General Updates

Monday – Subsequences.

Revision + vacation

2nd July → Intermediate Contest

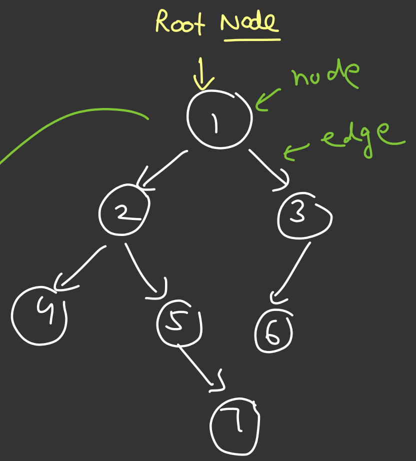
4th July → Discussion class. afterwards

6th July ?

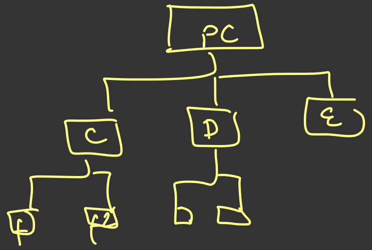
13th July – Advanced Batch ? Prateek Narang

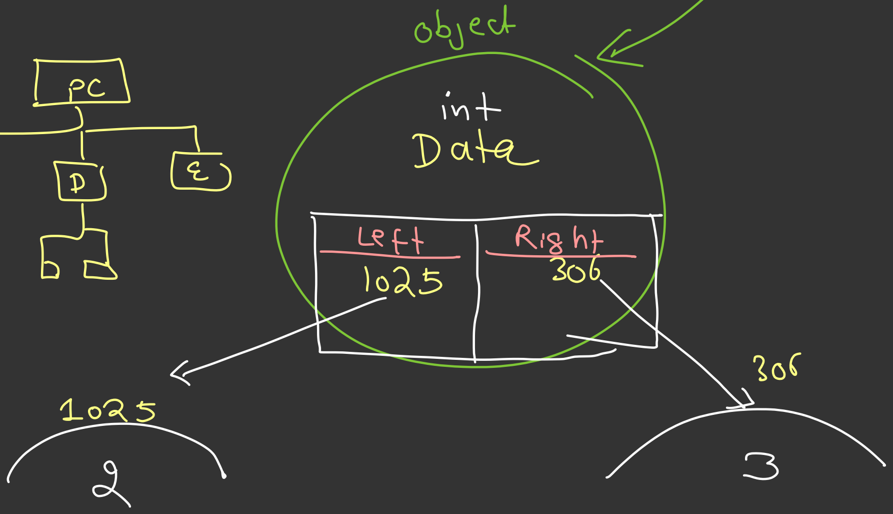
Binary Trees

- → each node can have atmost 2 children
- → hierarchy



user defined class
↑
Node Structure





1 → parent

direct

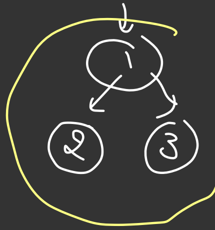
2, 3 → children of 1

7 → Desencand of 2

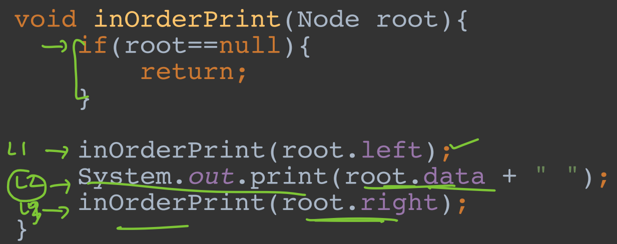
root

```
Class Node {
        int data,
        Node left ,
         Node right ,

      Node (int d) { data = d,
                left = right =null , }

}
```

