PW SKILLS

Binary Trees

Prerequisites

- Recursion → PIP
- Linked List
- Stacks & Queues (basic)

n = 2 → 2 1 1 1 2 1 1 1 2

PW SKILLS

Today's Checklist

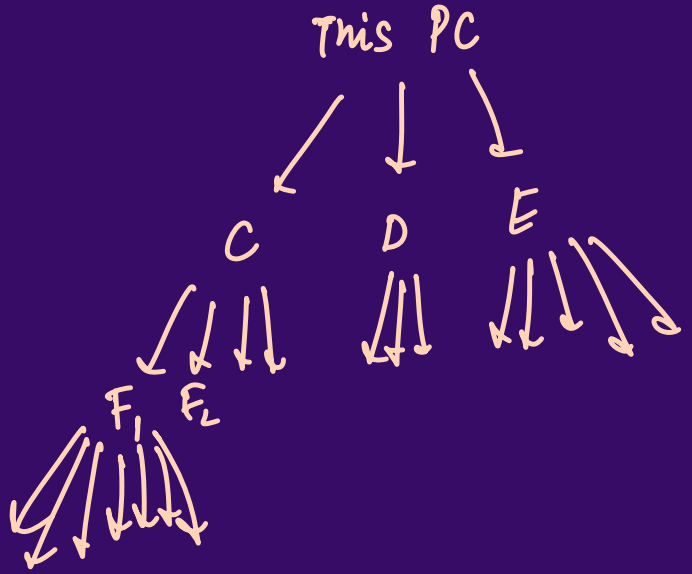
PW SKILLS

- What is a Tree data structure
- Representation
- Terminology
- Important properties of trees
- Types of Trees
- Applications of tree data structure
- What is a Binary Tree?
- Implementation
- Traversals
- Problems
- Types of Binary Trees

