Recursion on Trees and Doubt Clearing Session - LIVE

Special class

Love Babbar

↑

~~Vishal Pachauri~~ • Sept 3, 2022

α

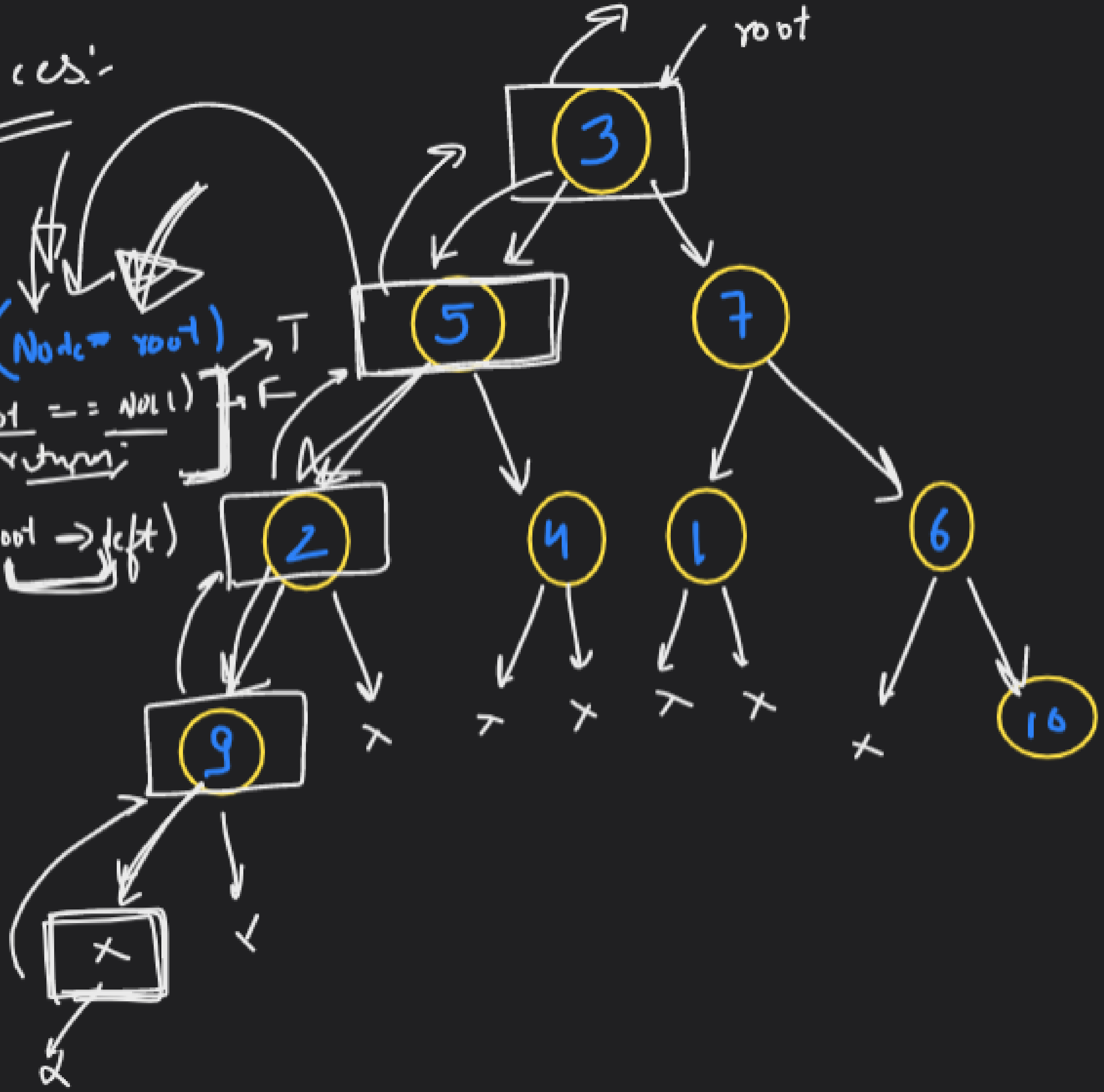
Recursion → what? types → /patterns

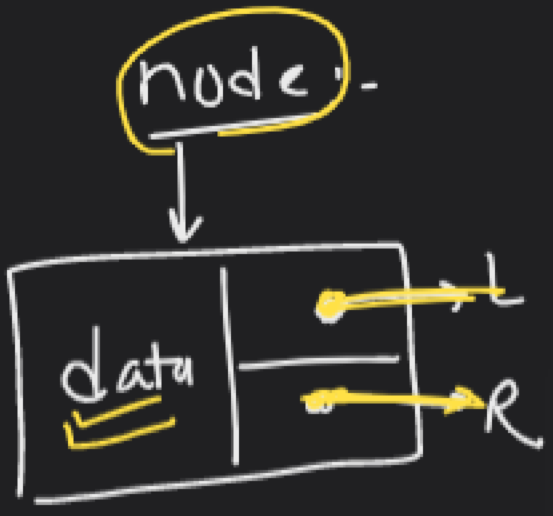
- arrays
- strings
- linked list

Trees

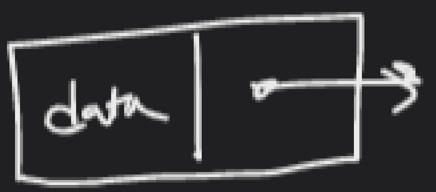
Trees:-

```
solve (Node* root)
{   if (root == NULL)
        return;

    solve (root -> left)
}
```

