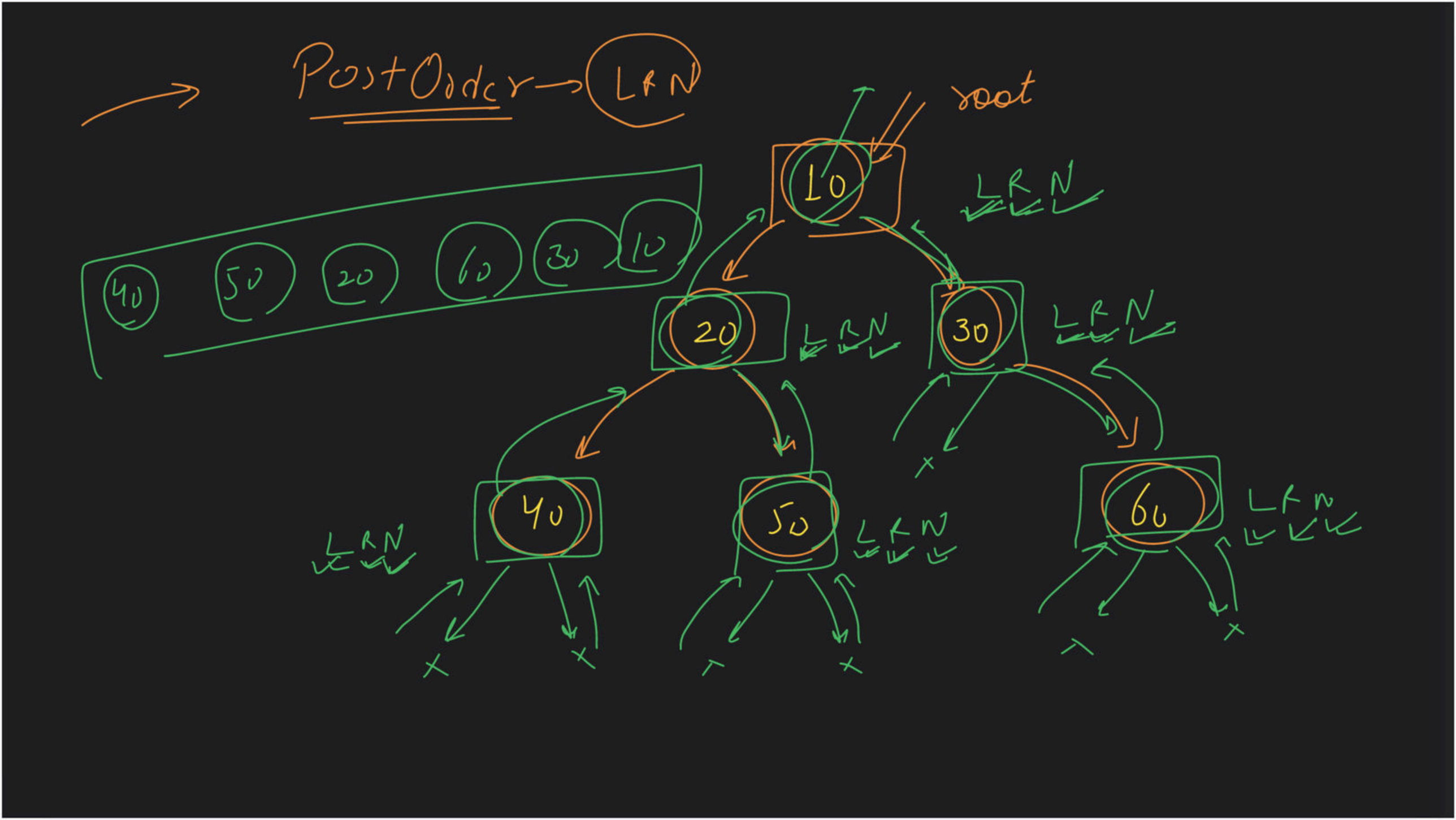



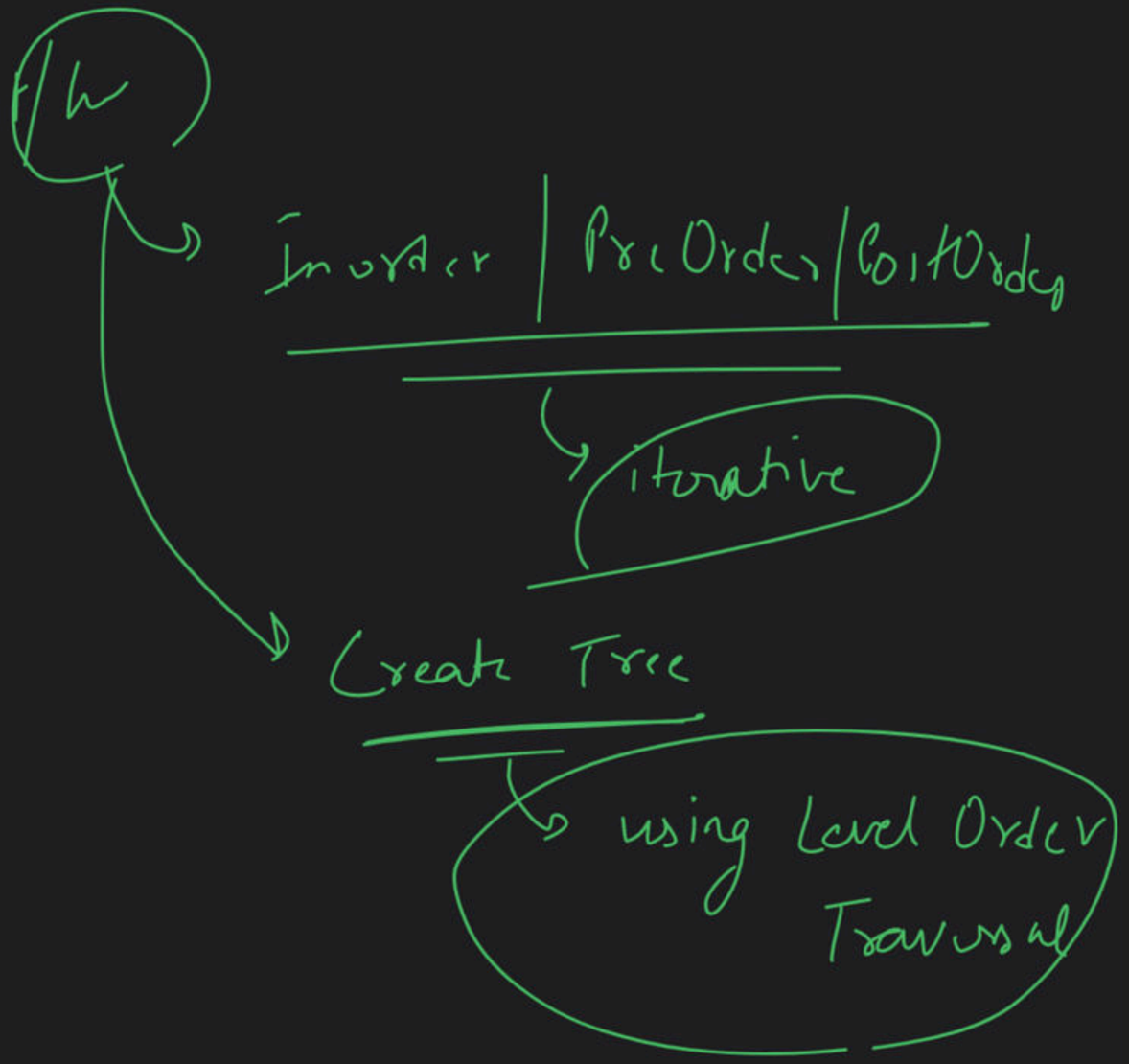

root
NULL
NULL
NULL


Inorder
PreOrder
Post Order
LNR
NLR
LRN
left