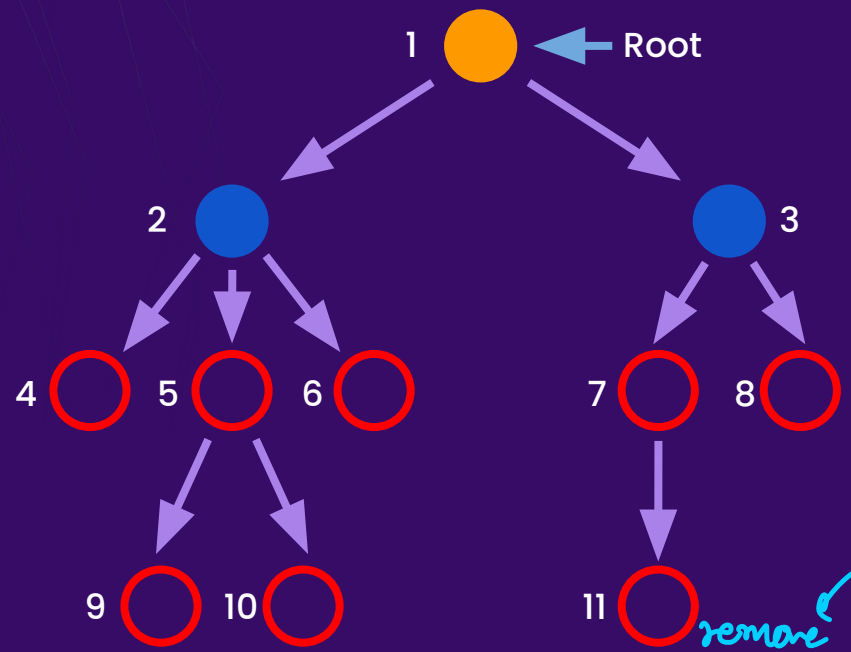
What is a Tree data structure

Array, LL, Stack, Queue → Linear data structures

Tree → heirarchical data structure



Representation

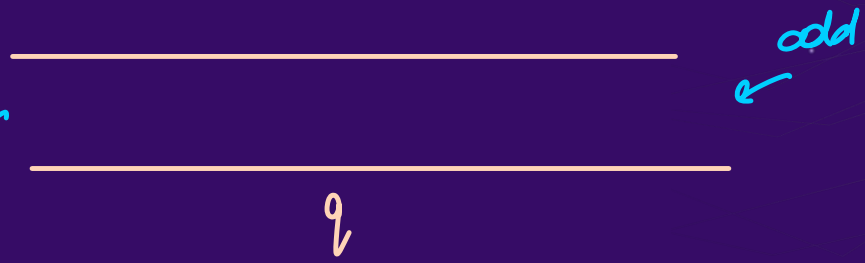


LL : head

Tree : Root

LL : tail

Tree : leaf nodes

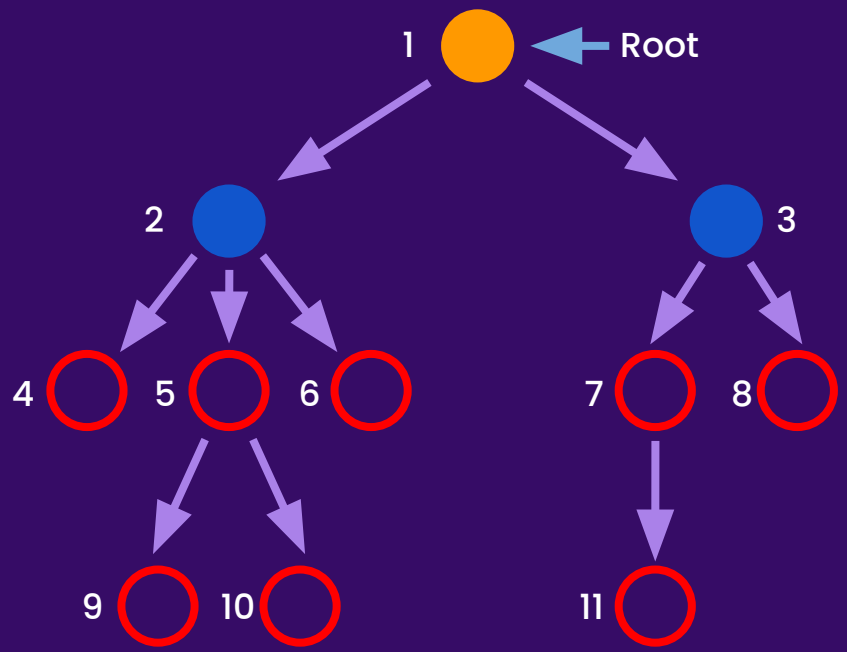


1 2 3 5 6 7 8 9 10 11

Terminology

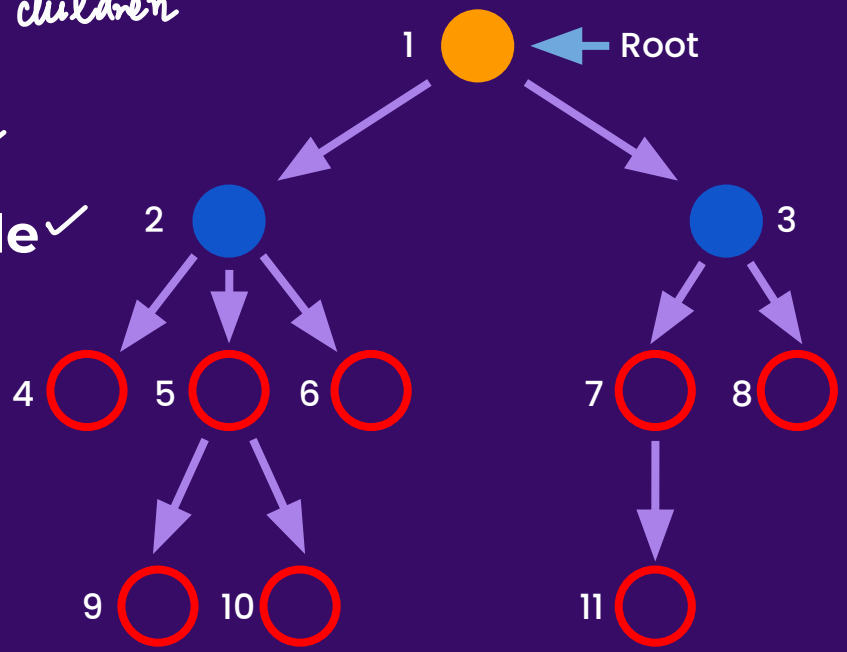
- Root ✓
- Child Node ✓
- Parent Node ✓
- Sibling Nodes ✓

means nodes with a common parent



Terminology

- **Leaf Node** → with 0 children
- **Internal Node**
- **Ancestor Node** ✓
- **Descendant Node** ✓

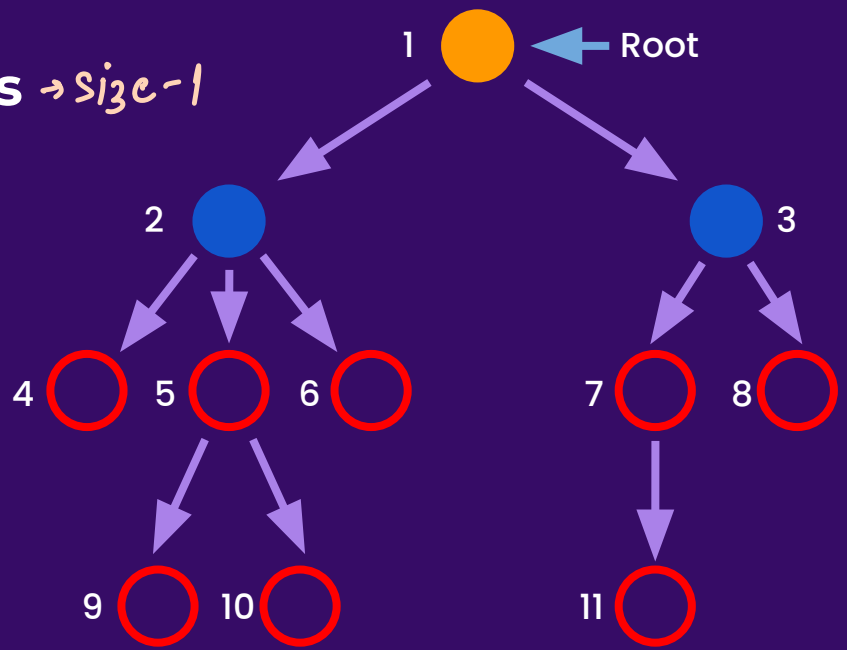


9 ke ancestor → 1, 2, 5

2 ke descendants : 4, 5, 6, 9, 10

Terminology

- **Level** : generations
- **Number of edges** → size - 1
- **Height** : levels - 1
- **Size** : no. of nodes



