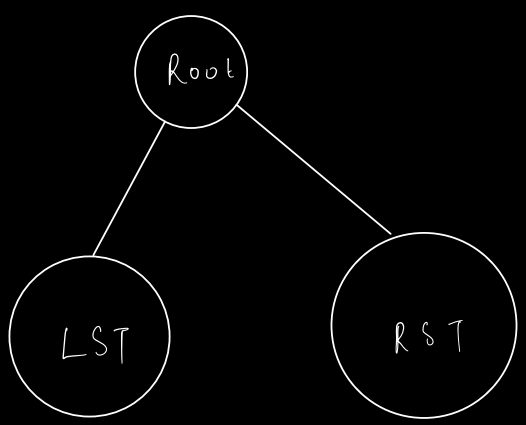
Today's Content :-

→ level order traversal

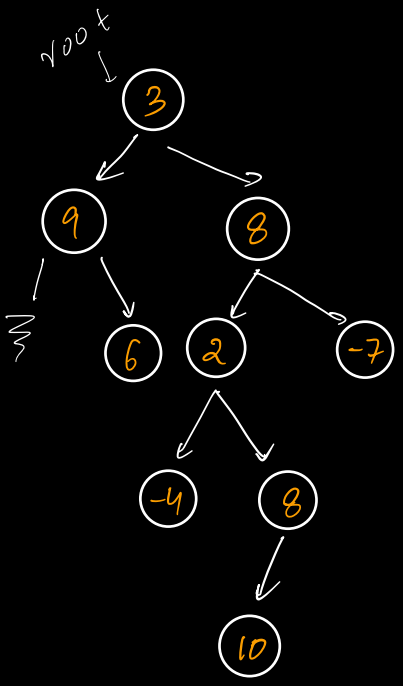
a) left view b) Right view

→ Construct Tree from Inorder & Preorder

→ Inorder Traversal Iterative



Level Order traversal :-



Expected Output :

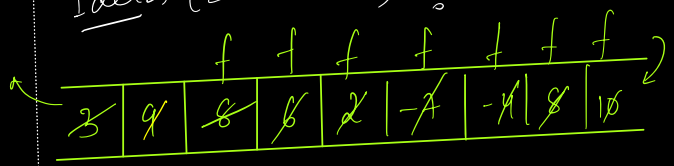
3 9 8 6 2 -7 -4 8 10

TC : O(N)

SC : ~ 10 Mins

Ideal (Recursion) → Subproblems X

Ideal (Iterative) ?



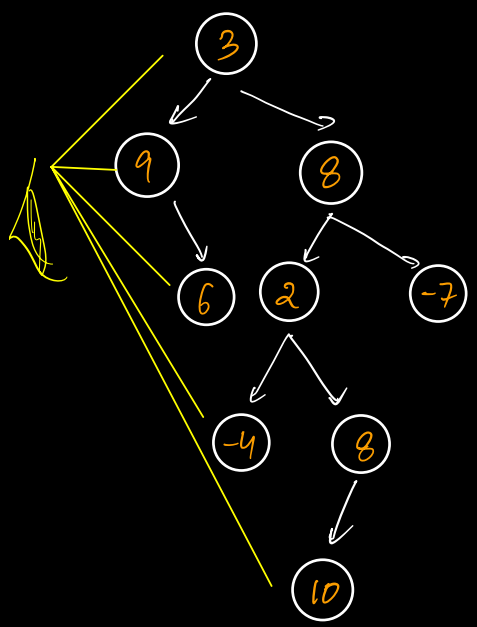
3 9 8 6 2 -7 -4 8 10

Pseudo Code

```
Queue < Node > q
q.insert(root)
while(q.size() > 0) {
    Node f = q.front()
    q.remove()
    print(f.data)

    if(f.left != NULL) {
        q.push(f.left)
    }
    if(f.right != NULL) {
        q.push(f.right)
    }
}
```