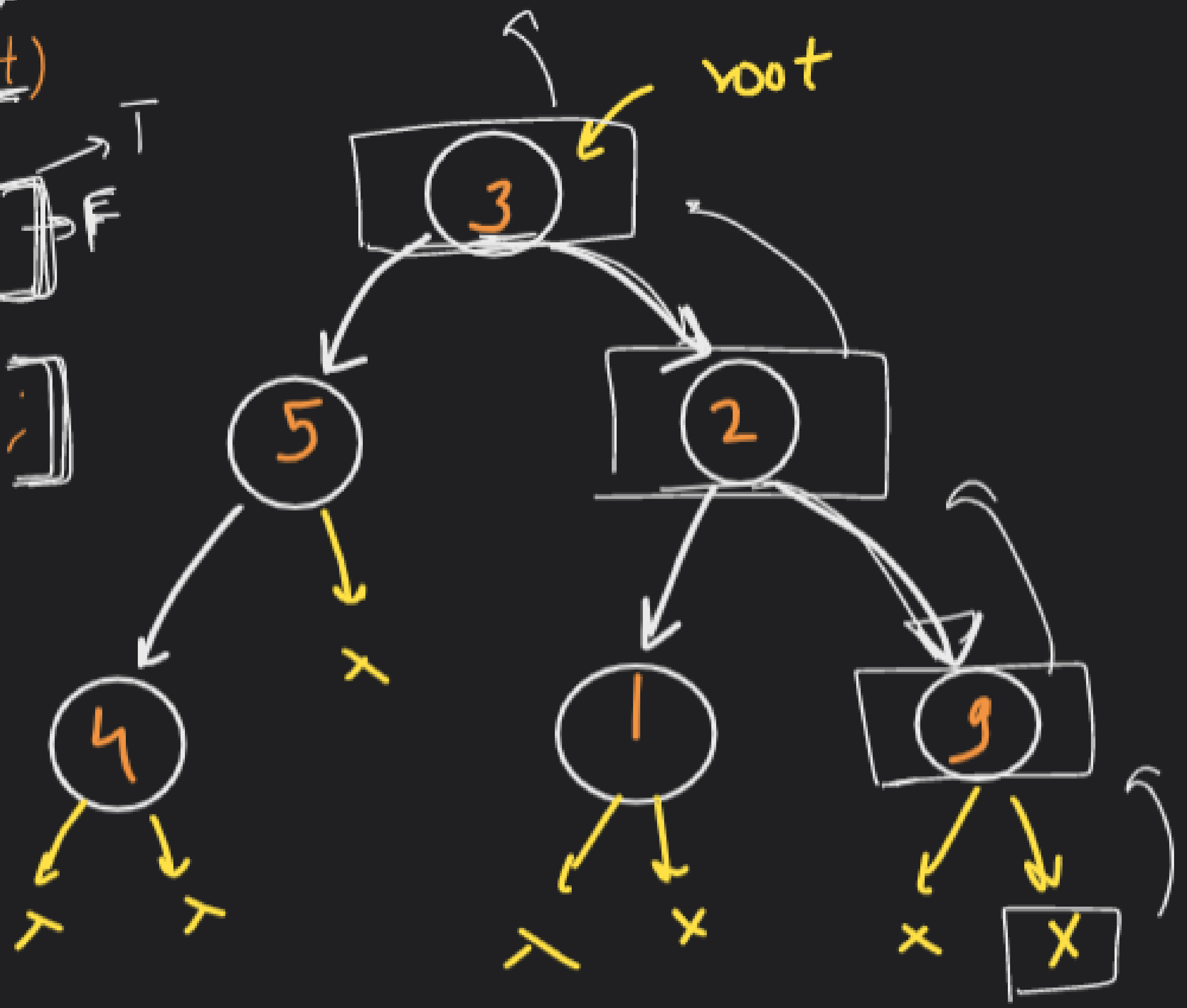




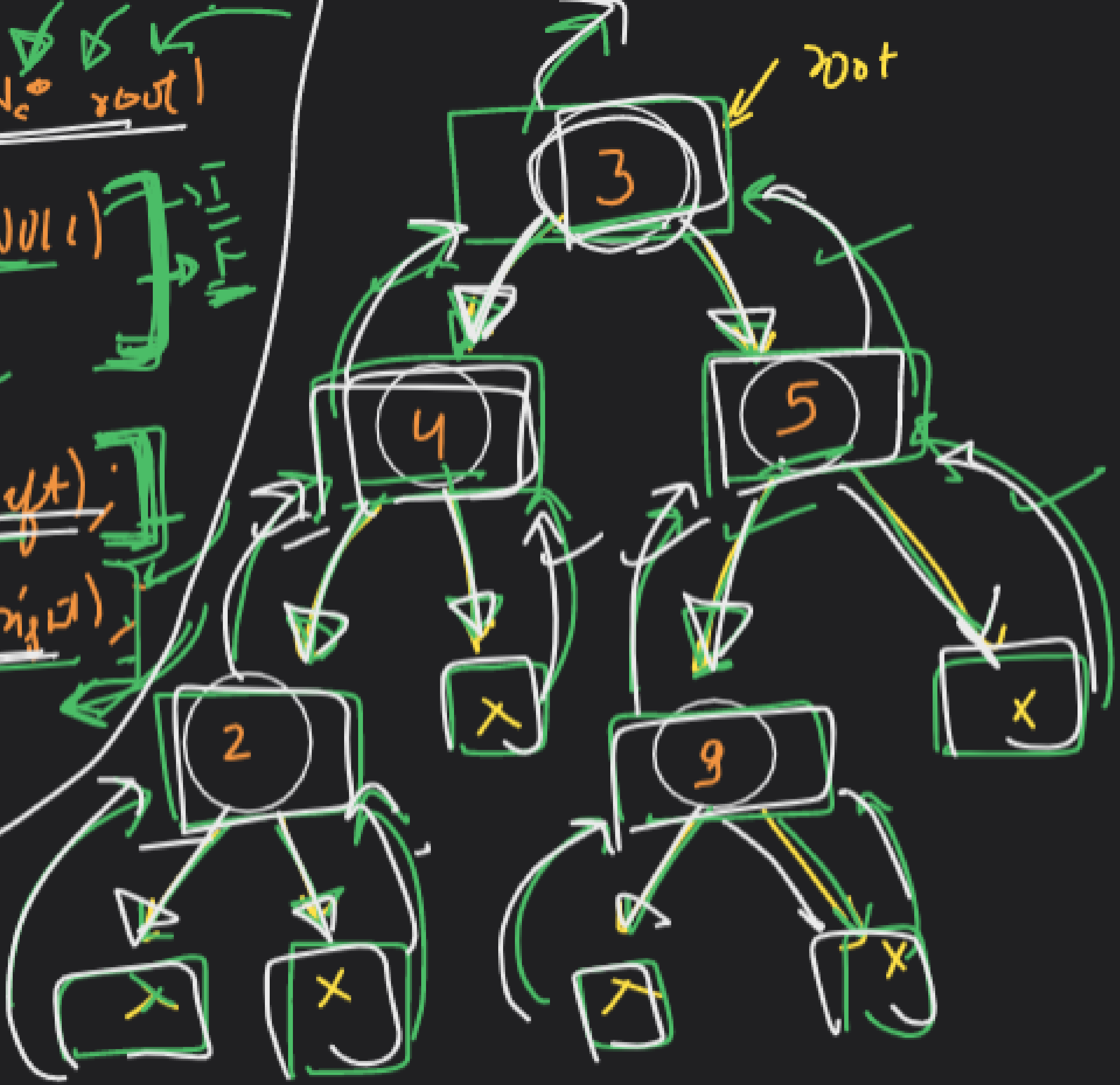
LL



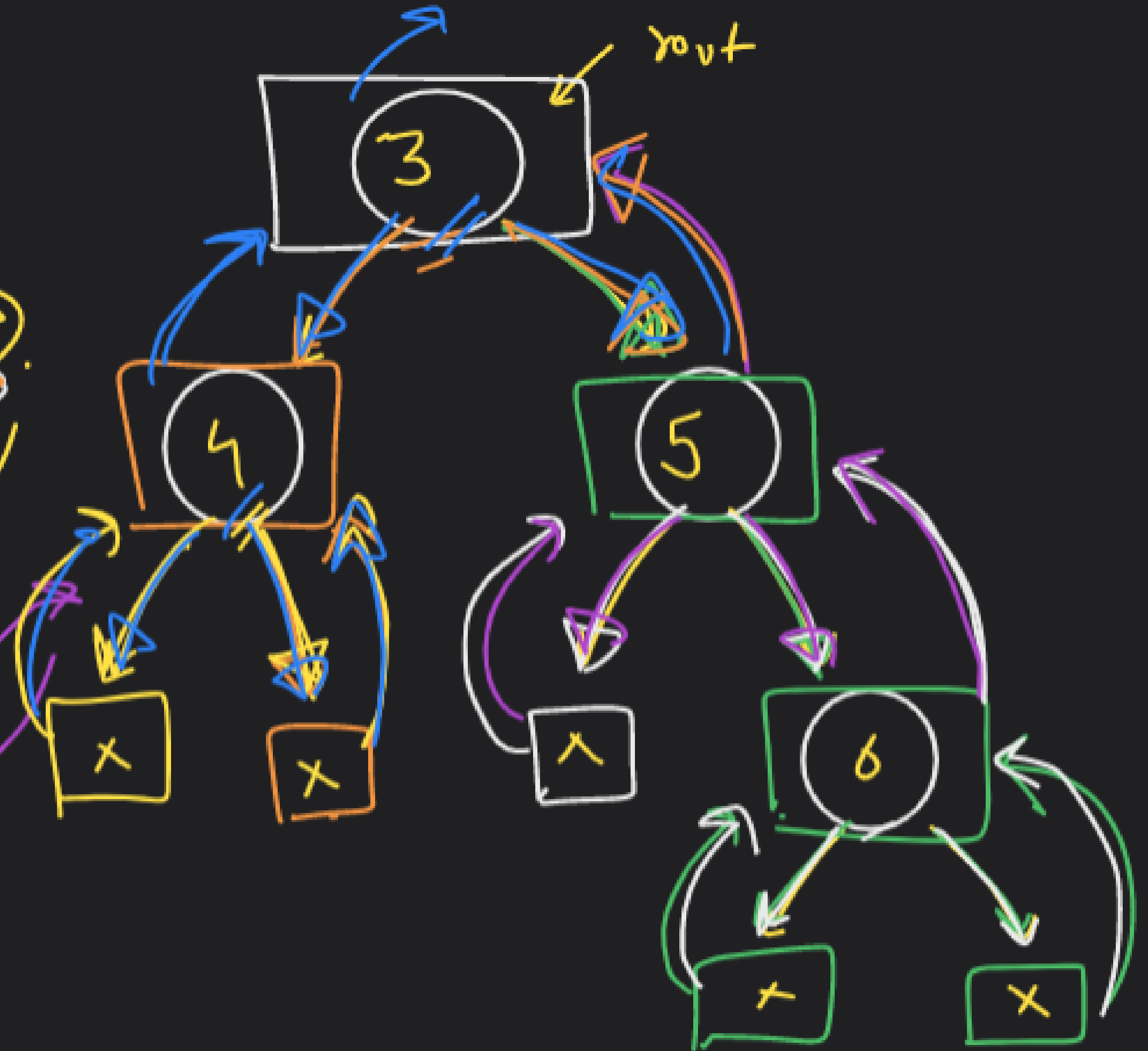
```
void solve (Node* root)
{
    if (root == NULL)        T
                             F
        return;

    solve(root->right);

}
```



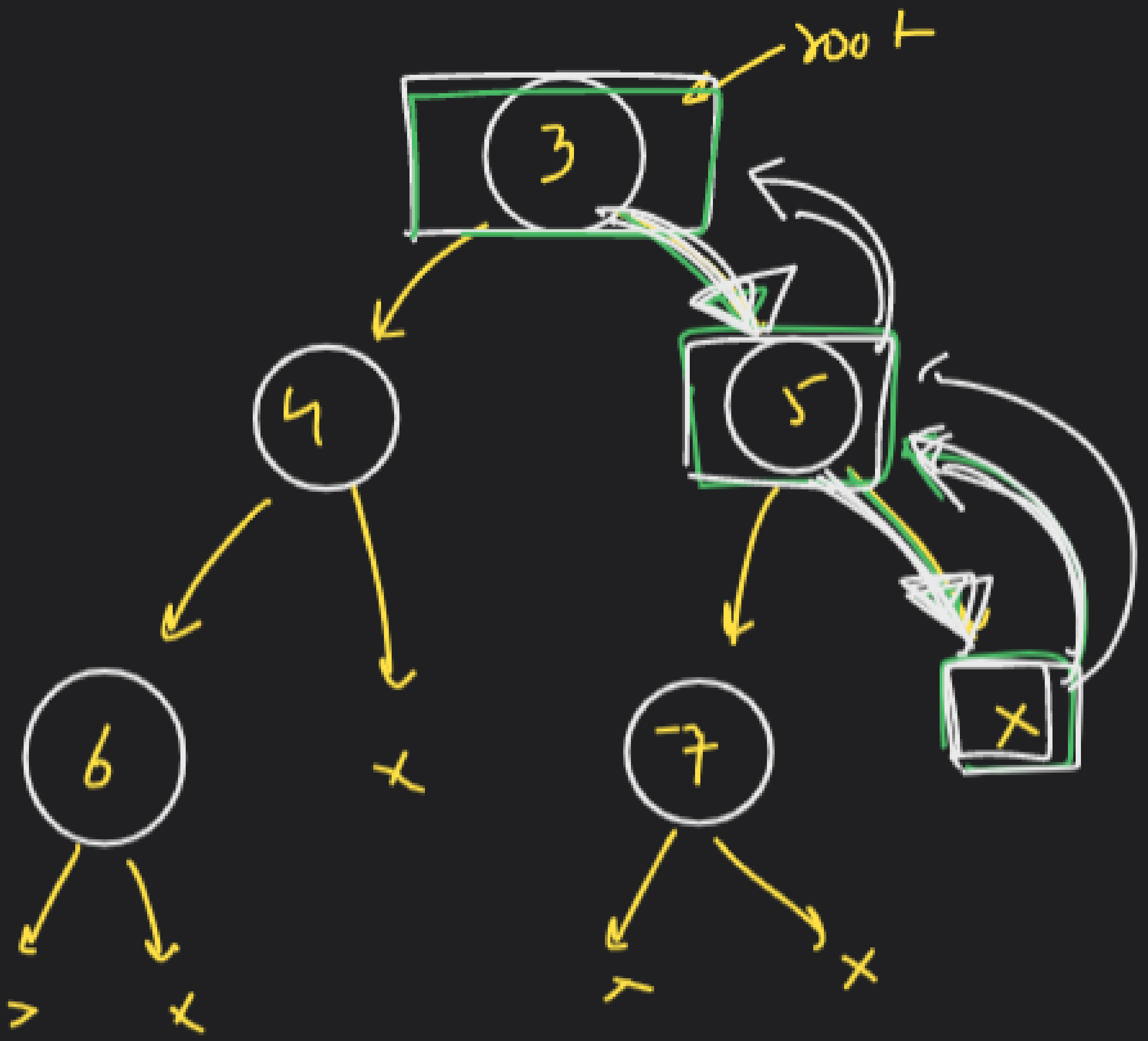
```
void solve (Node* root)
{
    if (root == NULL)
        return;

    solve (root -> left);
    solve (root -> right);
}
```