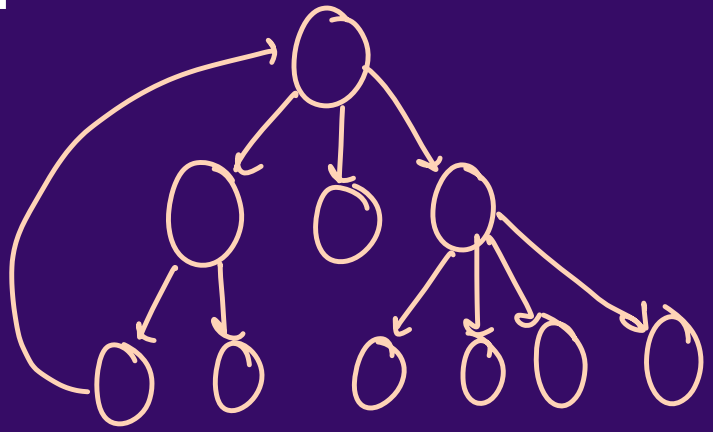
Levels = 4

Size = 11

Edges = 10

Important Properties of trees

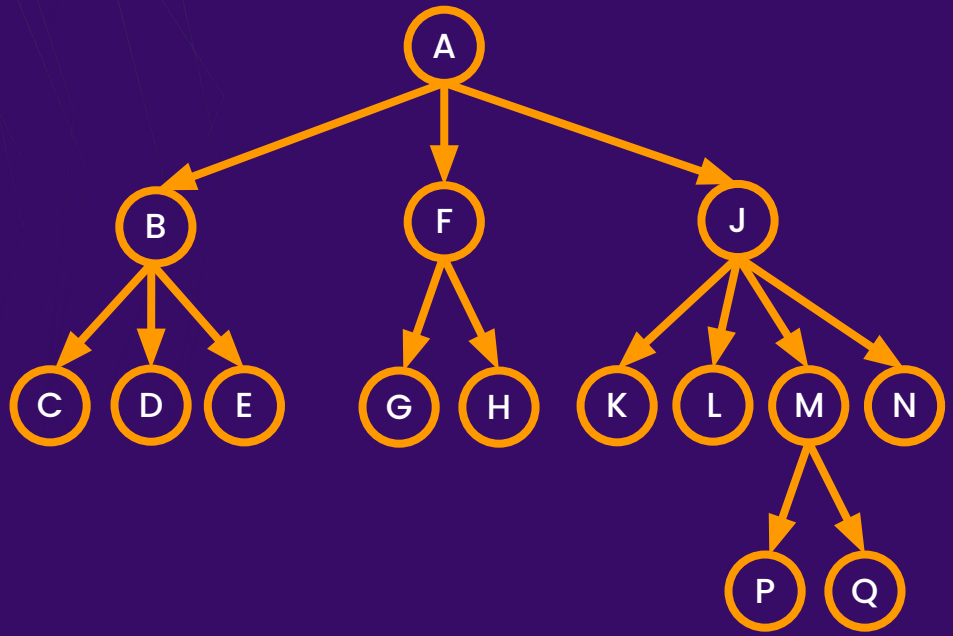
- Traversing in a tree is done by depth first search and breadth first search algorithm.
- It has no loop and no circuit.
- It has no self-loop.



Not a Tree

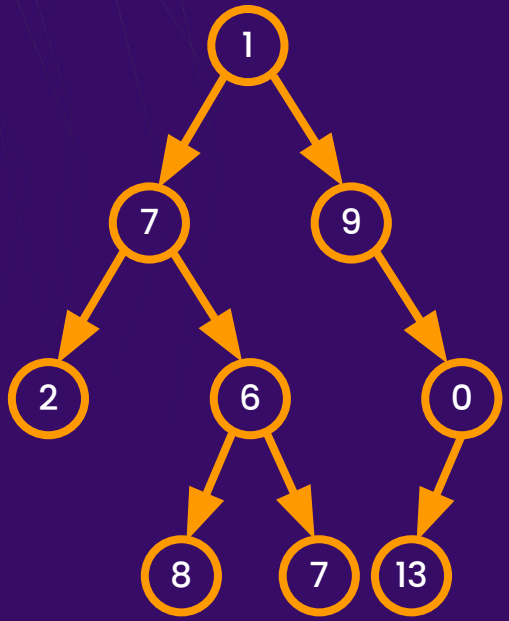
Types of Trees

1. Generic Trees



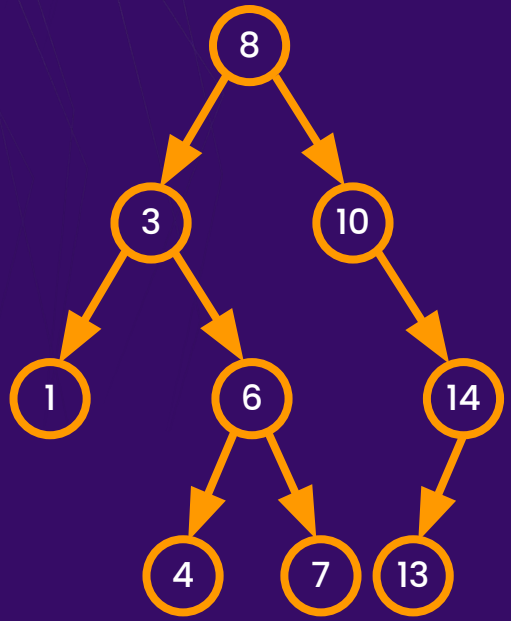
Any node in a generic Tree can have any no. of child nodes

2. Binary Trees



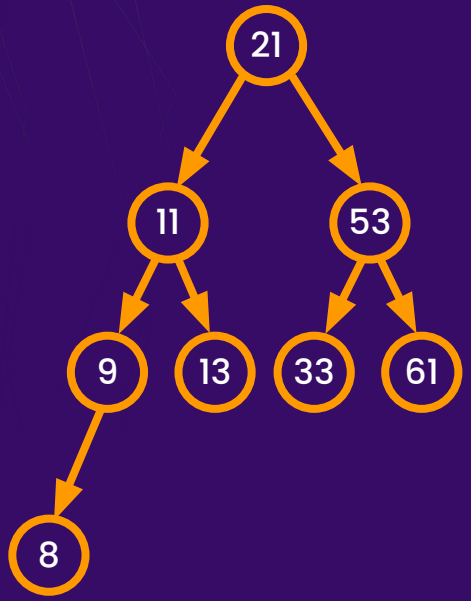
Any node can have 0, 1 or 2 children

3. Binary Search Trees



ISke baare me baad me acche se padhege

4. AVL Trees



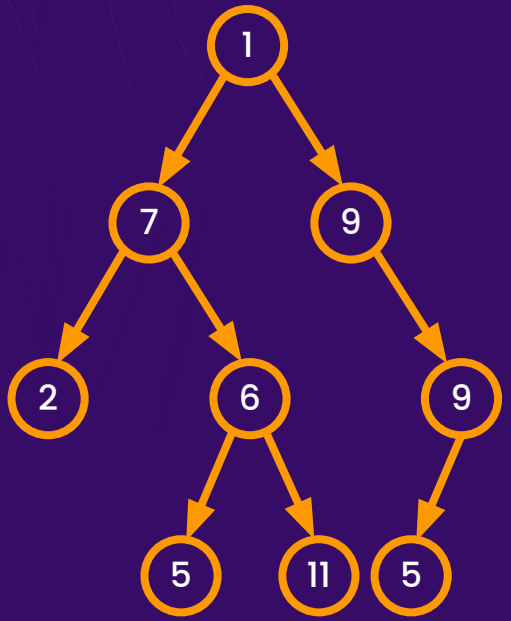
BST's which are self balancing

Applications of TREE Data Structure

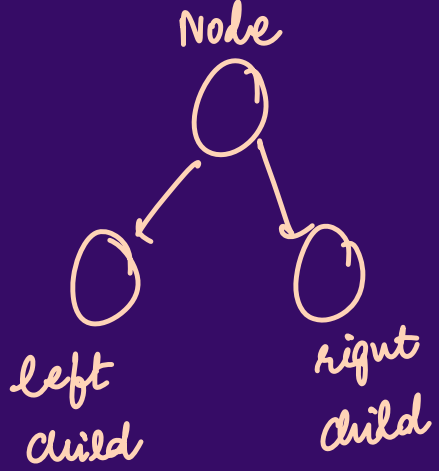
PW SKILLS

- Hierarchical data structure
- Searching efficiency
- Sorting
- Dynamic Data
- Efficient Insertion and Deletion
- Easy to implement

What are Binary Trees?