curr node
right

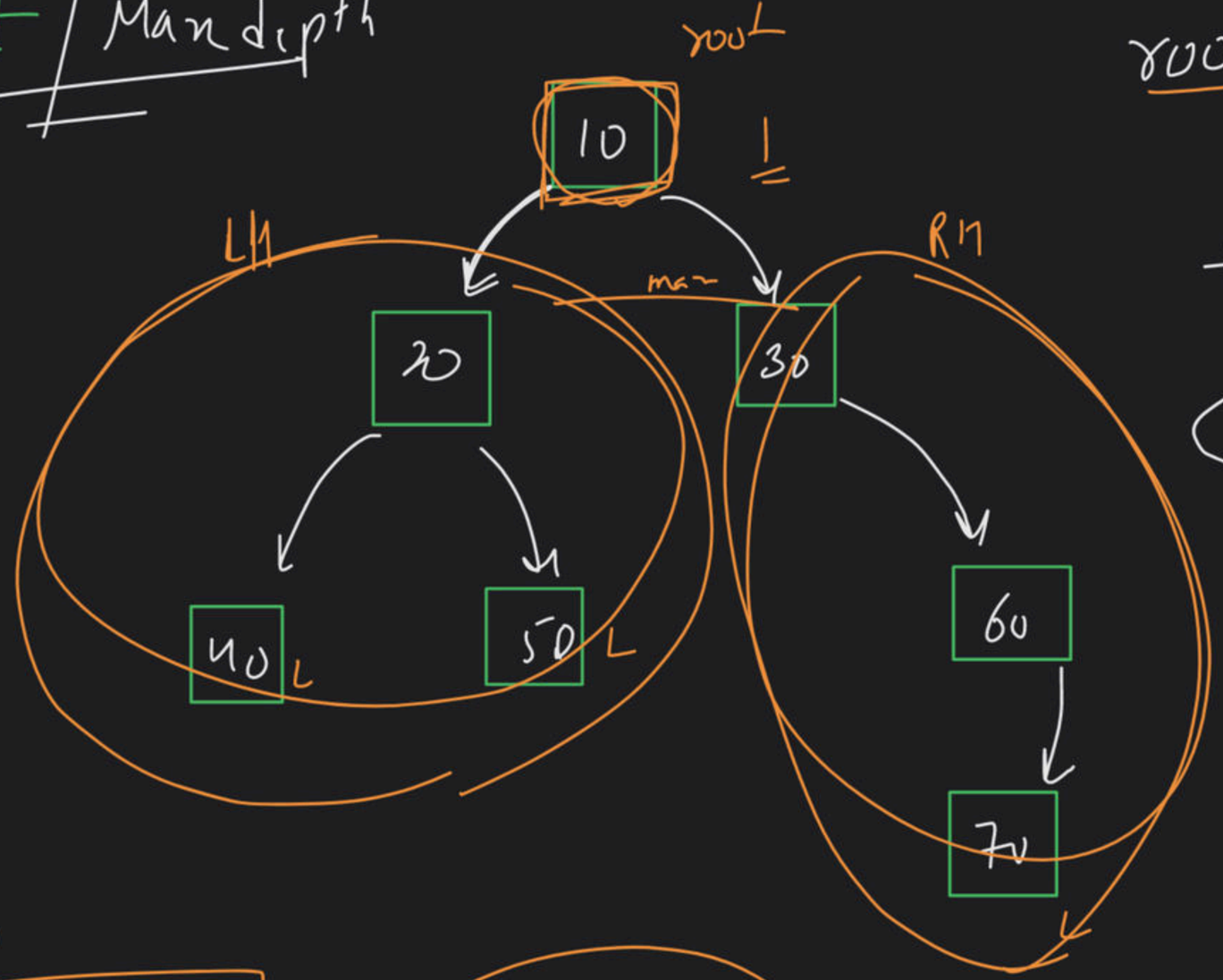

inorder
40 20 50 10 60 30 70
root
10
20
30
40
50
60
70
L N R
if (root == NULL)
return;
B.C →