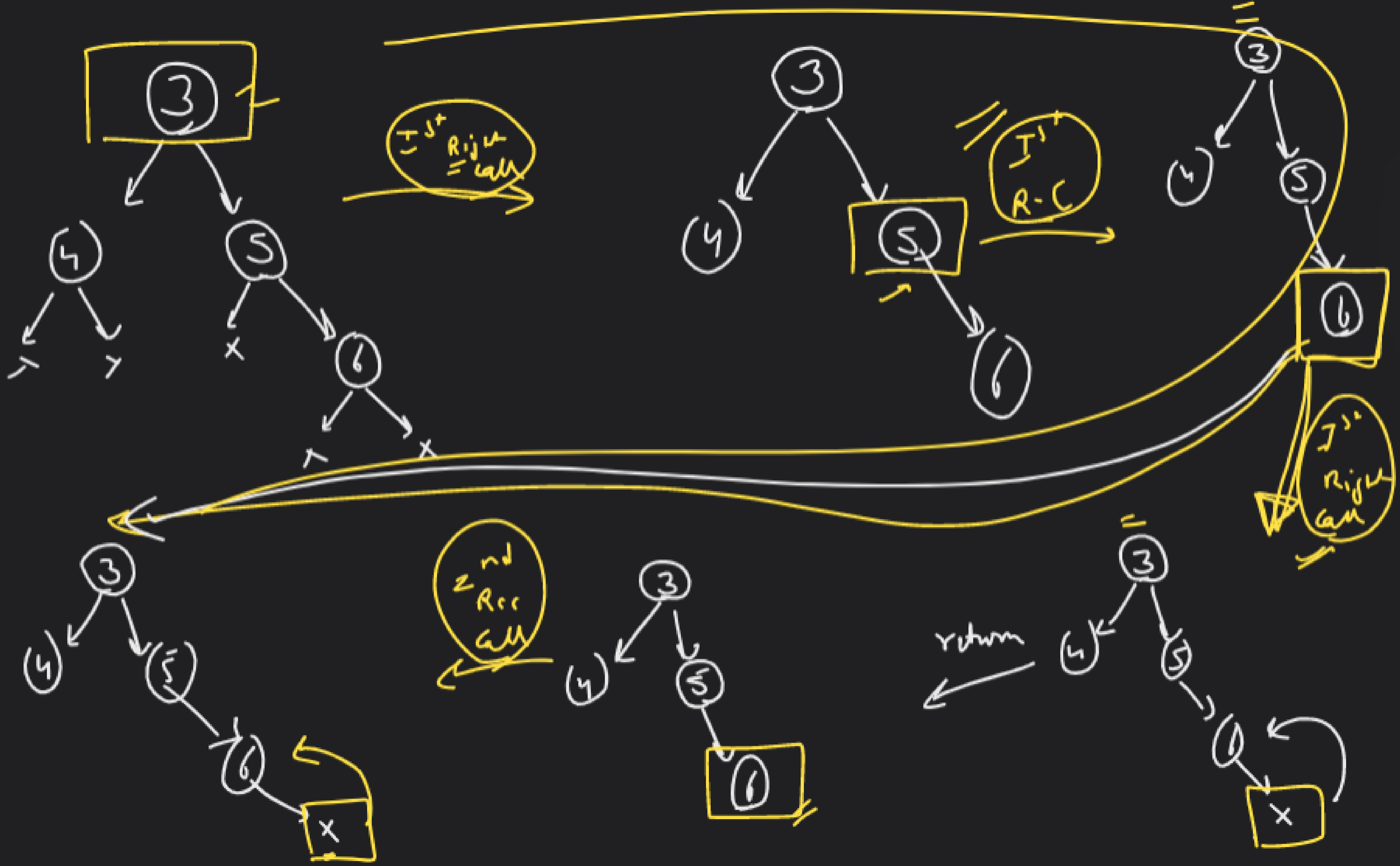


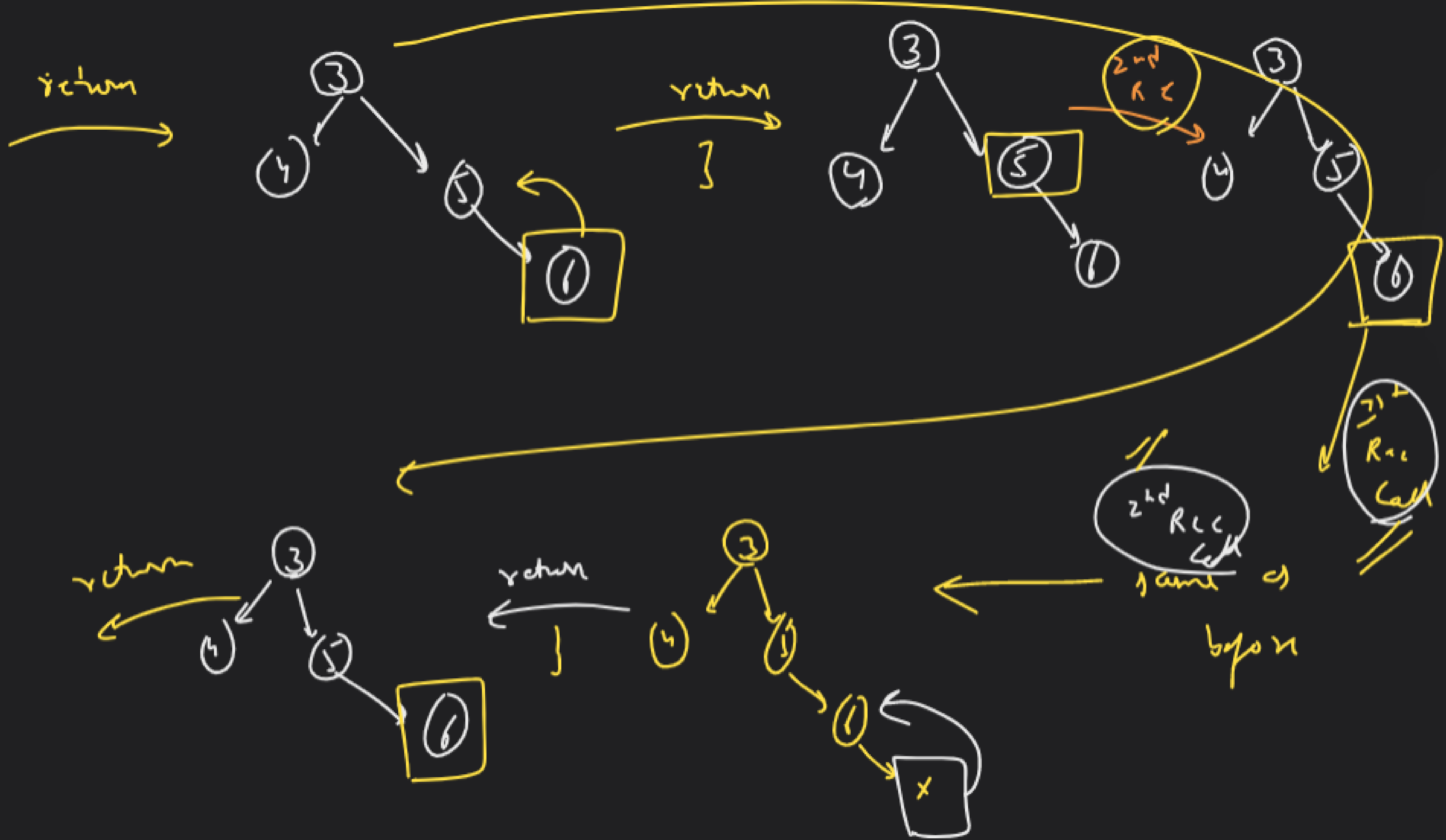
```
void   solve ( Node * root)
 {
     if  ( root == NULL )  → T
                           → F
          return ;

  ✓ solve ( root -> right )

  ✓ solve ( root -> left )

 }
```

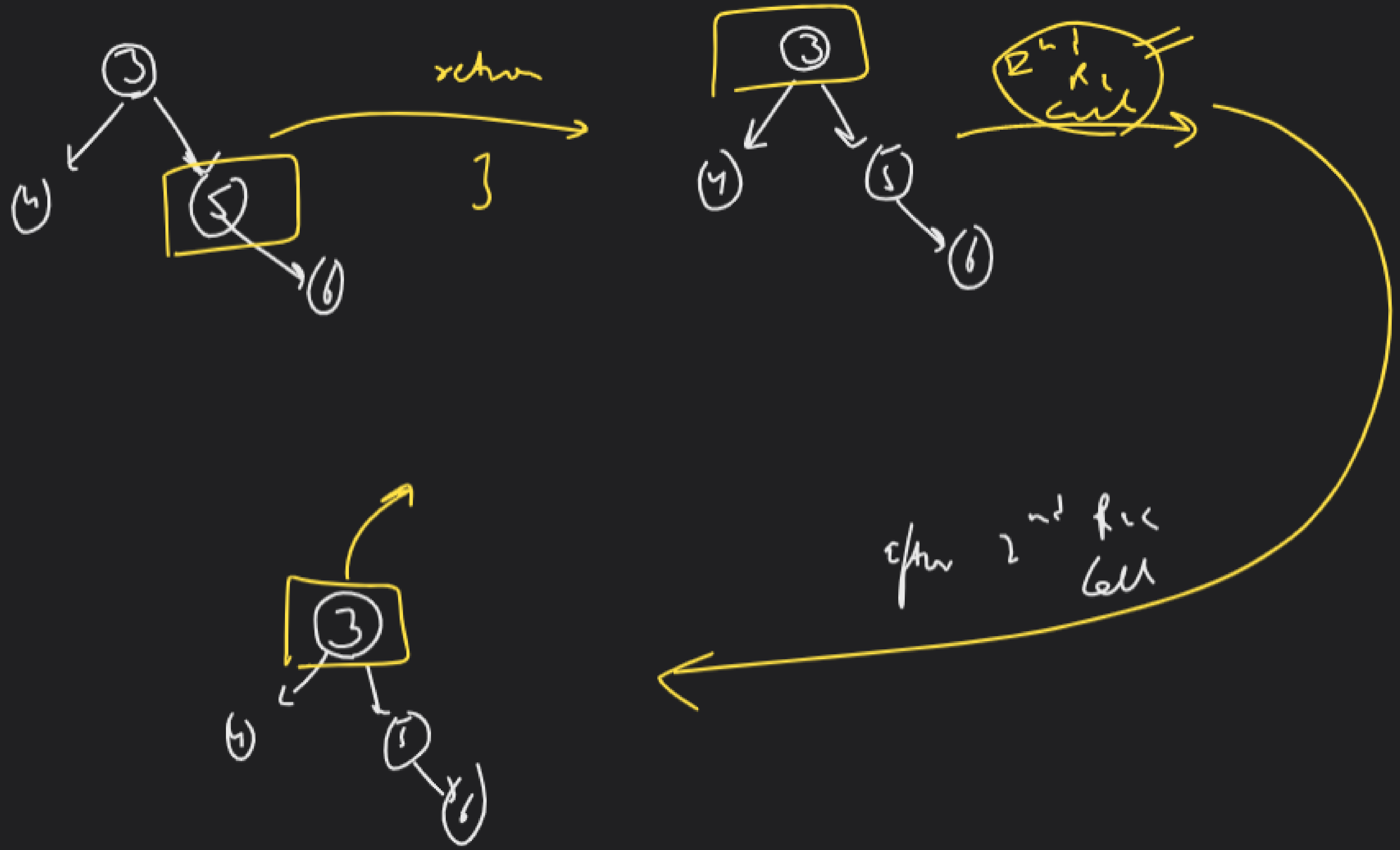


```
void  solve (Node* root)
{
      if (root == NULL)        T
                               F
          return;

 5R   solu (rot -> right);

 5L   solu (root -> right)

}
```