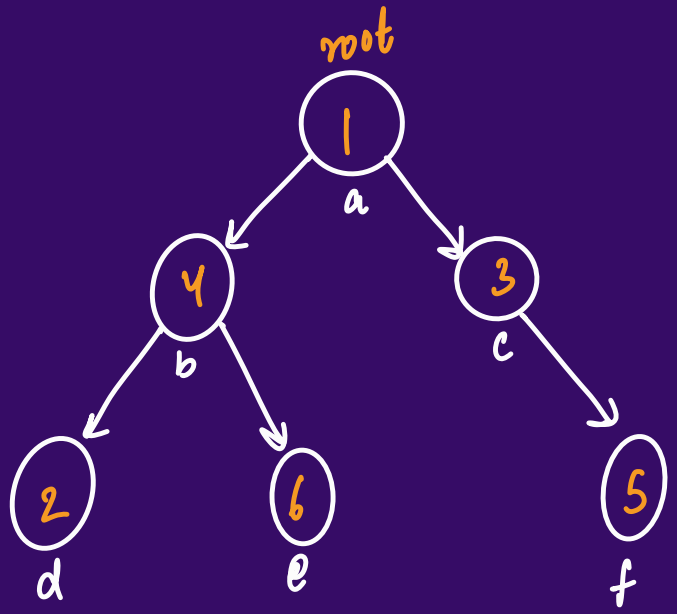
Trees where each node has either 0, 1 or 2 children



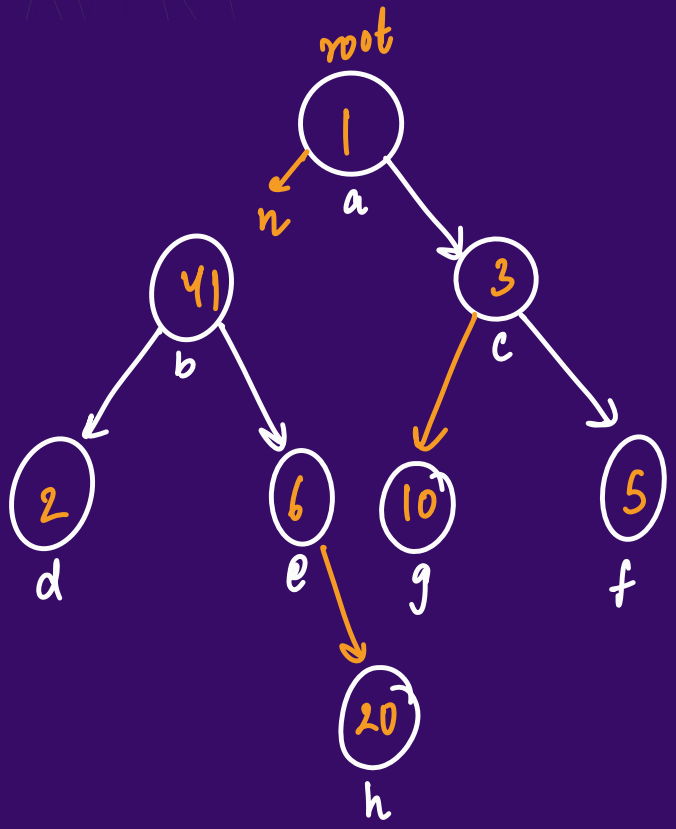
Implementation

- Creating a Node class

```
class Node {
    int val;
    Node left;
    Node right;
}
```