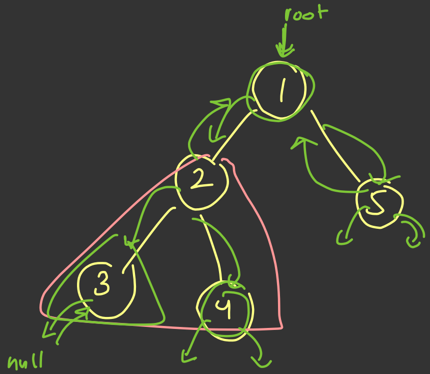
```
Node root = new Node(1)

   root. left = new Node(2)
   root right = new Node(3)
```

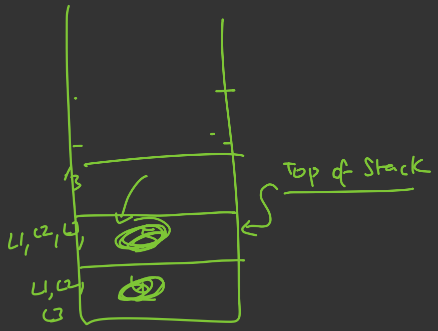


```java
Node buildTree(){
    int data = sc.nextInt();
    if(data==-1){
        return null;
    }
    Node temp = new Node(data);
    temp.left = buildTree();
    temp.right = buildTree();
    return temp;
}
```



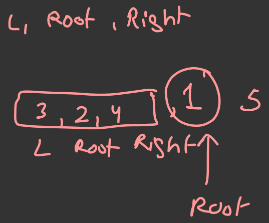
```
void inOrderPrint(Node root){
    if(root==null){
        return;
    }

    inOrderPrint(root.left);
    System.out.print(root.data + " ");
    inOrderPrint(root.right);
}
```





3, 2, 4, 1, 5 →

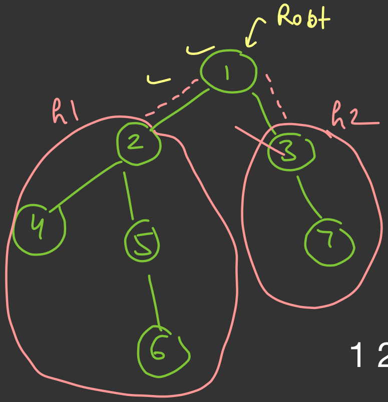
Preorder



Tree related problems

Height of Tree

1, 2, 4,



Node (4) → Confirm from Sample I/O.

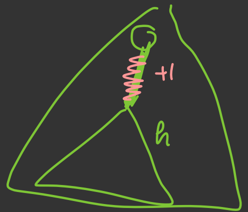
max(h1, h2) + 1

1 2 4 -1 -1 5 -1 6 -1 -1 3 -1 7 -1 -1

null ← 0

```
int   height ( root ) {
        if (root == NULL) { return 0, }

        h1 = height(root left)
        h2 = height(root right)

        return  max (h1, h2) + 1 ,
                └→ height at root Node
```