3
4
5
6
x
1st Right call
1st R-C
1st Right call
2nd Rec call
return


return
3
4
5
6
return
}
3
4
5
6
2nd
R C
3
4
5
6
1st
Rec
Call
2nd
Rec
Call
same as
before
3
4
5
6
x
return
}
3
4
5
6
return


3
4
5
6
return
3
3
4
5
6
2nd fn call
after 2nd fn call
3
4
5
6

Curr Node processing/operation

left part

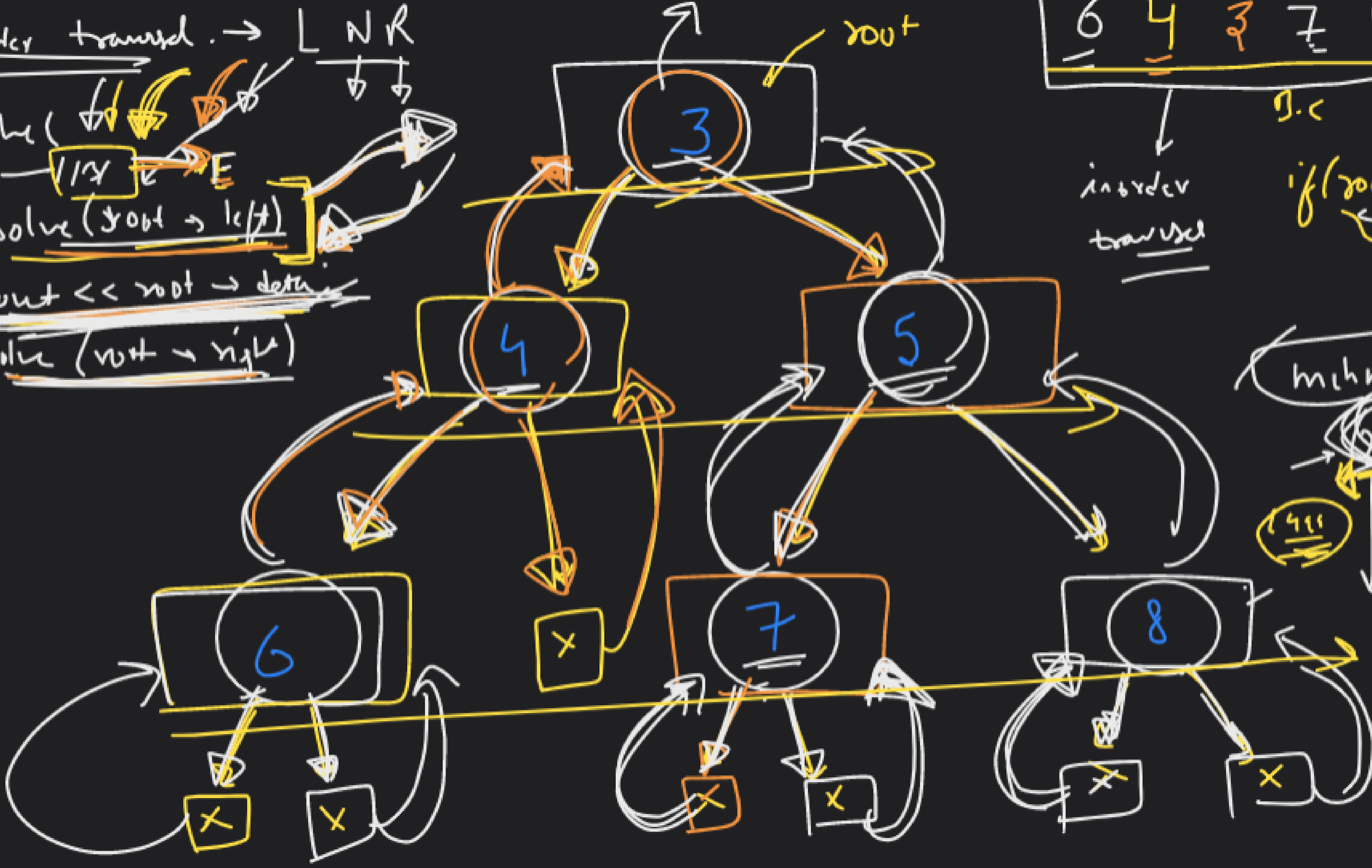
Right part

Traversals

- Inorder → LNR
- PreOrder → NLR
- PostOrder → LRN

Inorder traversal → L N R

```
solve( )
{
    // X  E
    solve(root → left)
    cout << root → data;
    solve(root → right)
}
```



6 4 3 7 5 8

T.C

inorder traversal

if(root == Null)
return;