Level Order traversal 2



Expected Output

```
3 \n
9 8 \n
6 2 -7 \n
-4 8 \n
10 \n
```

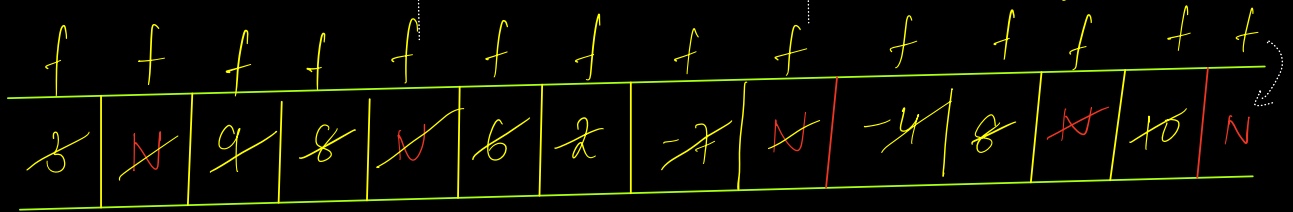
Left View: To view tree from left.

Obs: 1st node of every level

for every 1st element in level, prev = NULL

Edge Case: Root Node
↳ print at start

Node - reference




```
3 \n
9 8 \n
6 2 -7 \n
-4 8 \n
10
```

```
if (f == NULL) {
    print("\n")
    q.insert(NULL)
}
```

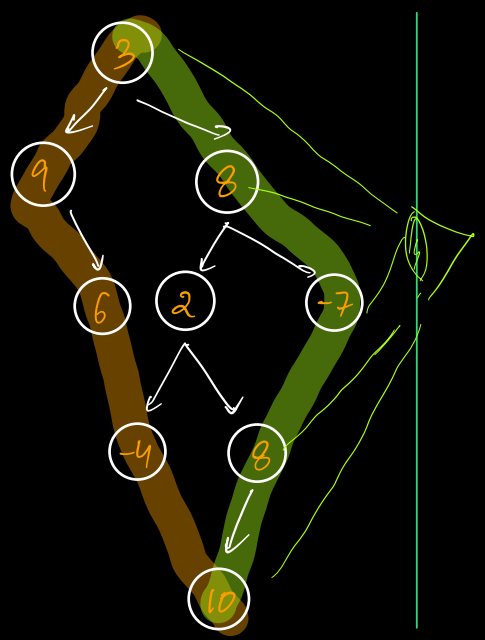
While (q.size() > 1)

```
Queue < Node > q
q.insert(root)
q.insert(NULL)
while(q.size() > 1) {
    Node f = q.front()
    q.remove()
    if(f == NULL) {
        print("\n")
        q.insert(NULL)
    }
    else {
        print(f.data)
        if(f.left != NULL) {
            q.push(f.left)
        }
        if(f.right != NULL) {
            q.push(f.right)
        }
    }
}
```

TC : O(N)

SC : O(Max Nodes we can have in level)

Level Order traversal 3 (Right to left)



3 \n

8 9 \n

-7 2 6 \n

8 -4 \n

10 \n

While inserting child Nodes

- push (right child)
- push (left child)

{LV + RV = Boundary view | Circumference | Perimeter.}

Right view: Keep eye on right side & view.

3 8 -7 8 10

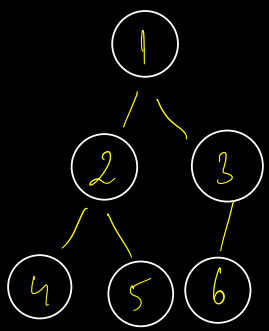
Idea: first node of every level in level order traversal from R-L.