Display

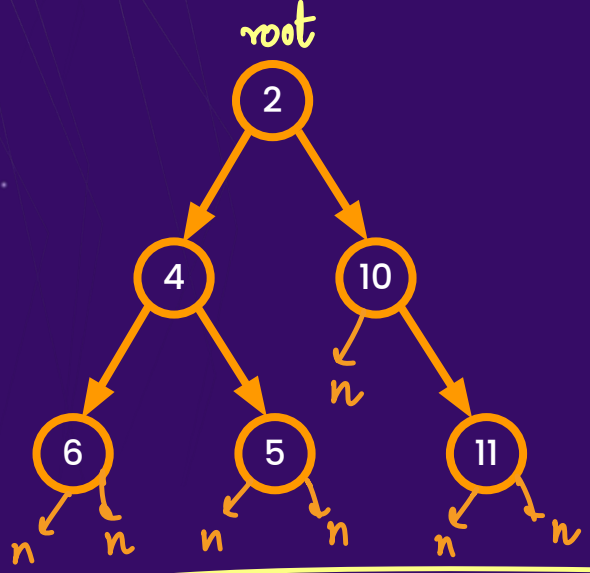


Every node in the tree is either the tree or a subtree

To Display

1) Ensure a base case
2) Print root's val
3) Recursion will print LST & RST

Display

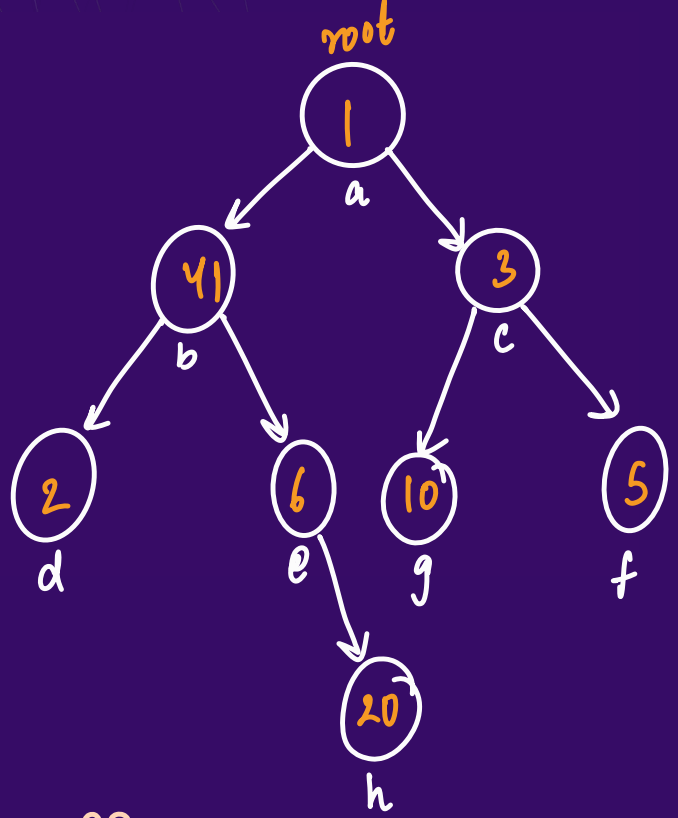



```java
private static void display(Node root){
    if(root==null) return; // Base Case
    System.out.print(root.val+" ");
    display(root.left); // Left SubTree
    display(root.right); // Right SubTree
}
```

Output

2 4 6 5 10 11

Find sum of tree nodes



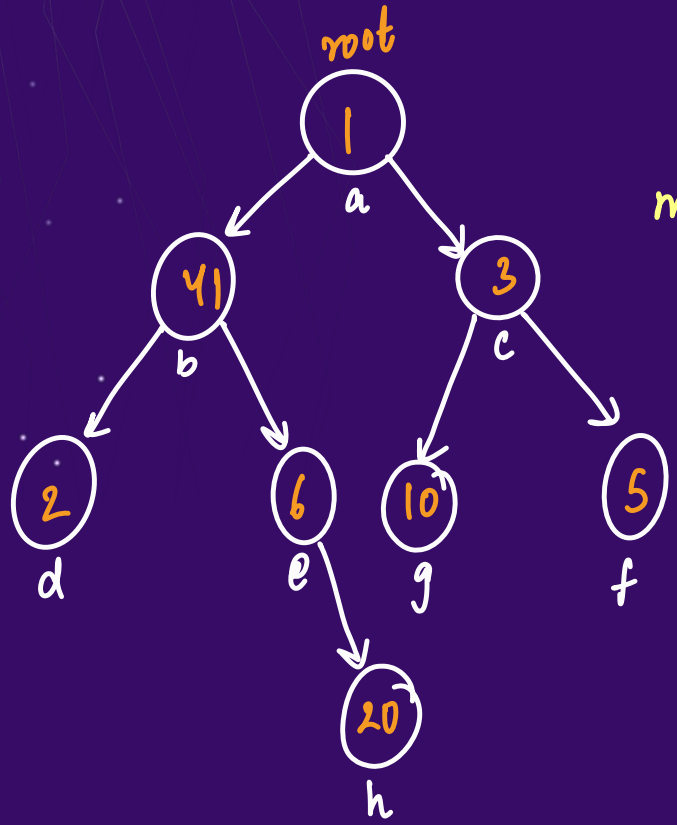

```
if(node==null) return 0;
Sum = root.val + sum(LST) + sum(RST)
```

Sum = 88

Homework :

1. Find product of tree nodes
2. Find product of non-zero elements of nodes

Find node with max value



max : 41

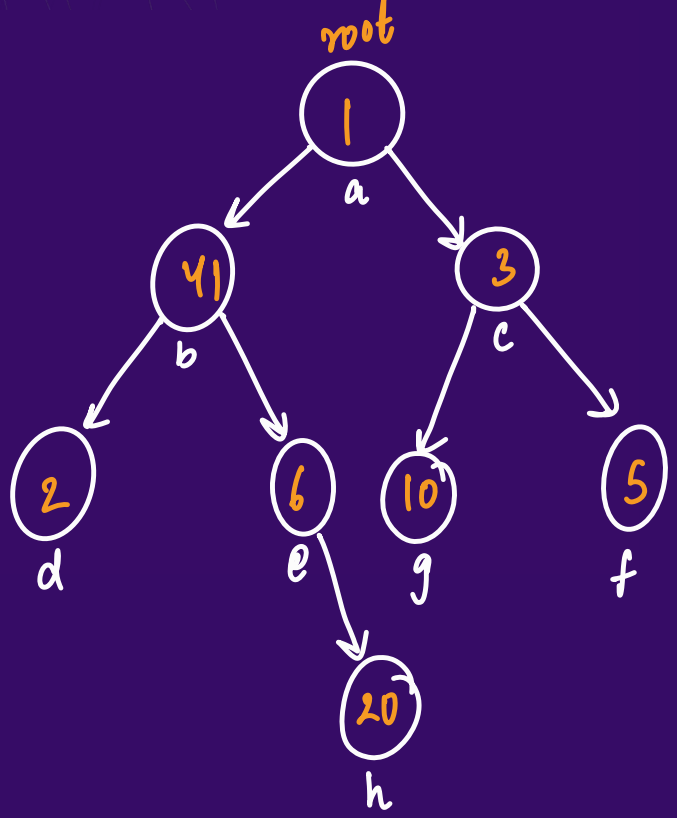
max = Max (root.val , Max (max(root.left) , max(root.right)))

Homework :