Pre-Order → NLR
Pre Order: 10 20 40 50 30 60 70
root
10
20
30
40
50
60
70
N L R
X


PostOrder → LRN
40
50
20
60
30
10
root
10
20
30
40
50
60
LRN
X


f/w
InOrder | PreOrder | PostOrder
Iterative
Create Tree
using Level Order Traversal

# Height / Max depth




root → leaf

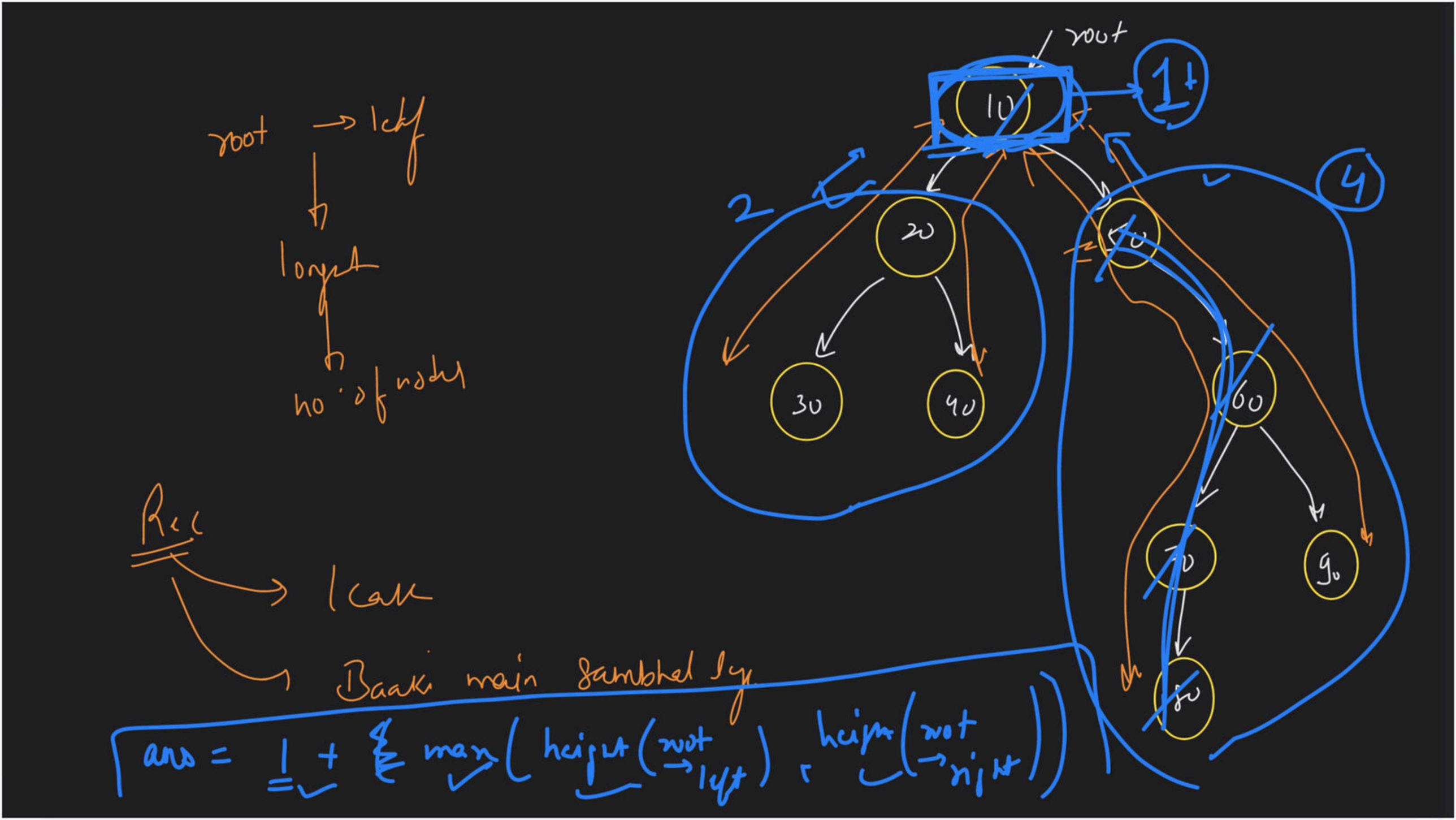