mihret

649

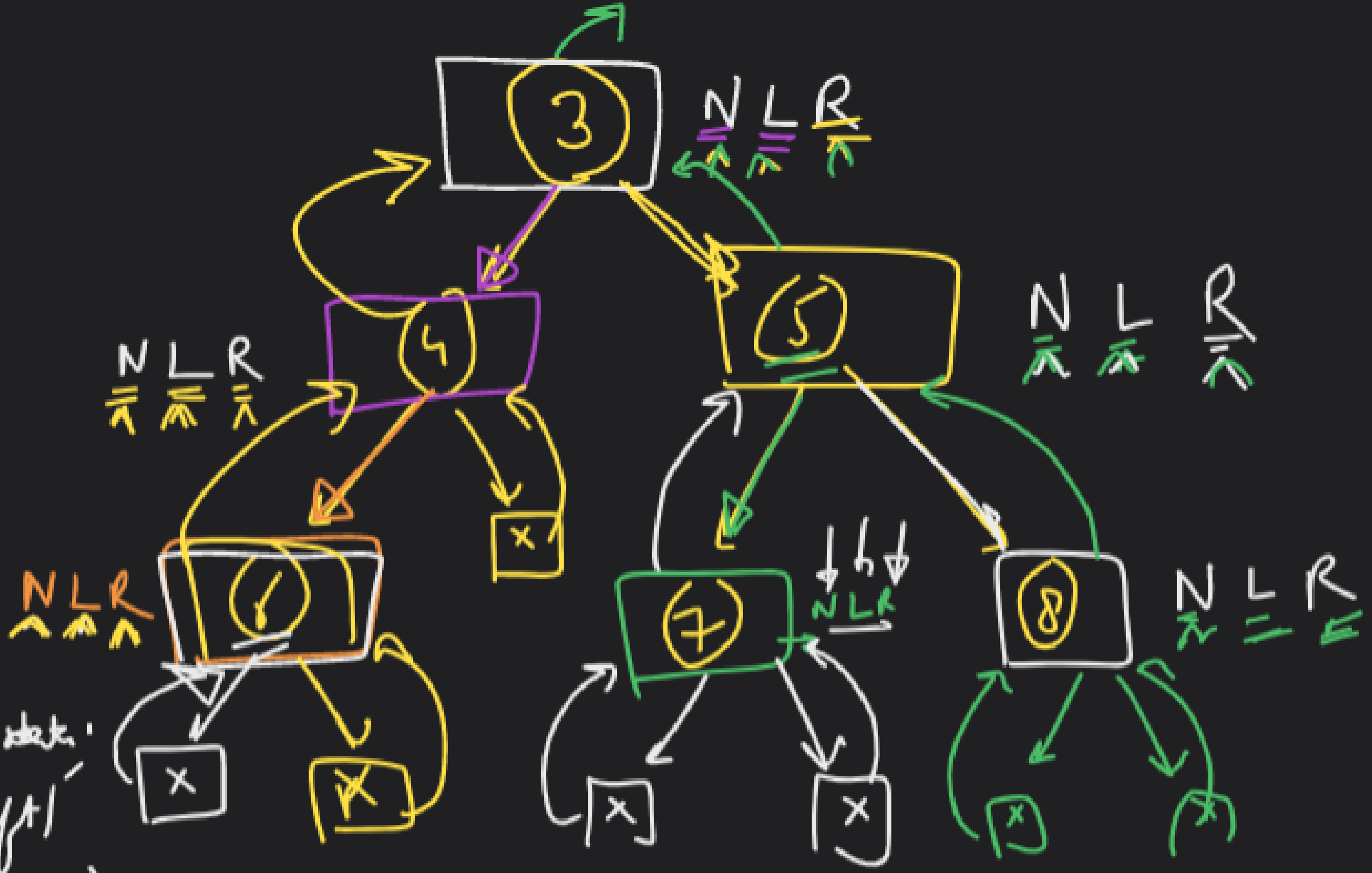
1000E

3 4 6 -1 -1 -1 5 7 -1 -1 8 -1 -1

PreOrder

NLR

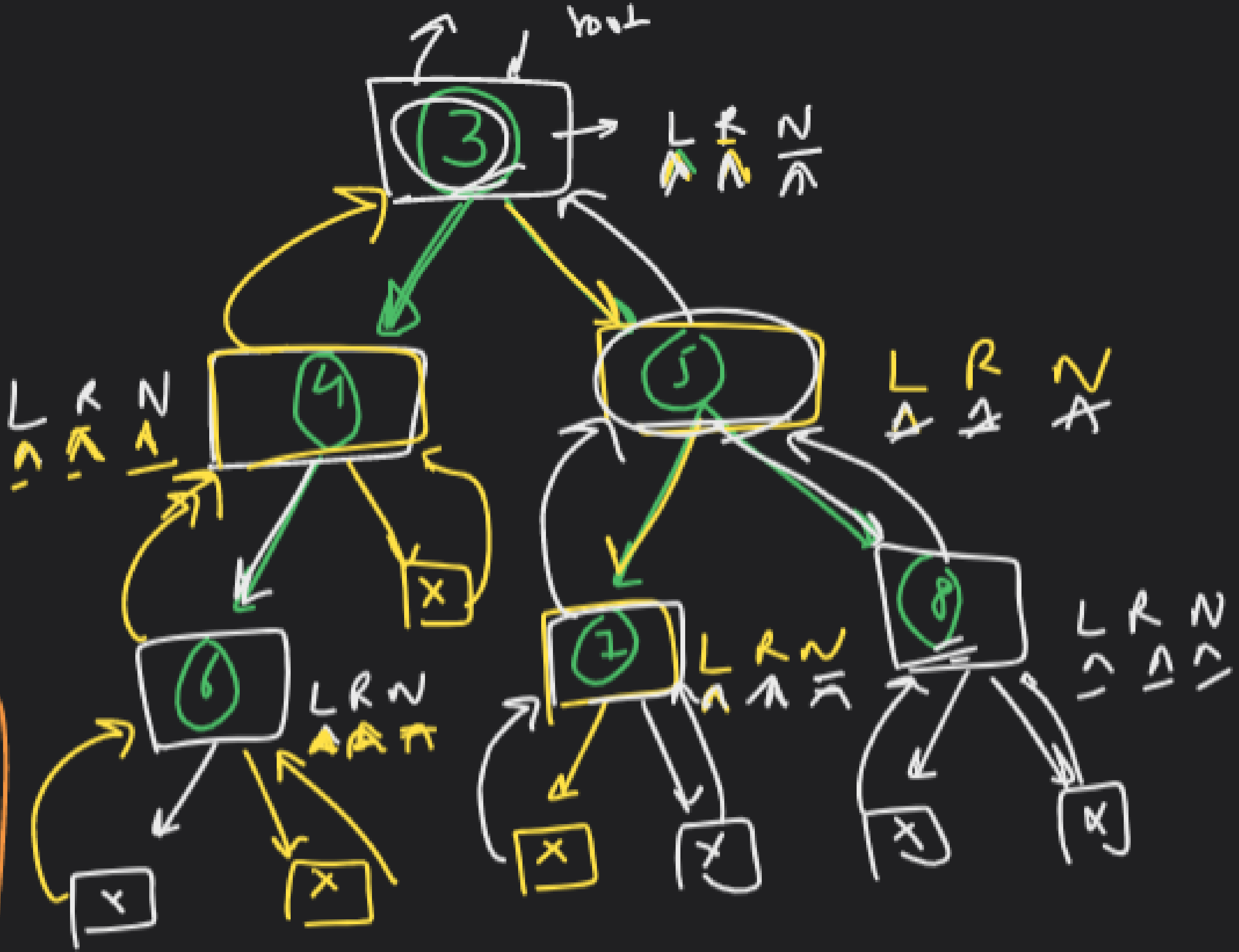
```
solve ( root)
{
    //B.C
    ✓cout << root -> data;
    ✓solve(root -> left);
    ✓solve (root -> right);
}
```



3 4 6 5 7 8 → preOrder

Post Order

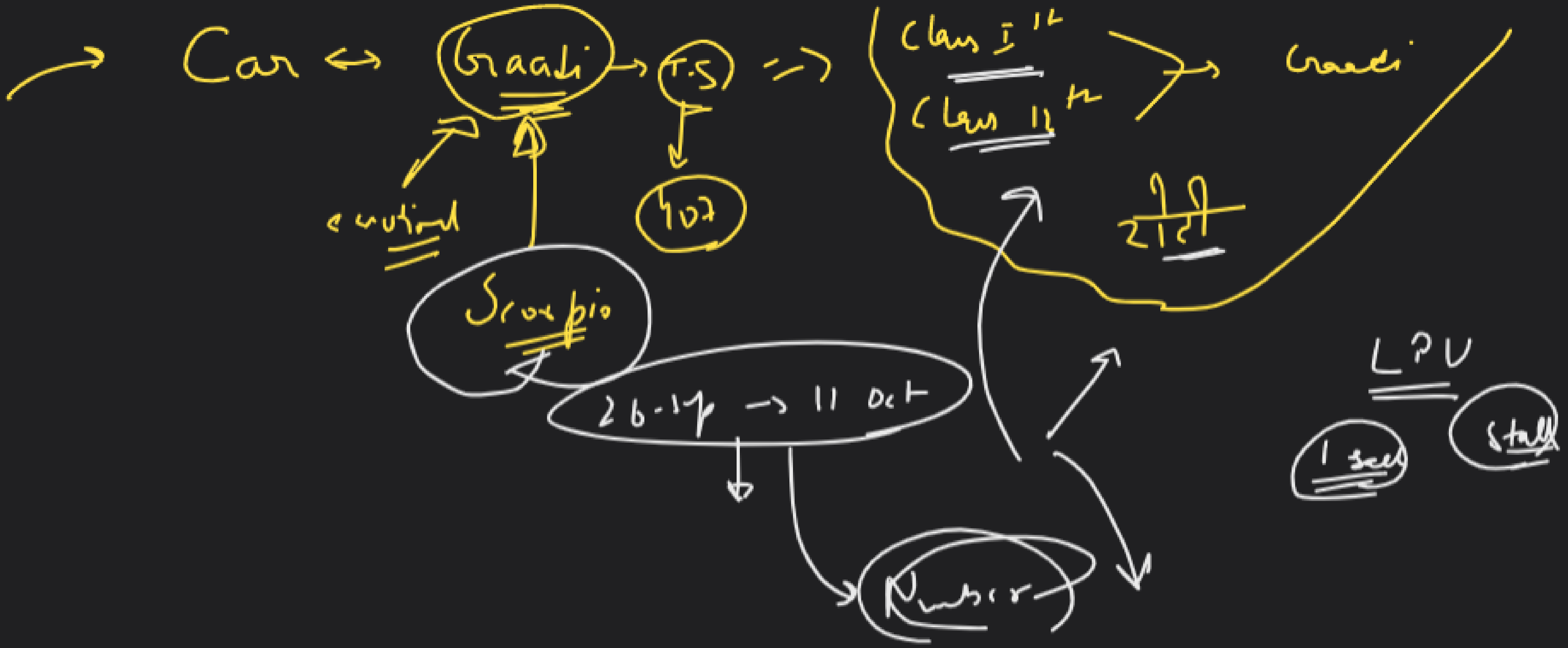
```
postOrder (root)
}
        //B-C

    // → LRN

L→ postOrder(root->left)
R→ postOrder(root->right);
N→ cout << root->data;
}
```