Find Node with minimum value

PW SKILLS

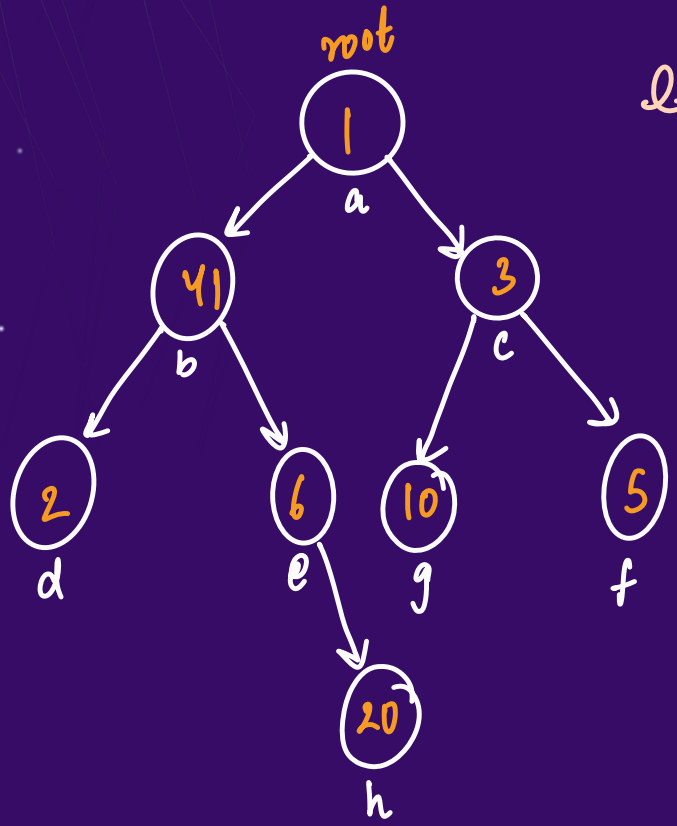
Find size of Binary Tree : no. of nodes



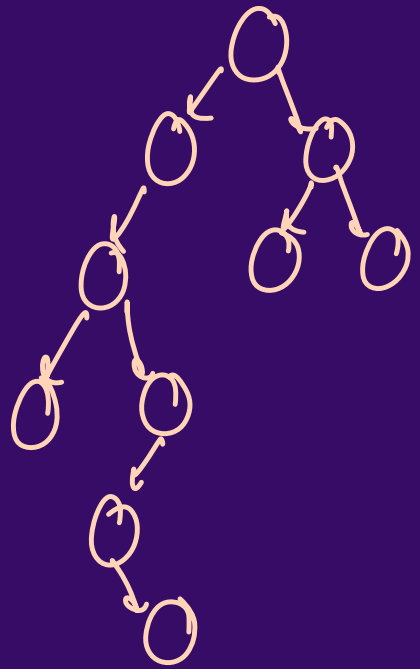
Size = 8

```
if(root ==null) return 0;
size = 1 + size(LST) + size(RST)
```

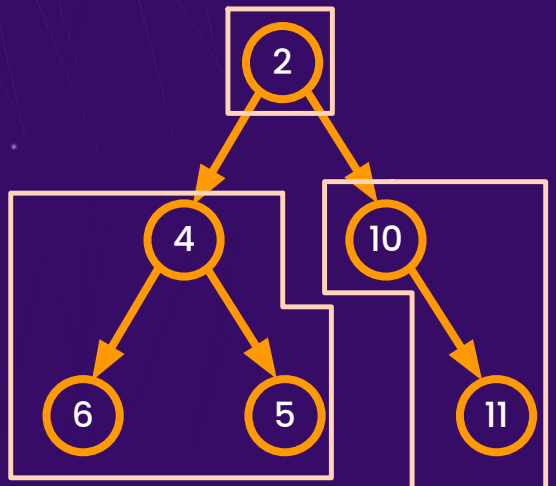
Find levels / height of Binary Tree



$$levels = 1 + \max(levels(LST), levels(RST))$$



Traversals



T.C. = O(n)

S.C. = O(n) → Recursive stack space

Root L R — PreOrder

L Root R — Inorder

L R Root — Postorder

Root R L — Reverse PreOrder

R Root L — Reverse Inorder

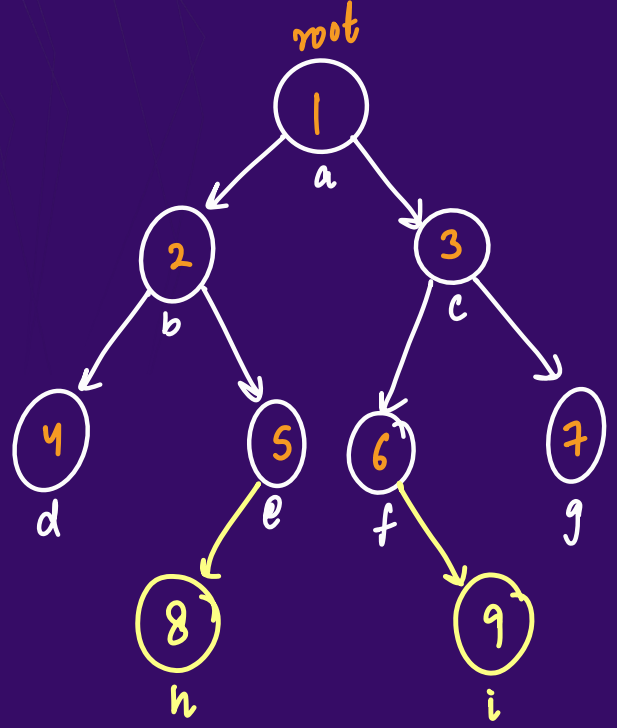
R L Root — Reverse Postorder

SKILLS

Ques:

PW SKILLS

Q : Binary Tree Preorder Traversal Root Left Right



1 (4 8 2 5) (3 6 7 9)

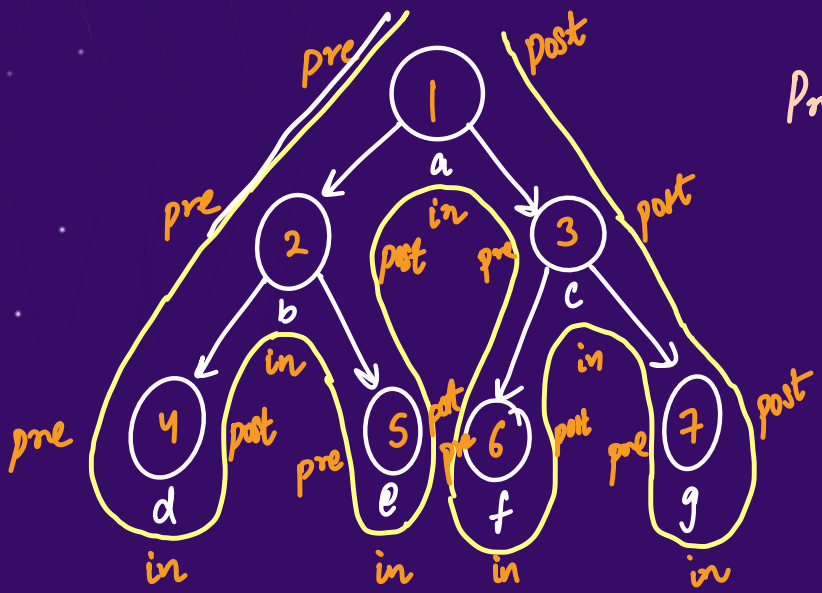
1 2 4 (8 5) 3 (6 9) 7

1 2 4 5 8 3 6 9 7

[Leetcode 144]