farthest

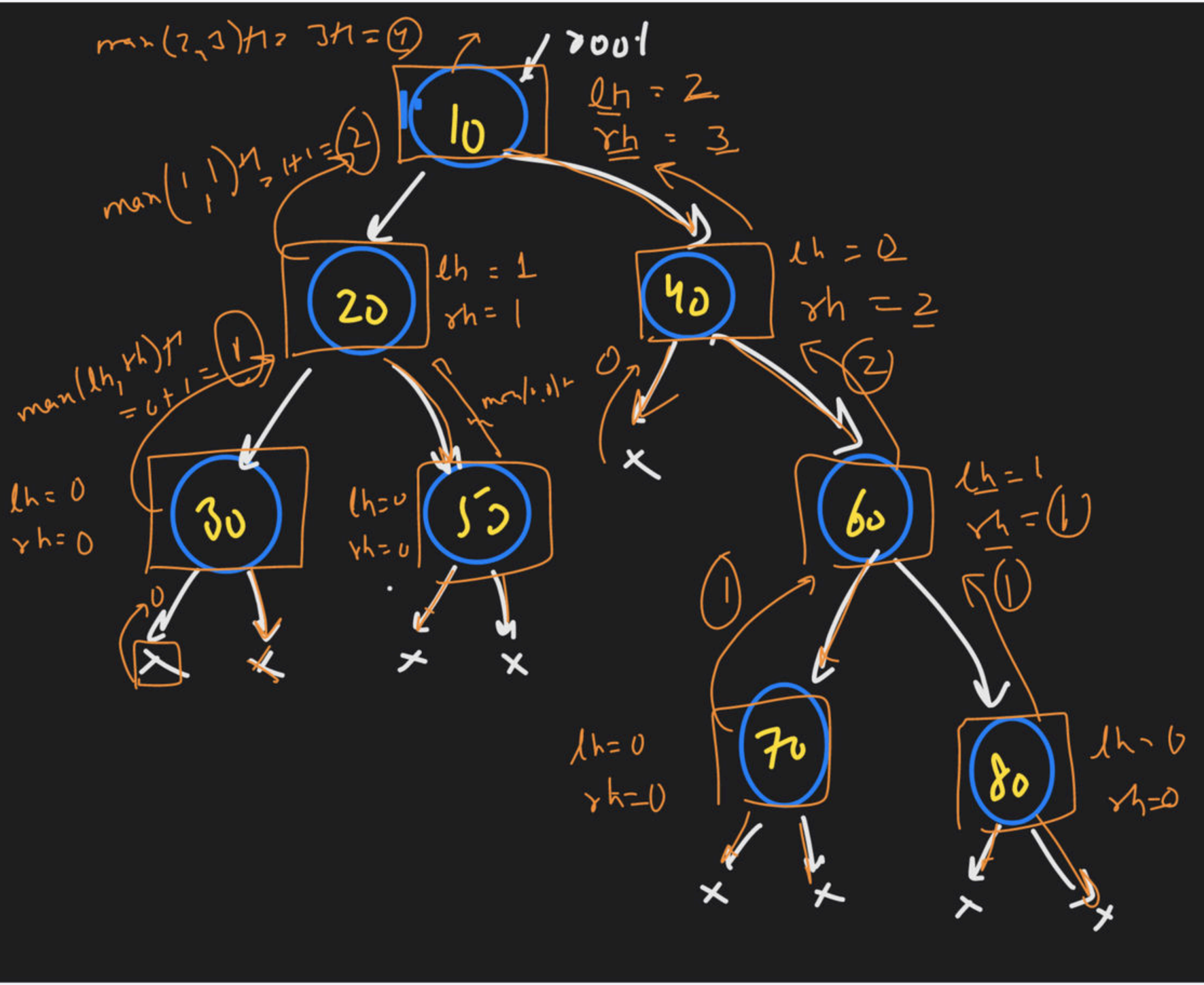
no of nodes

10 → 20 → 40

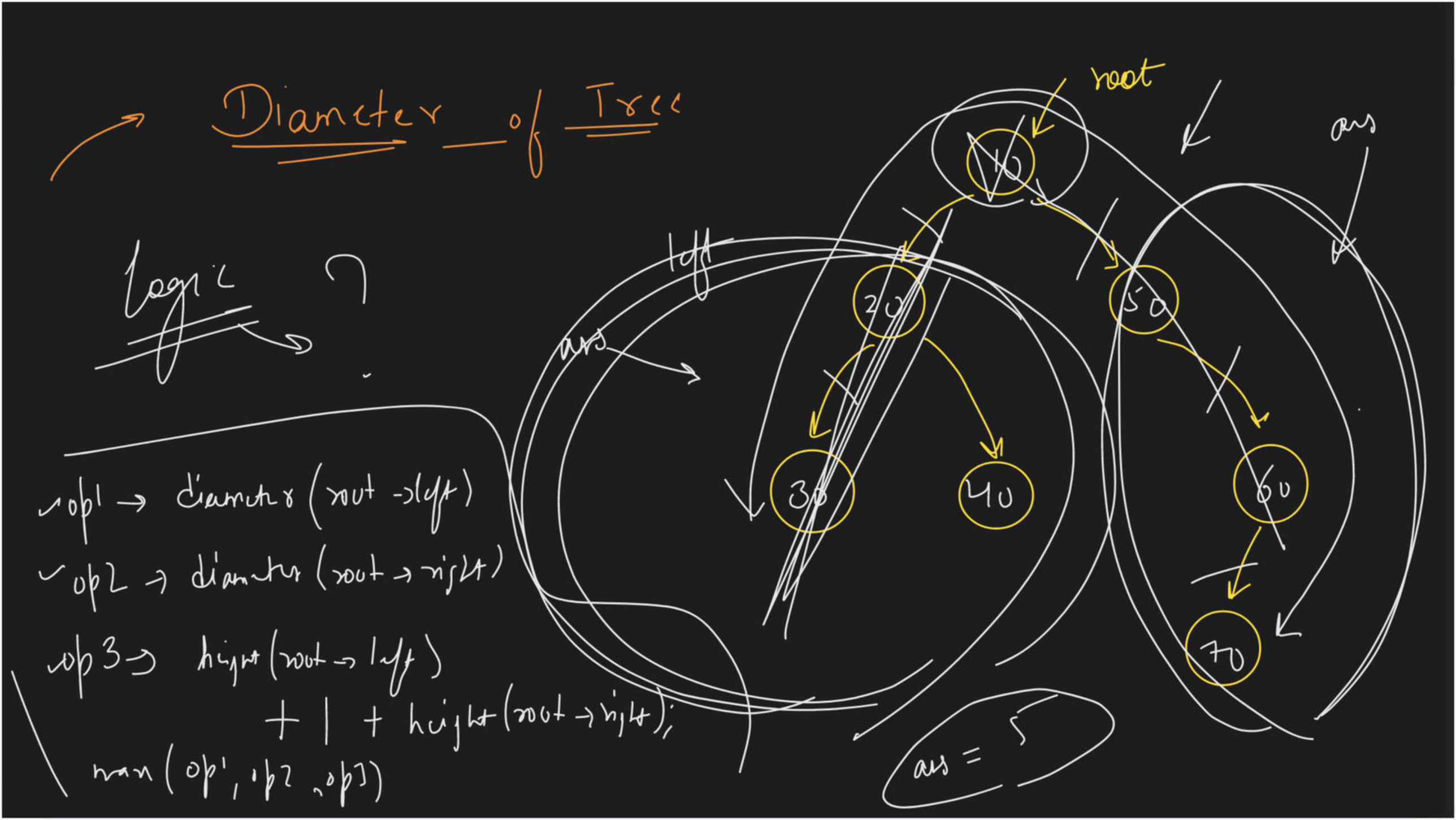
10 → 20 → 50

10 → 30 → 60 → 70 → Height = 4


root → left
height
no. of nodes
Rec
call
Baaki main sambhal lega
root
10
1+
2
20
30
40
50
4
60
70
80
90