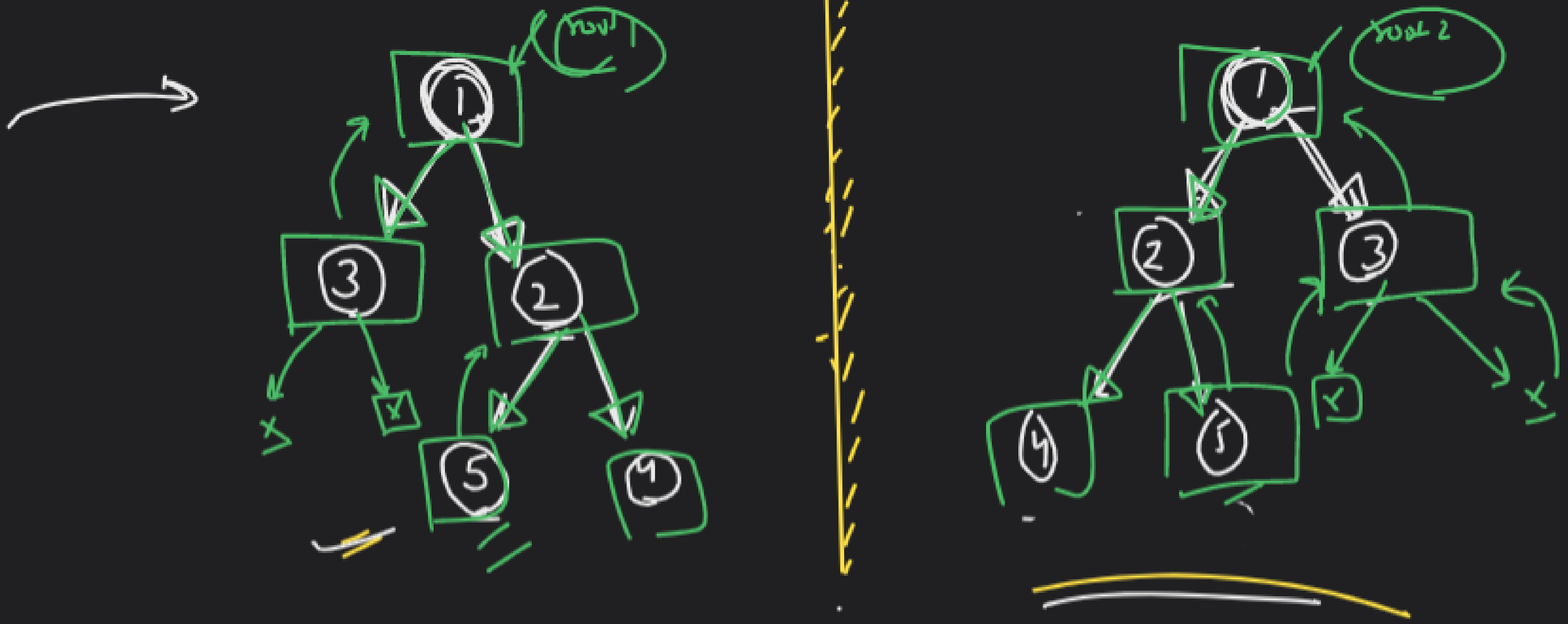


postOrder -> 6 4 7 8 5 3

2 min

Break

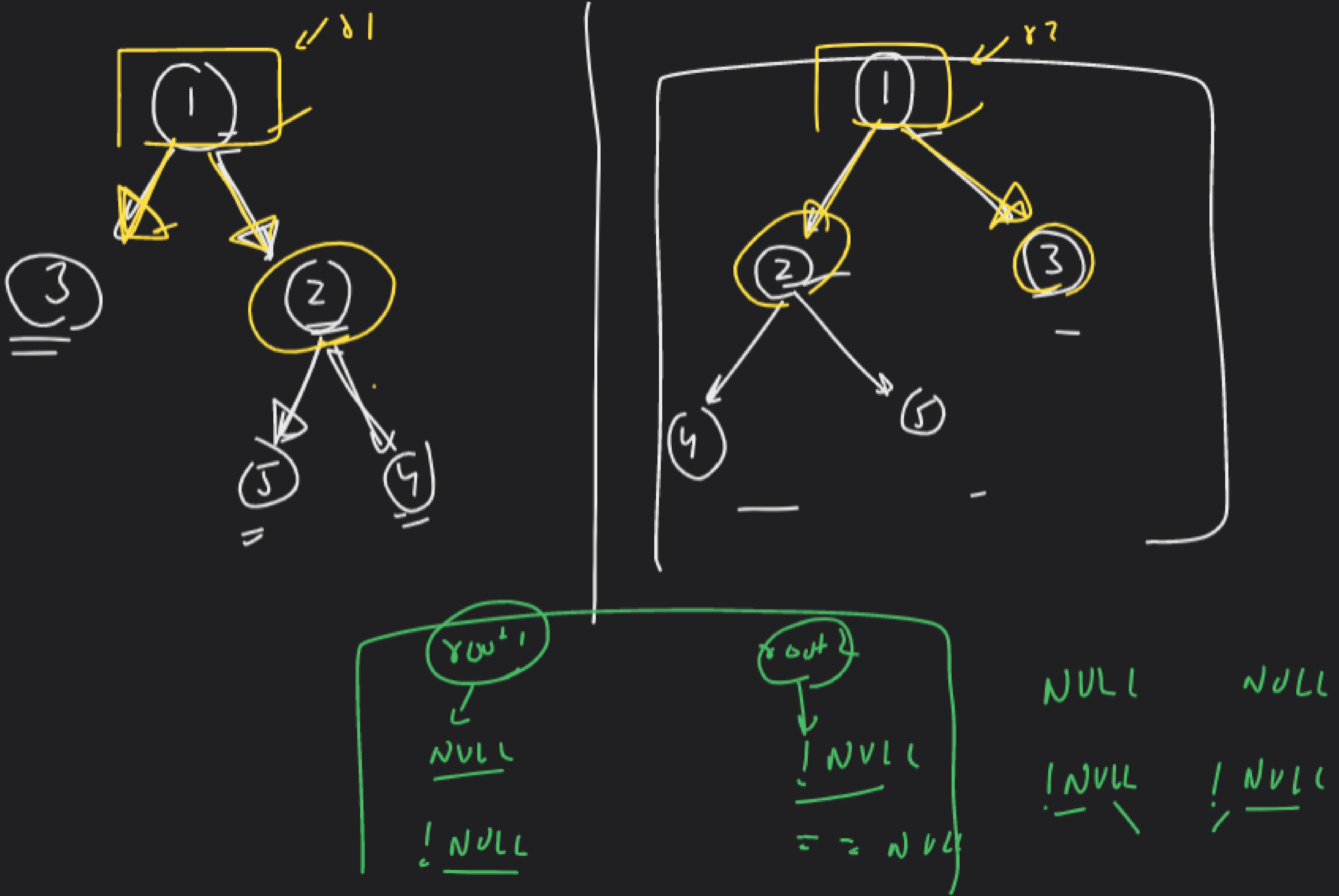

Car ↔ Gaadi → T.S → Class Ith / Class IIth → Gaadi
4v7
emotional
Scorpio
रोटी
26-Sep → 11 Oct
Number
LPU
1 seat
stall



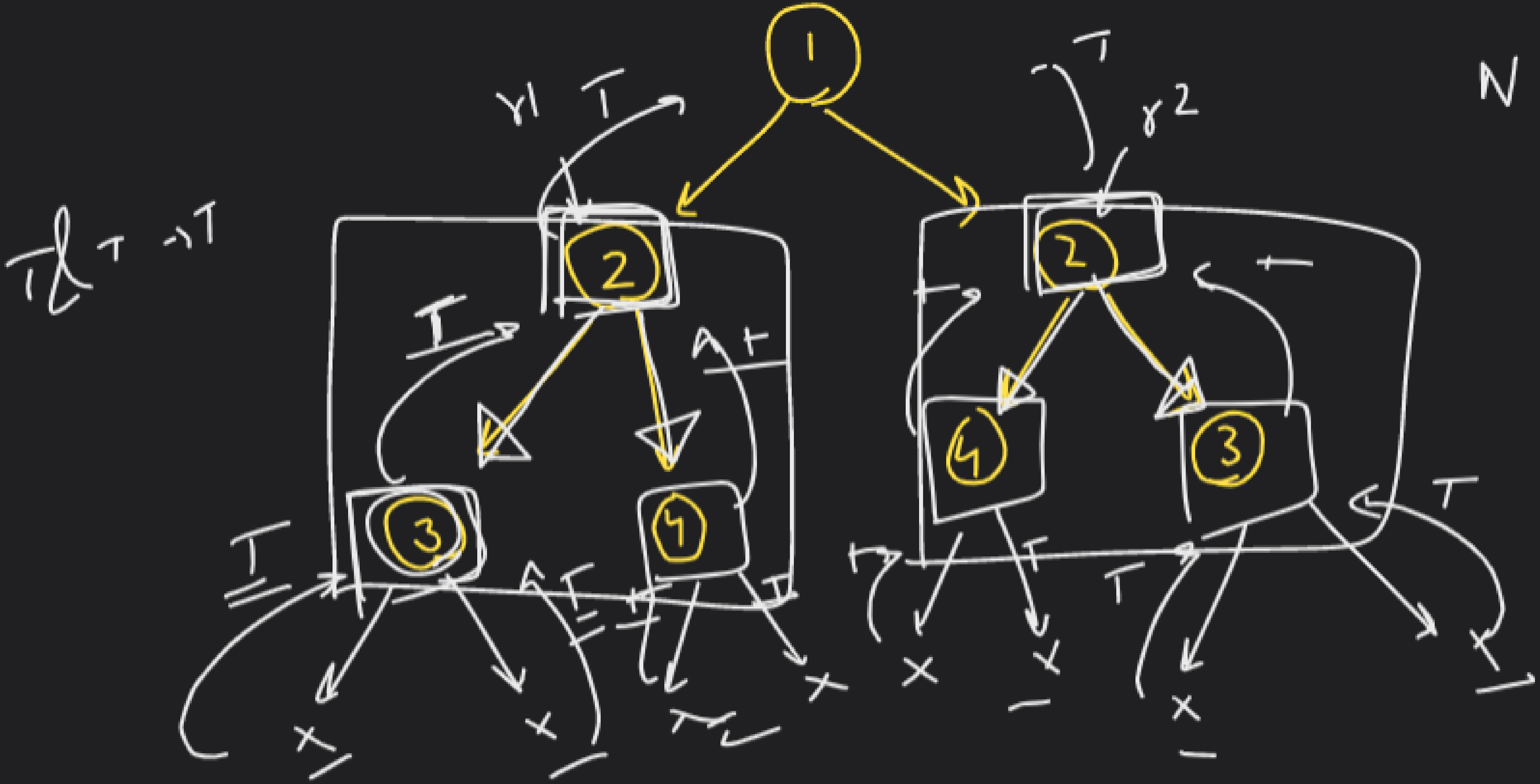
Check if they are mirror tree or not

cp1 → [tree1 → left , tree2 → right]

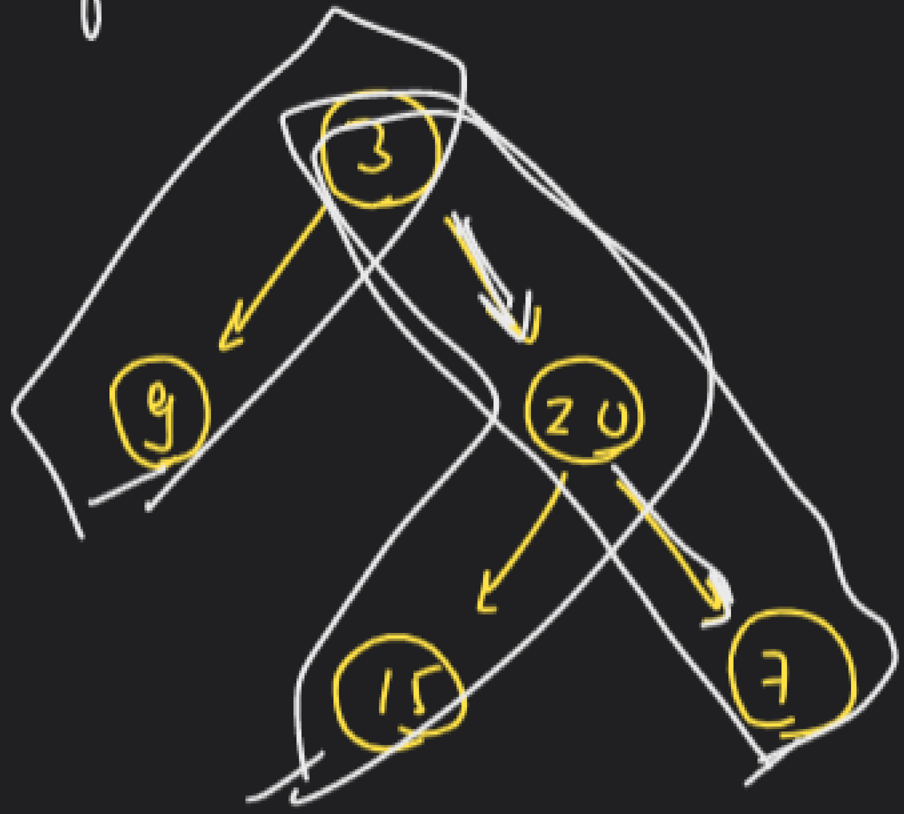
cp2 → [tree1 → right , tree2 → left]



r1
1
3
2
5
4
r2
1
2
3
4
5
root1
NULL
!NULL
root2
!NULL
== NULL
NULL
NULL
!NULL
!NULL