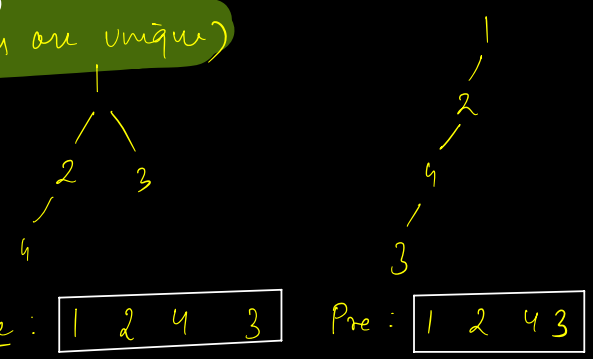
Root ✓

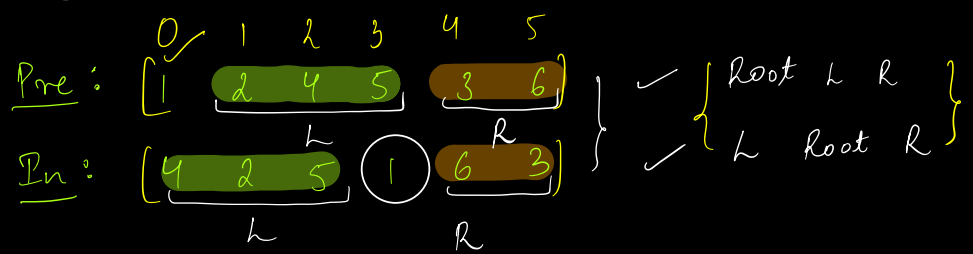
{& Min brealk}

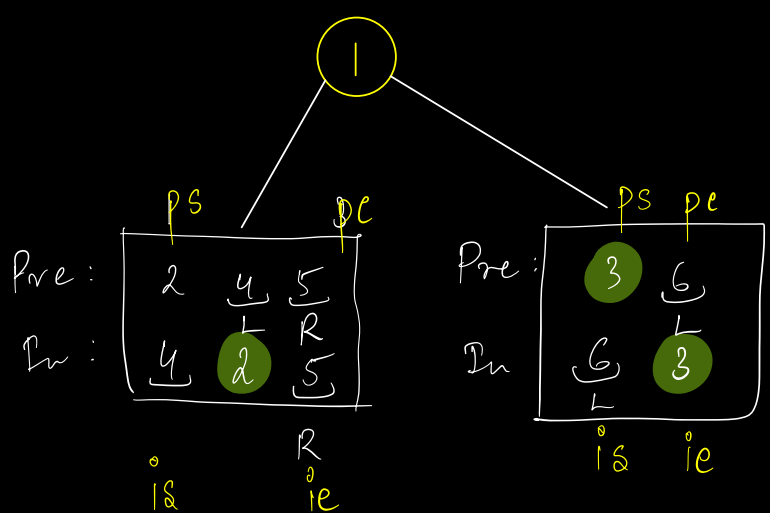
Q2) Given inorder & Preorder of a BT. Construct the tree.

(elements are unique)