Postorder
- Left
- Right
- Root

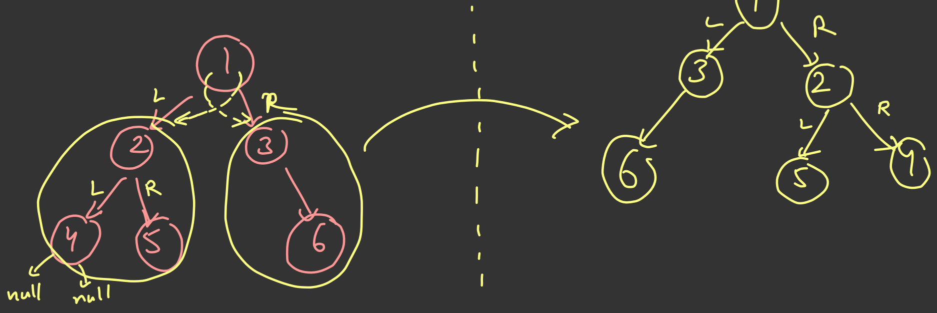


3

PROBLEMS

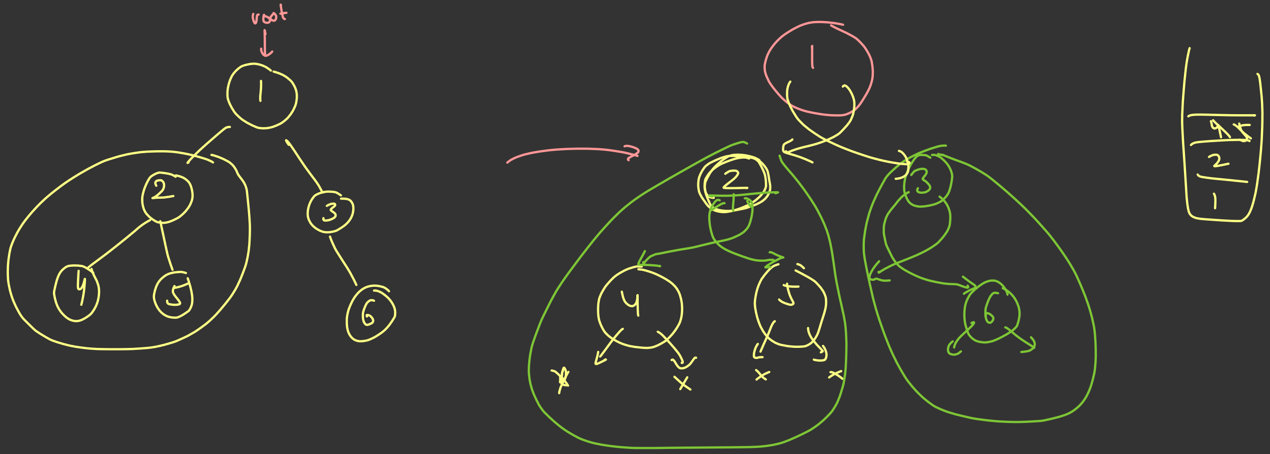
Tree → Invert / Mirror

①



```
void   mirror ( Node  root) {

    if (root == null) { return ;   };
             root
  → mirror(left)
  → mirror (root right);
  → swap(root.left , root.right);

}
```

Post order

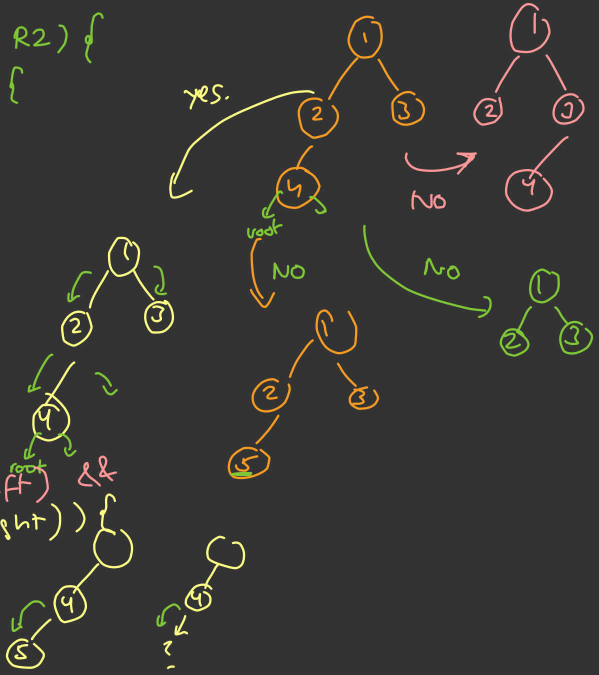

root
1
2
3
4
5
6
1
2
3
4
5
6
4 5
2
1

Q [Given Two Trees, check if they are identical]