ans = 1 + max( height(root → left) , height(root → right) )


root
10
lh = 2
rh = 3
20
lh = 1
rh = 1
40
lh = 0
rh = 2
30
lh = 0
rh = 0
50
lh = 0
rh = 0
60
lh = 1
rh = 1
70
lh = 0
rh = 0
80
lh = 0
rh = 0

# Diameter of Tree




## Logic

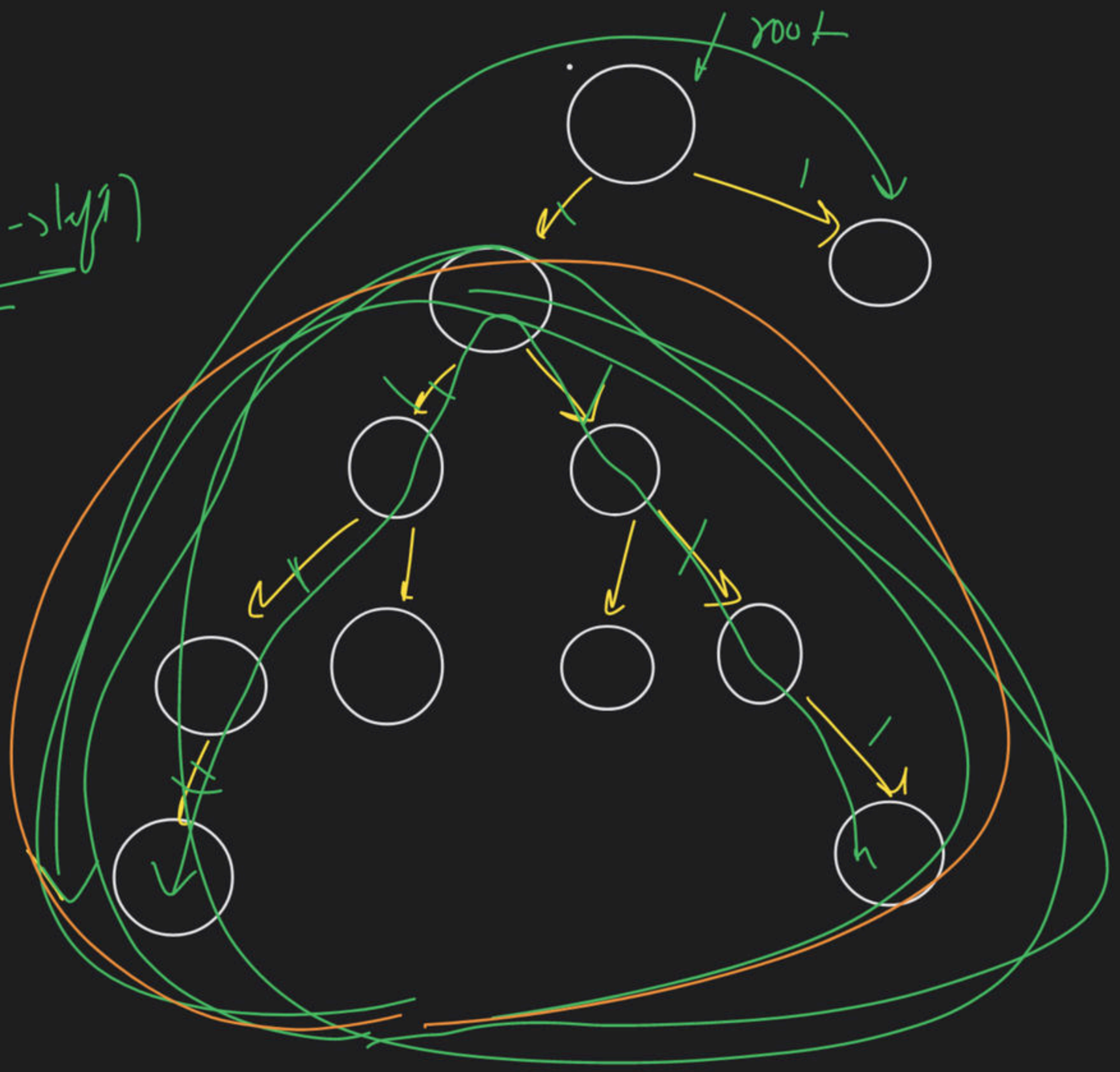
op1 → diameter(root->left)

op2 → diameter(root->right)