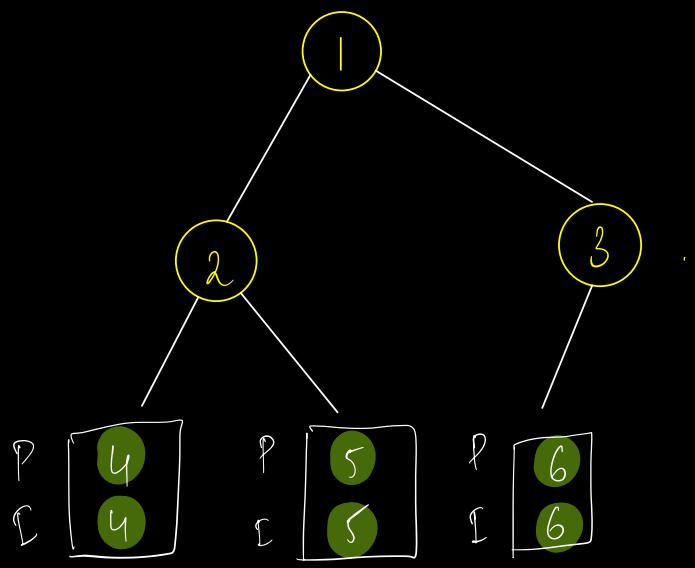


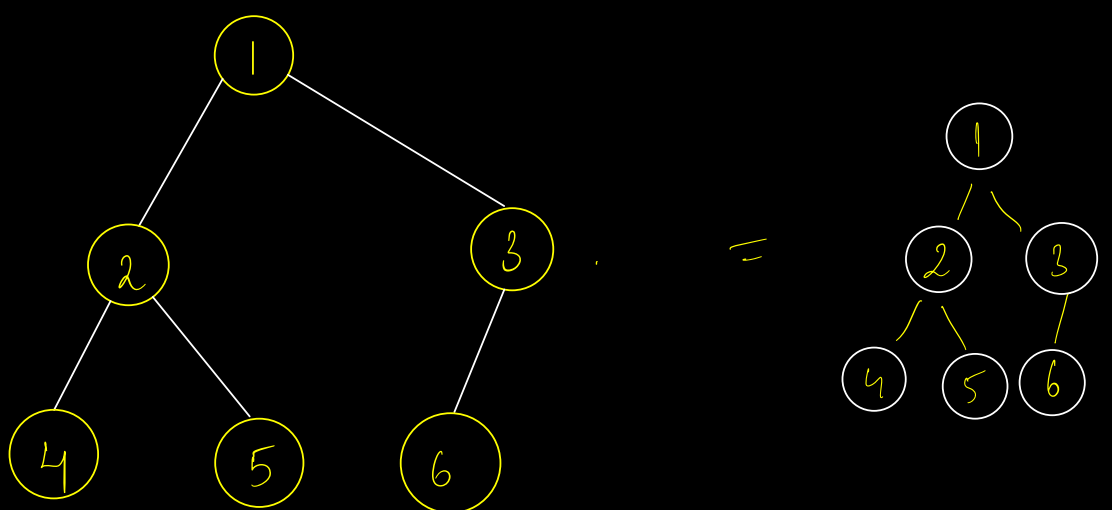




Recursion

{Pre / In}



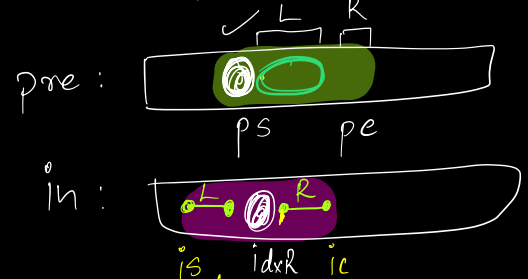


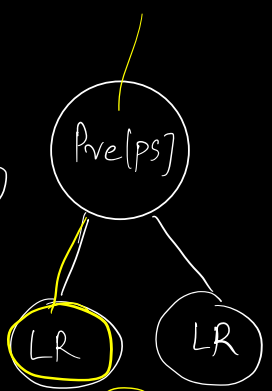
Next class

Ass: Given preorder & inorder, Construct tree & return root Node

```
Node createTree(pre[], in[], ps, pe, is, ie) {
    if(ps > pe) {
        return NULL;
    }
    Node root = new Node(pre[ps]);
    int idxR = find(pre[ps], is, ie, in[]); // Optimize using Hashmap
    root.left = create(pre, in, ps+1, ps+idxR-is, is, idxR-1)
    root.right = create(pre, in, ps+idxR-is+1, pe, idxR+1, ie)

    return root;
}
```