Base case

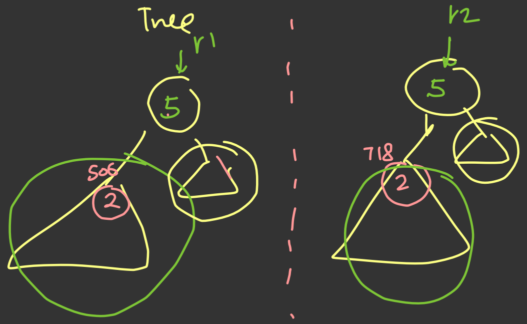
Structure

```
bool    isSame ( Node R1 ,    Node R2) {
    if( r1 = =null   &&   r2 = =null) {
              return true;
          }
    if  (r1 = =null   ||   r2 = =null){
              return false
       }

if ( r1.data  = =  r2.data  &&
       isIdentical (r1.left , r2.left)  &&
       is Identical( r1.right , r2.right)) {

       return true;

   }
```



return false.

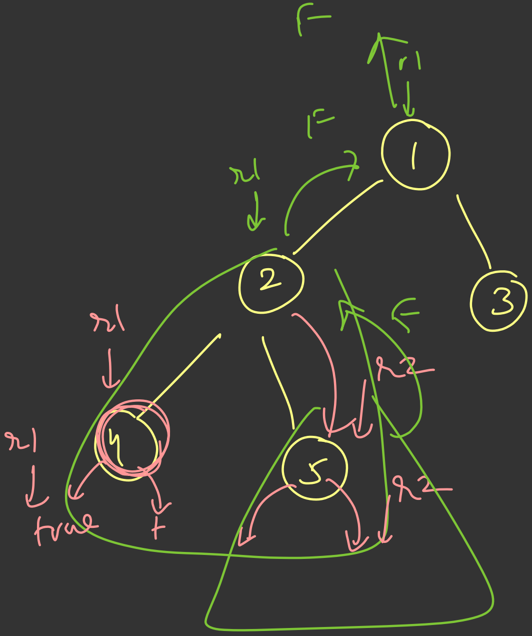
}

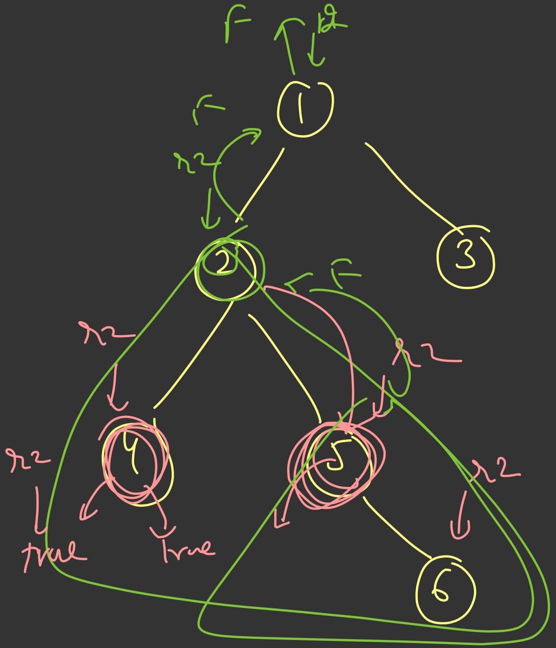


Data should be same & Structure

Rec Defn

```
if (r1.data == r2.data &&
        isIdentical(r1.left, r2.left) &&
        isIdentical(r1.right, r2.right)) {

        return true;

}
return false
```