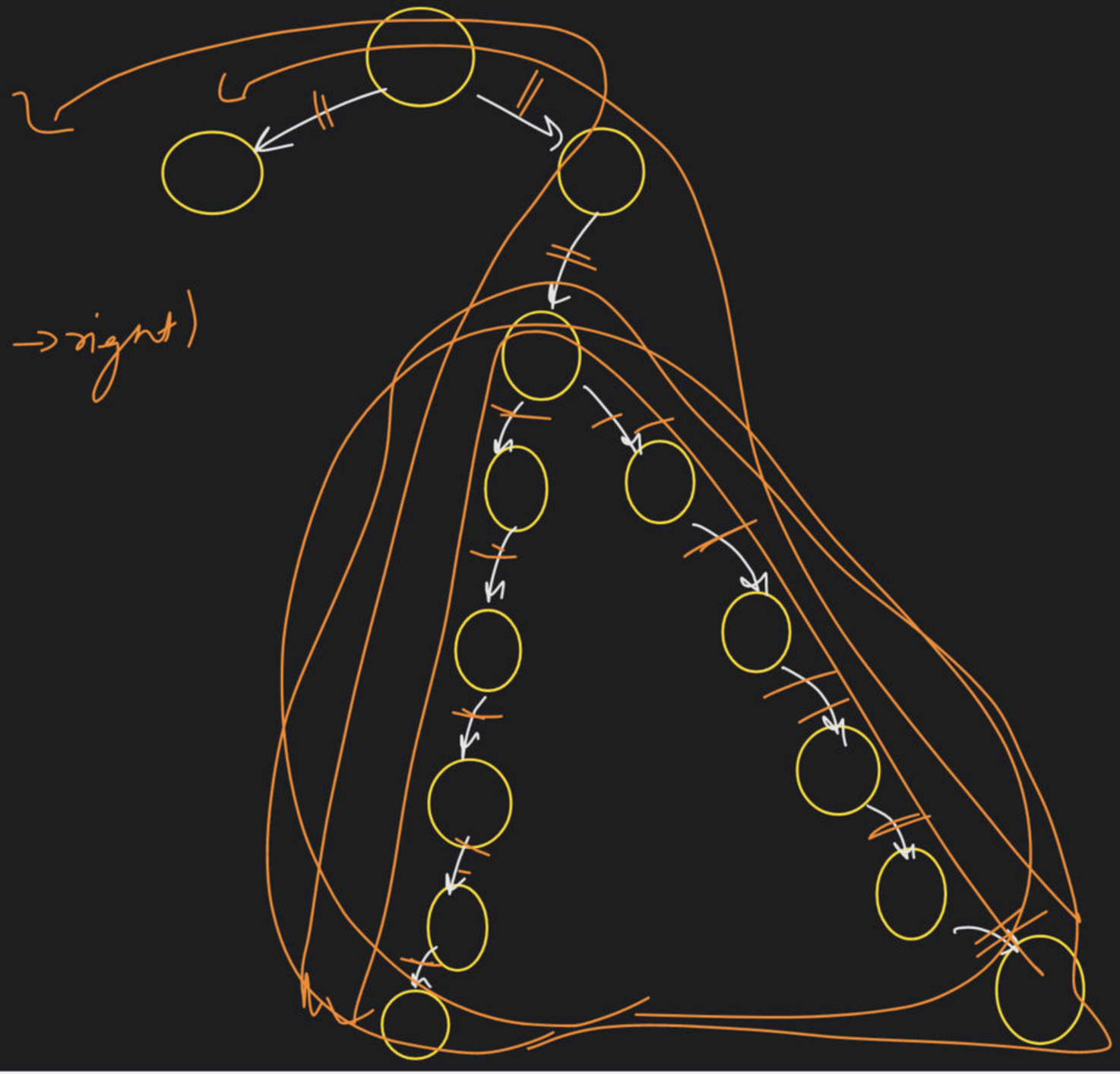
op3 → height(root->left) + 1 + height(root->right);