left subtree

nL = idxR - is

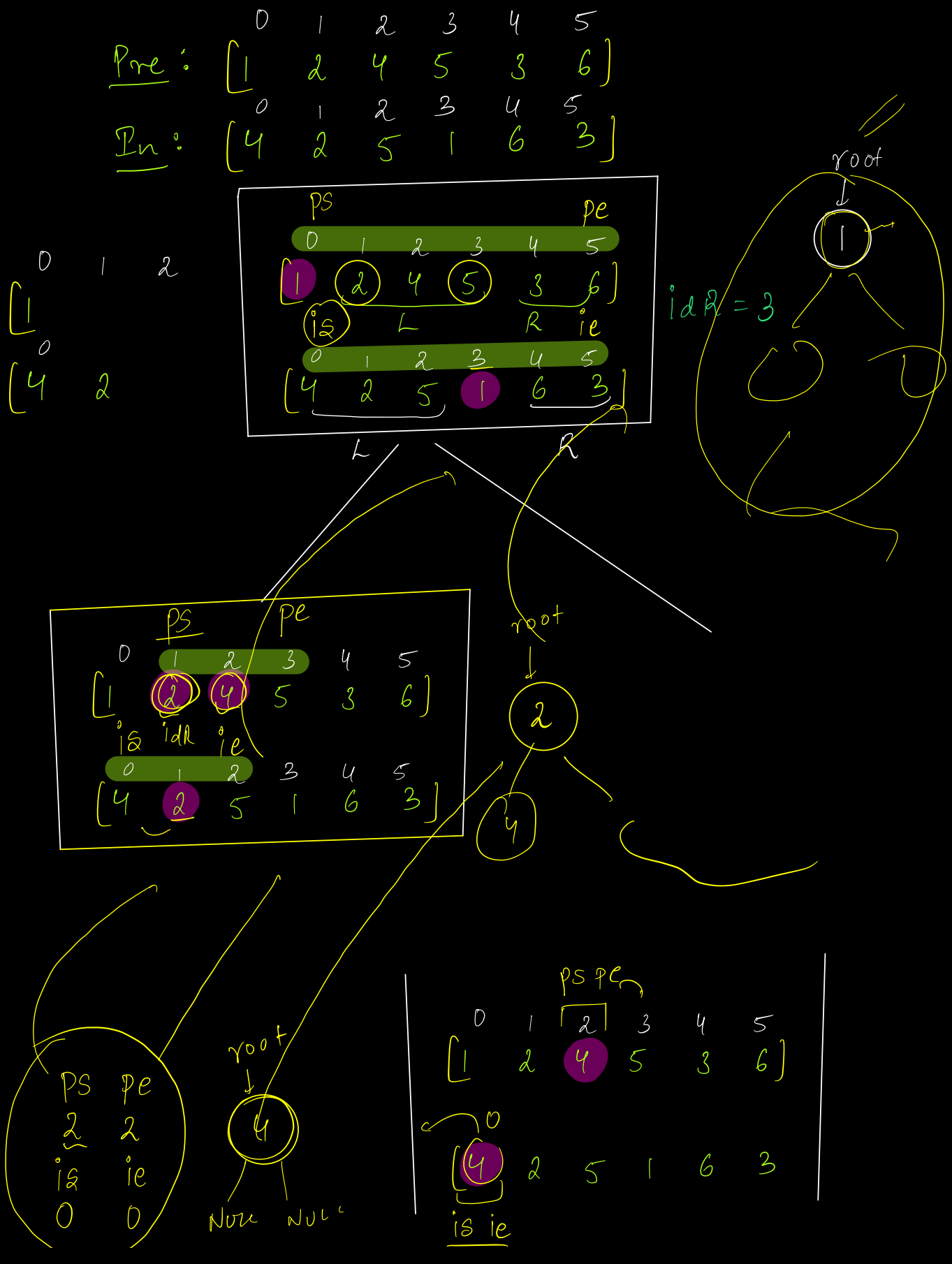
In(is, idR-1) (idR+1, ie)

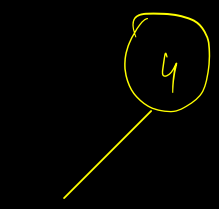
Pre(ps+1, ps+idR-is)

(ps+idxR-is+1, pe)

TC: O(N²) → O(N)

SC: O(N)


Pre: [1 2 4 5 3 6]
In: [4 2 5 1 6 3]
PS
PE
is
L
R
ie
idR = 3
root
NULL
NULL
PS PE
is ie



ps pe
3 2+0-0
3 > 2

Doubts - As | HW

1. 30 - 40 Min
2. Discuss in WA with your peers
3. TA
4. me