1
2
3
4
2
4
3
r1
r2
T
N
I
I
T
T
T

→ Height of tree



Max^m depth:

is the no. of nodes along the longest path from root down to farthest leaf node

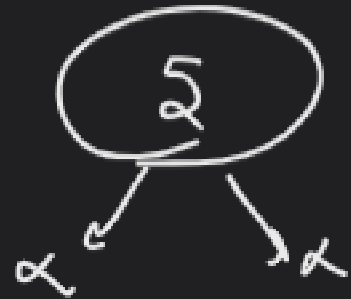
3

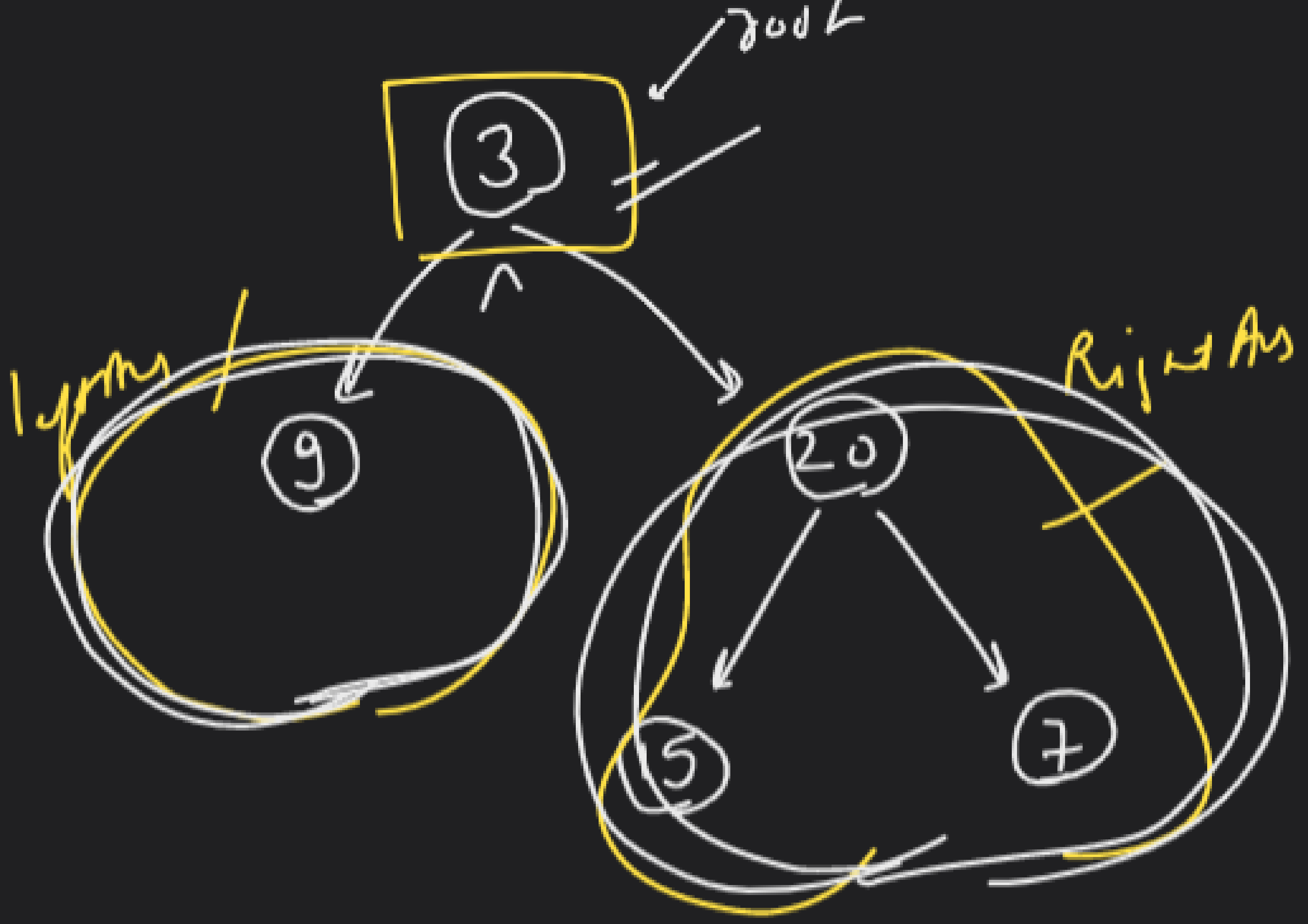
3 → 20 → 15

3 → 20 => 7

3

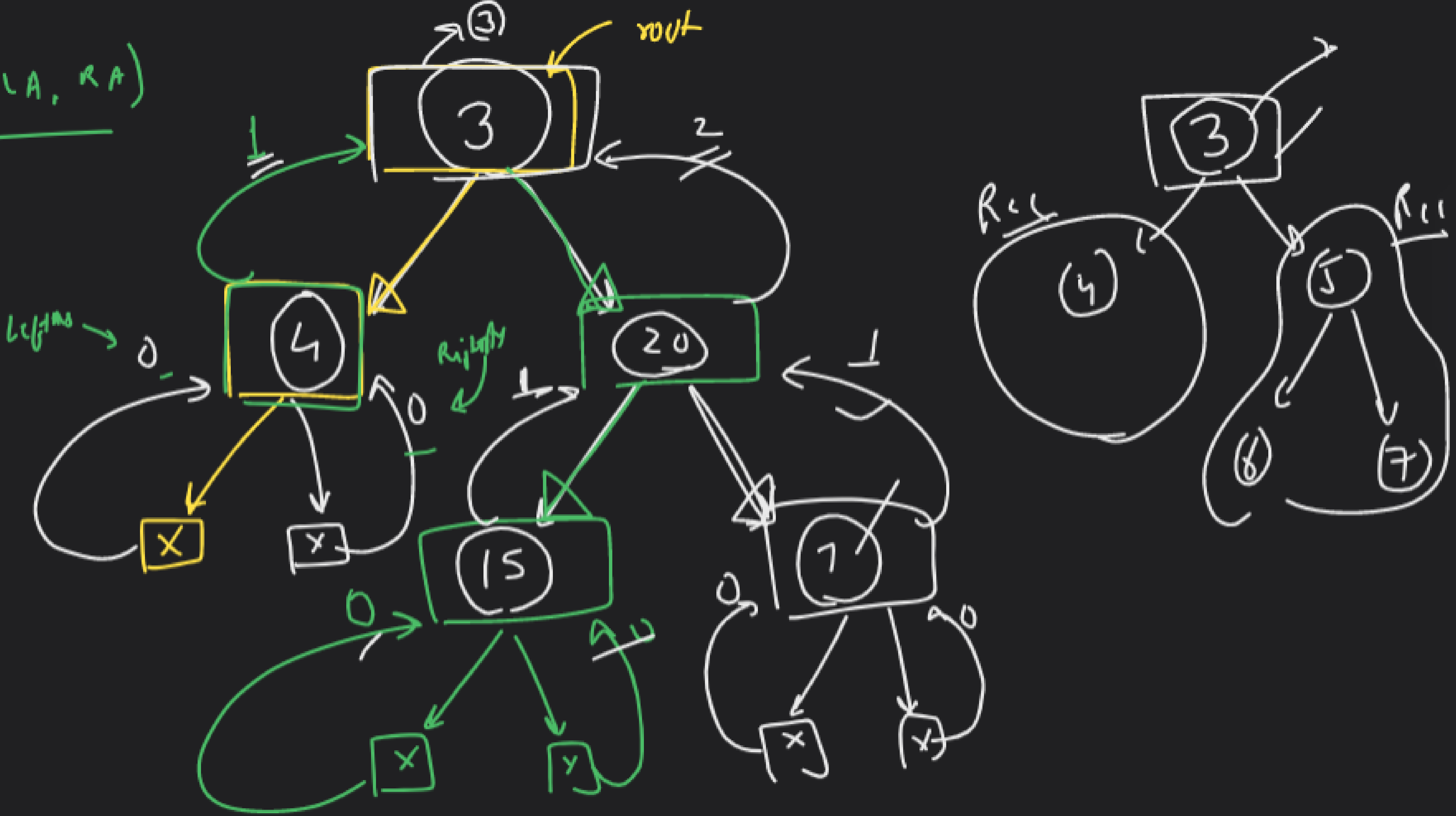
ans = 3



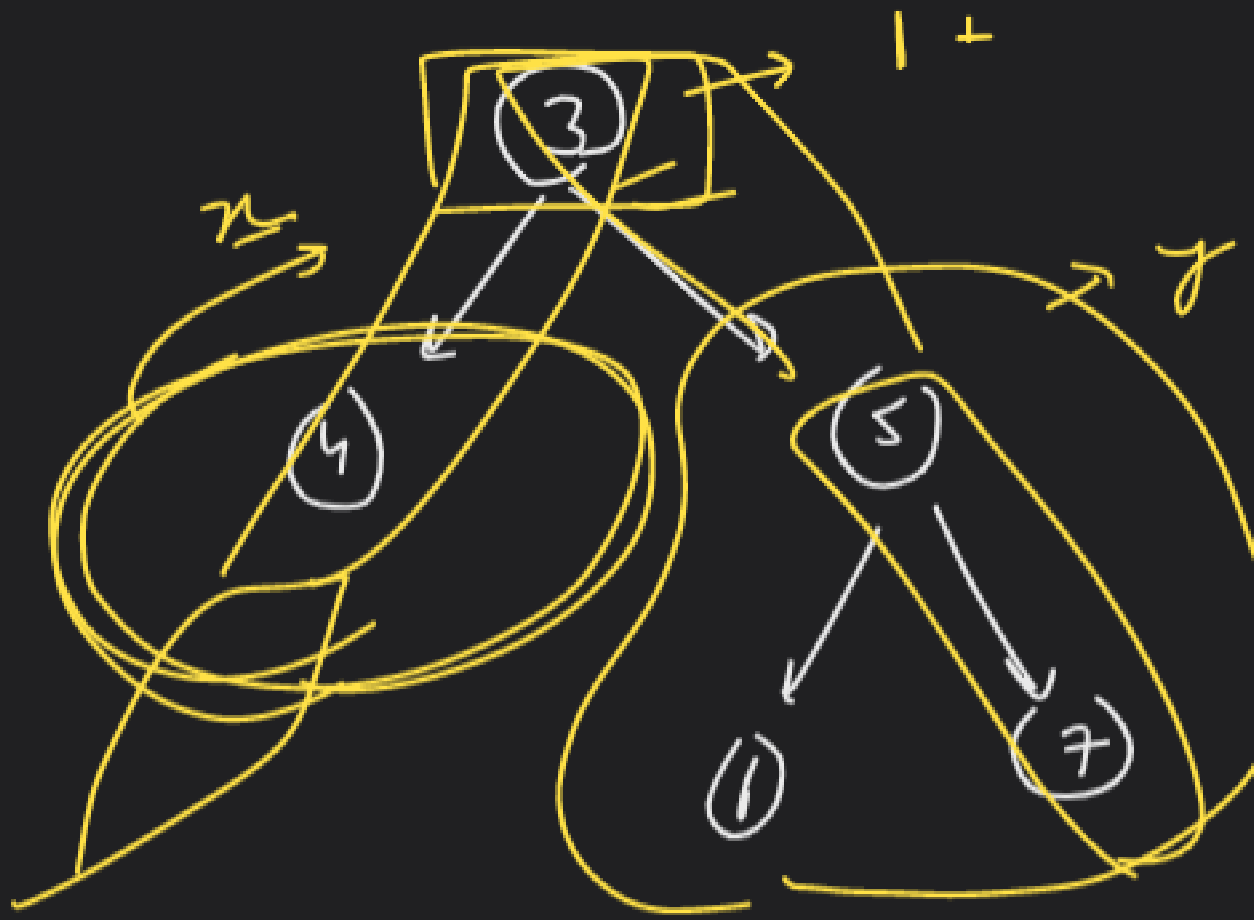


return 1+ max (leftAns, rightAns);

$1+\max(LA, RA)$



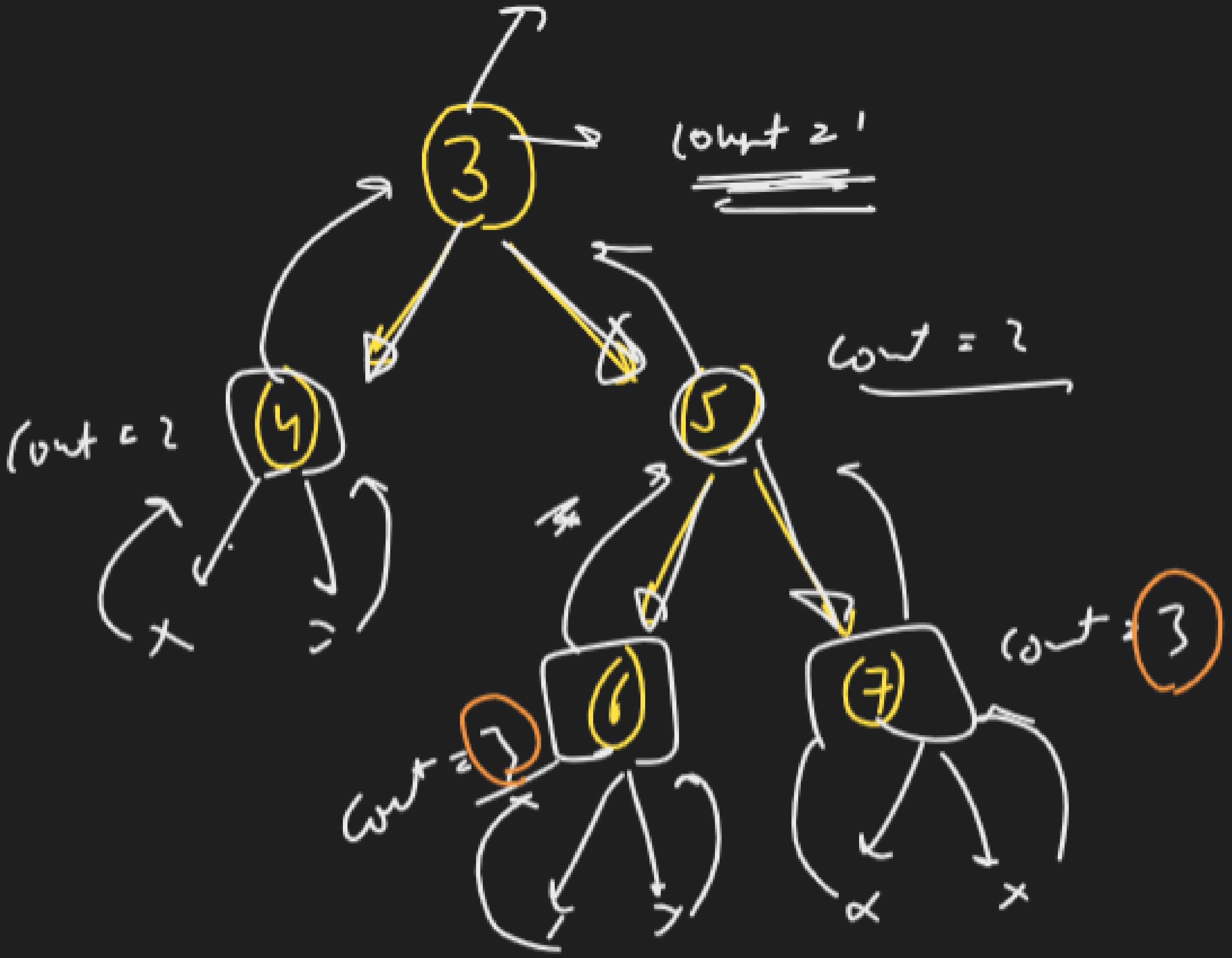
3
root
3
1
2
4
20
Leftms
0
Rightly
0
1
1
X
X
15
7
0
0
0
0
X
Y
X
X
Rcc
Rcc
3
4
5
6
7



max(x, y)

+

1


3