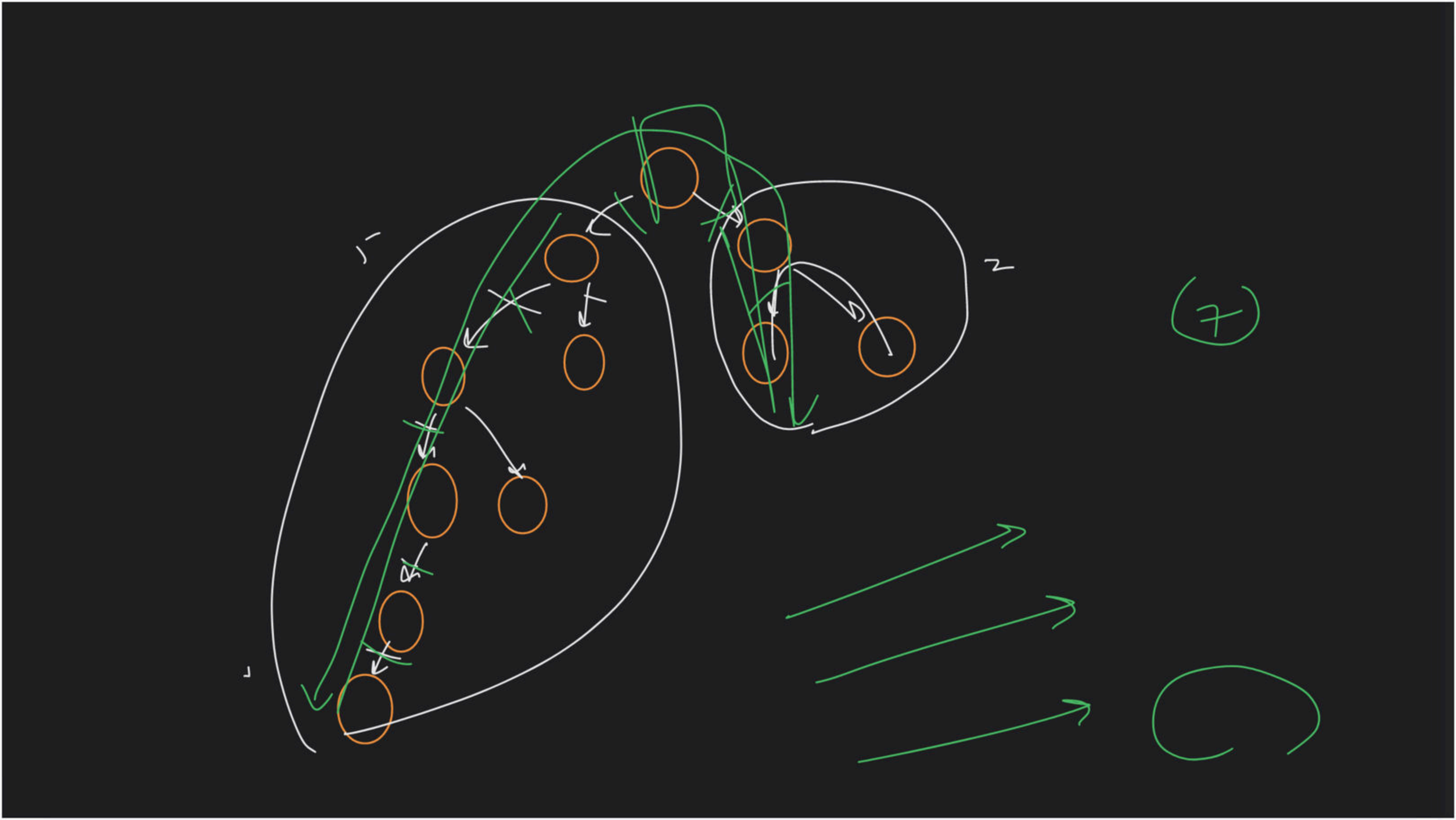
max(op1, op2, op3)

'int lpl = diameter(root->left)



int op2 = diameter (root -> right)