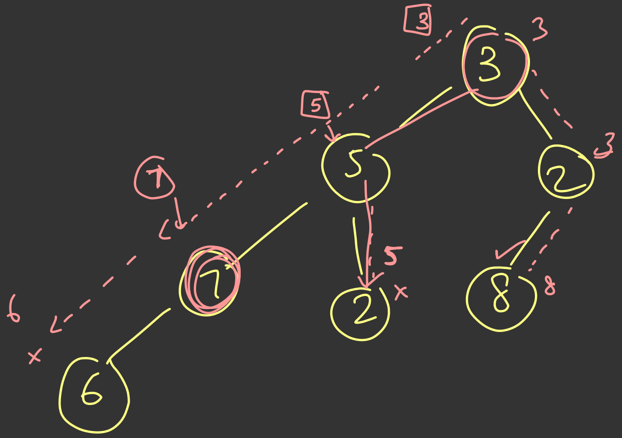


1 == 1 && ___ && _

2 == 2 && true &&

4 == 4 && t && t_

5 == 5 && t && f



Count No of Nodes

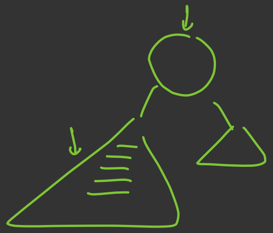
which ~~are~~ have

the

largest val in the

path — root Node

7 > 5, 3



f (root)

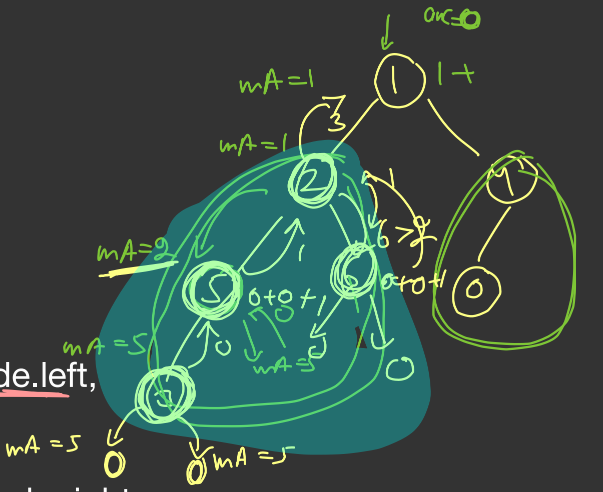
max So Far →

cnt = (f(root→left) + f(root.right)

+1 if c.N satisfies.

+0 if ———— ")

```
int getCountOfgreaterNodes ( node , arc,

      if(node == null) { return 0; }
----> int cnt = 0 -> local variable
[     ~~if( node.val > maxAncestor )~~
      ~~{ maxAncestor = node.val; count++; }~~   cnt++

Left  int leftCount = getCountOfGreaterNodes(node.left,
      maxAncestor, 0);

Right int rightCount = getCountOfGreaterNodes(node.right,
      maxAncestor, 0);

Root  return leftCount + rightCount + count; }
```



1 if node.val > mA

$\boxed{3} + \cancel{X} + LS + RS$

1 if 3 > 5
otherwise 0

$1 + 1 + \boxed{2>1}$

=

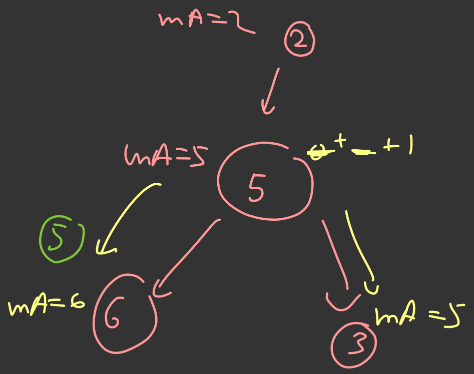
```
int    Calc Nodes(  Root , mA ) {
       if (root == NULL )     return 0,
       cnt = 0
       if( Root.val  > mA) {
            cnt = 1
            mA = Root.val
       }
       LC = calc Nodes (Root left , mA )
       RC = calc Nodes (Root Right , mA )

       return    LC + RC + cnt
}
```

local cnt ⇒

used here

total cnt

0 or 1



BST

Binary Search Tree

Adv. Batch