count = 1
4
count = 2
x
5
count = 2
6
count = 3
7
count = 3
x
x

HomeWork:-

→ Search a target in a Binary tree

→ Diameter of tree

→ Tree is Balanced or not

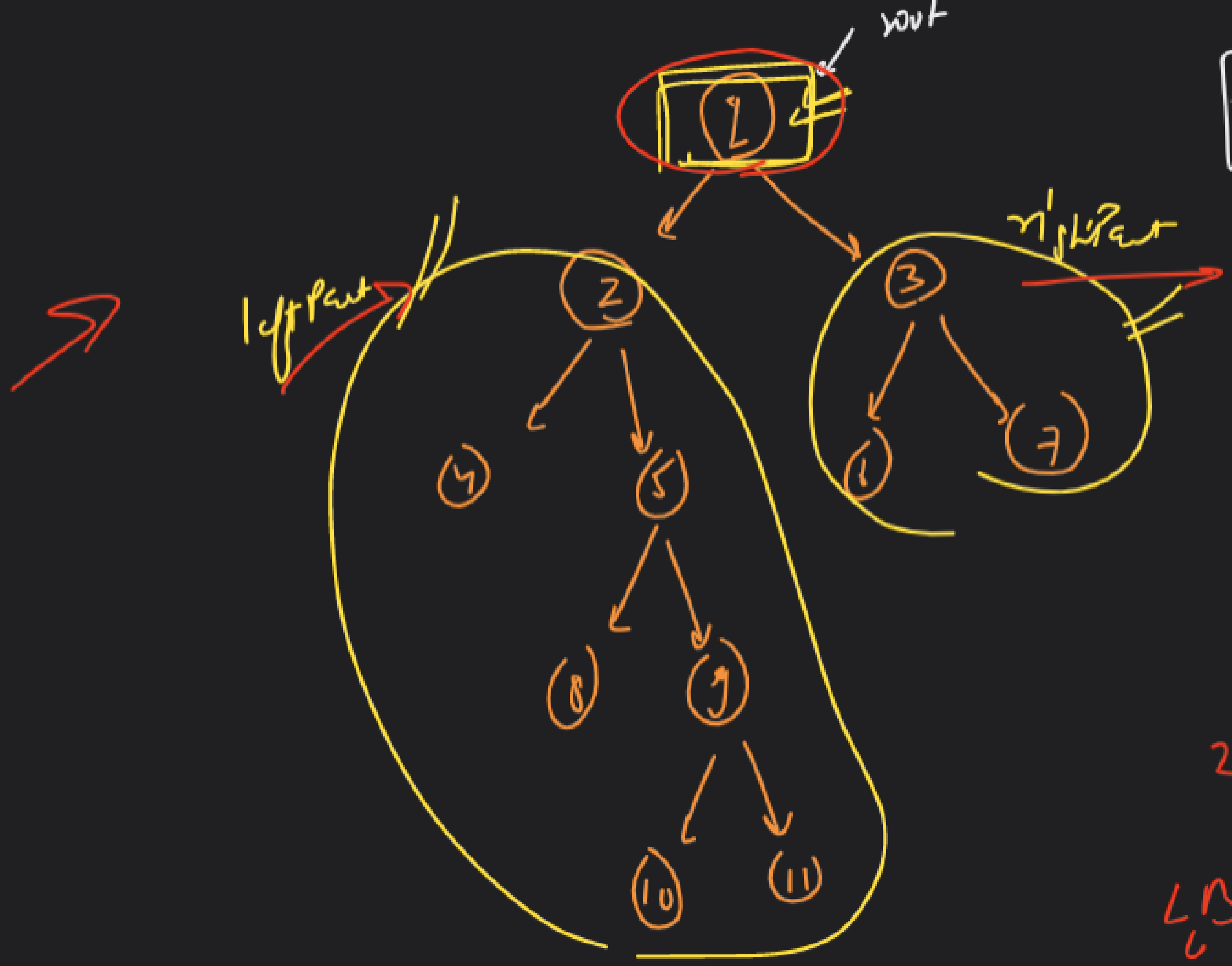
→ convert a tree into its Mirror Tree

→ Check if Binary tree is Sum tree or not

→ print all root to leaf path

→ print LCA of 2 nodes

→ print longest root → leaf path



target = 9

1 GB file

~~2 ...~~ 2 size gap

LB
index

Rec

file banalen

DSA → 9-10:30

for next days

T → 4-6 → 75%

F → 9-11

4-6 pm

Sat/Sun

Merge Sort

&

Quick Sort

9-11 pm

Sat/Sun