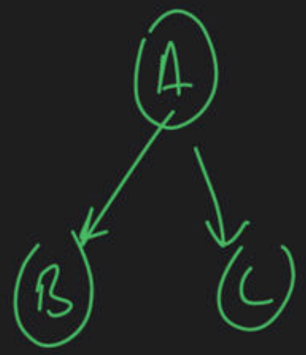


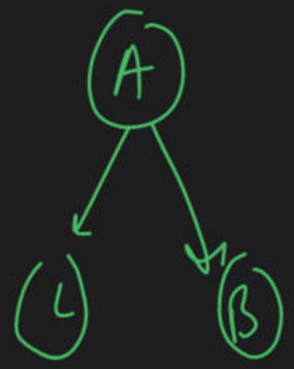


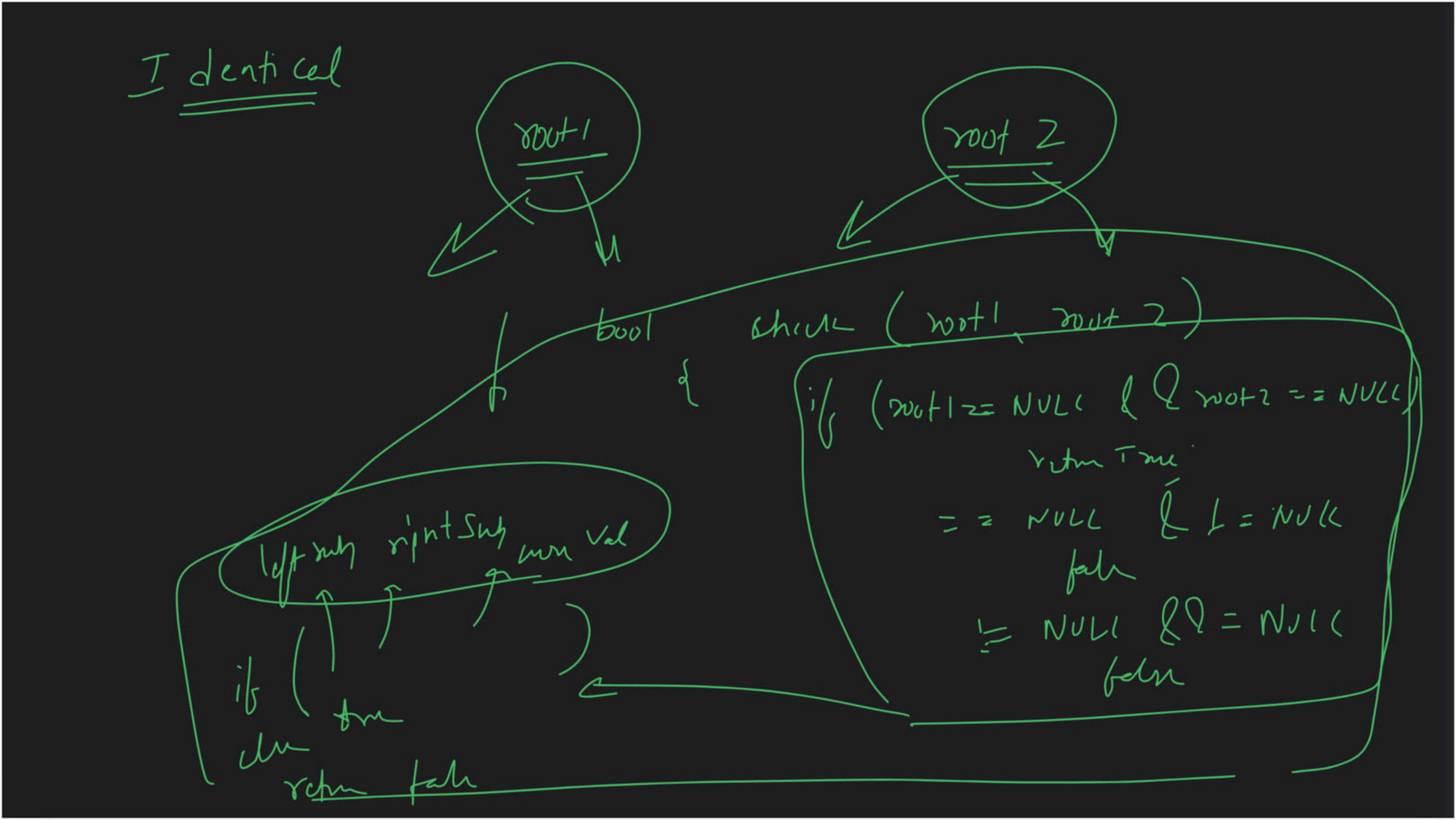
H/W

Check if two trees are identical or not

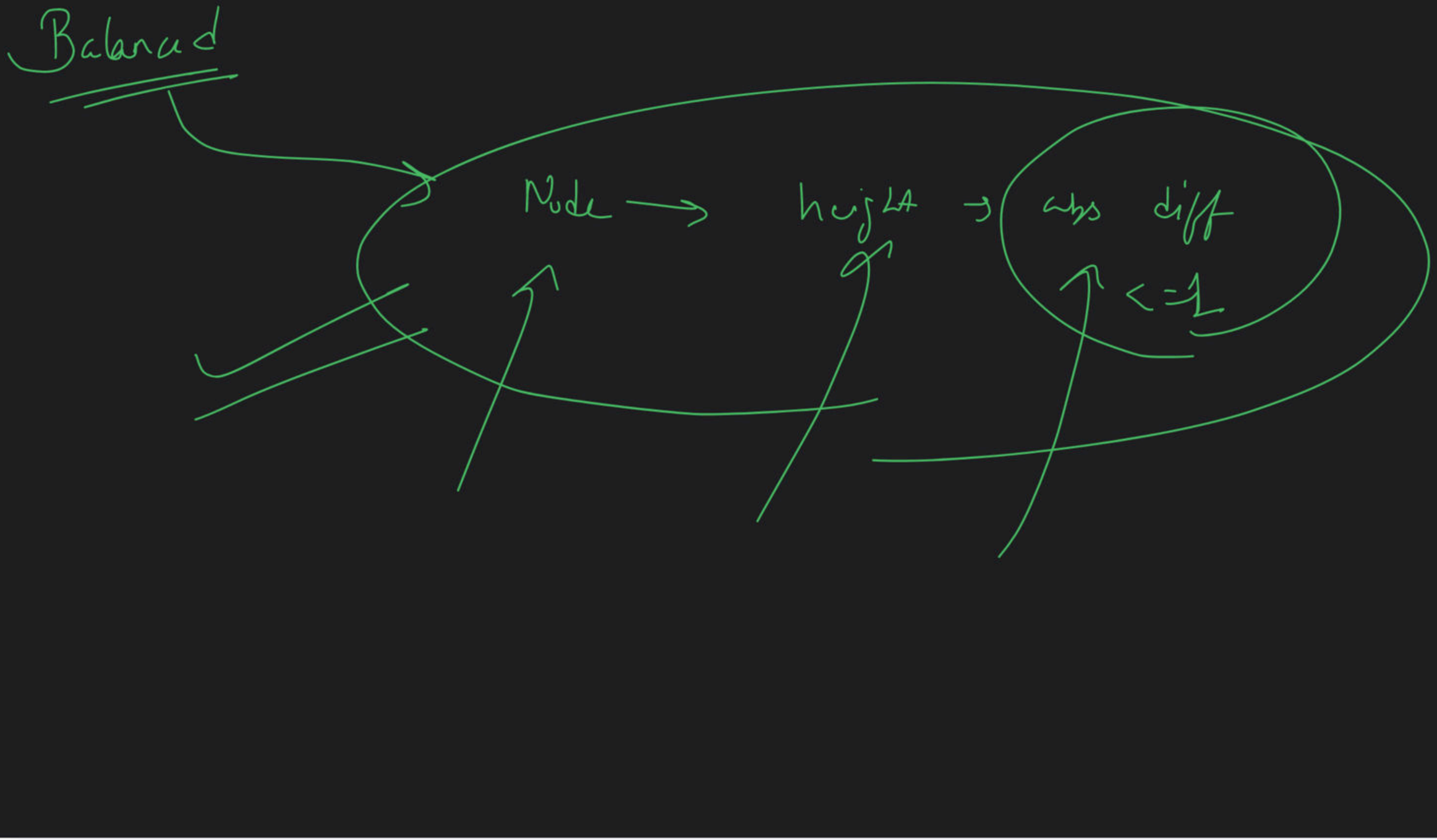
Check if 2 trees are mirror of each other or not




Identical
root1
root 2
bool check ( root1 , root 2 )
{
if (root1 == NULL && root2 == NULL)
return True
== NULL && 1 = NULL
false
!= NULL && 0 = NULL
false
leftSub rightSub currVal
if ( true
else
return false


Mirror
Yout1
Yout 2


Balanced
Node
height
abs diff
<=1