Ques:

Q : Binary Tree Preorder Traversal



Pre: 1 2 4 5 3 6 7

In: 4 2 5 1 6 3 7

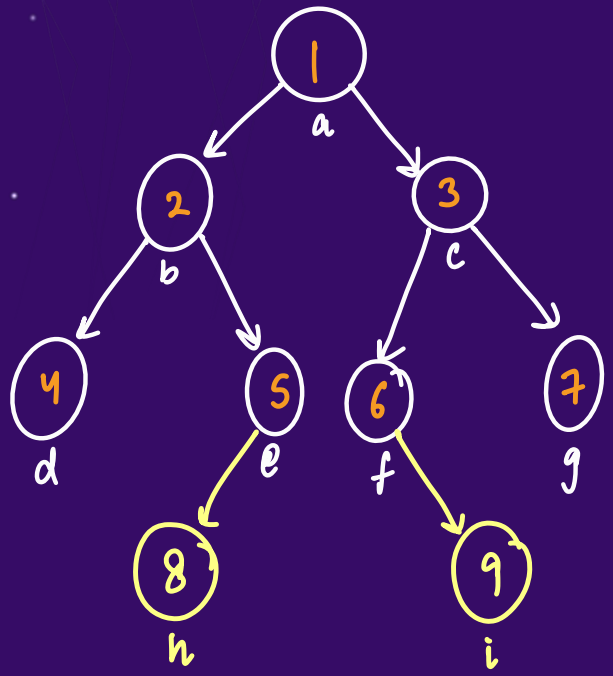
Post : 4 5 2 6 7 3 1

[Leetcode 144]

Ques:

PW SKILLS

Q : Binary Tree Inorder Traversal Left Root Right



(2 4 5 8) 1 (3 6 7 9)

4 2 (5 8) 1 (6 9) 3 7

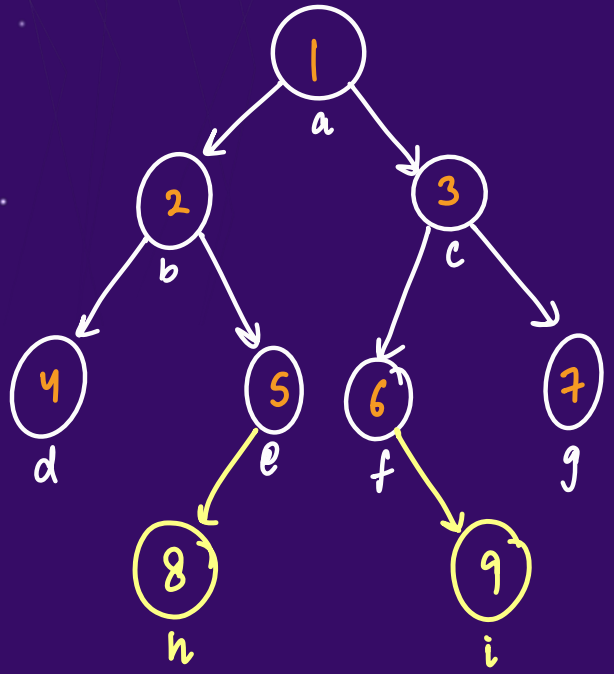
4 2 8 5 1 6 9 3 7

[Leetcode 94]

Ques:

PW SKILLS

Q : Binary Tree Postorder Traversal Left Right Root

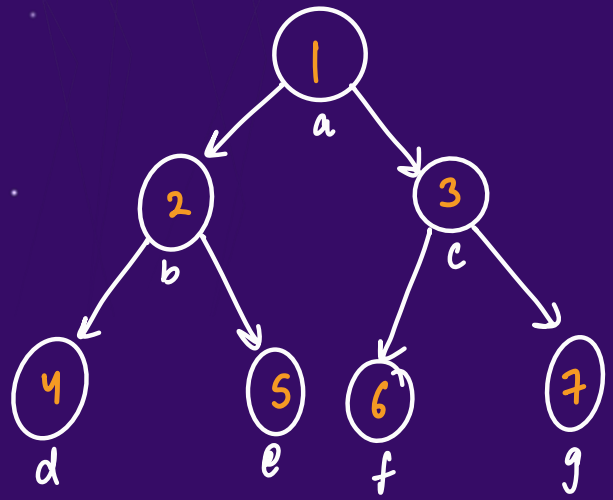


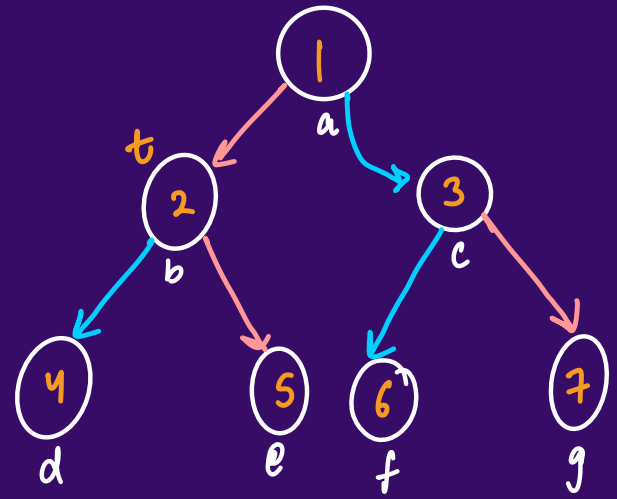
[Leetcode 145]

Ques:

PW SKILLS

Q : Invert Binary Tree



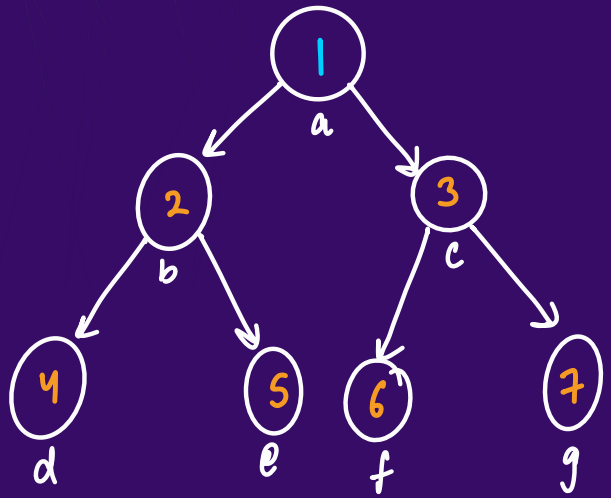


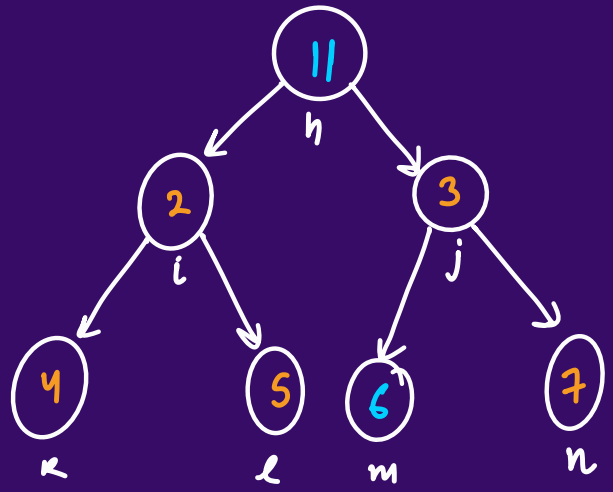
[Leetcode 226]

Ques:

PW SKILLS

Q : Same Tree

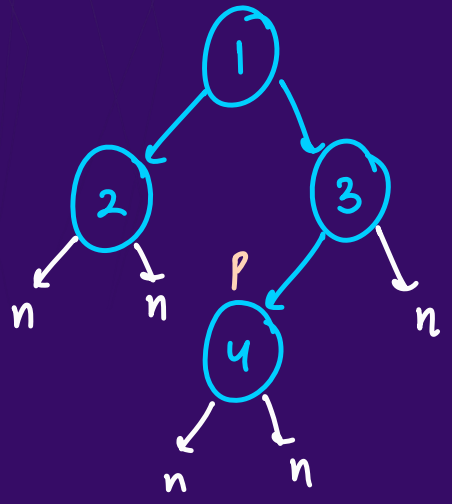


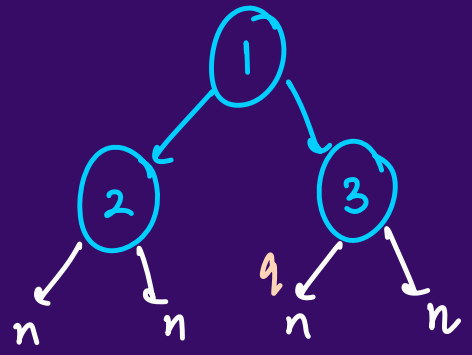


[Leetcode 100]

Ques:

Q : Same Tree





ans = false

[Leetcode 100]

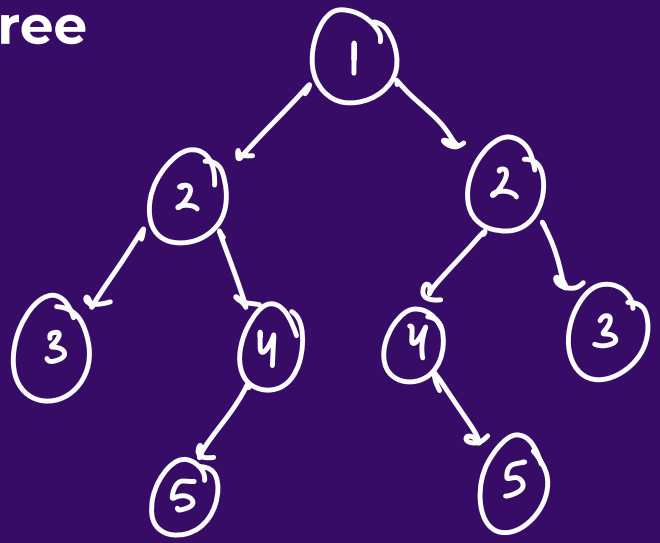
Homework:

Q : Symmetric Tree

Follow Up :

↓

Do it & at the end, the structure of tree should be same as before



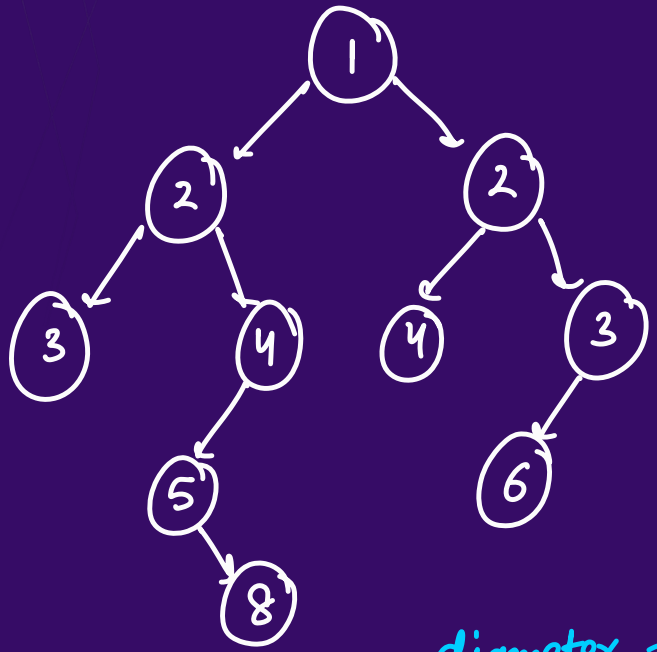
yes this is symmetric

[Leetcode 101]

Ques:

(largest path)

Q : Diameter of Binary Tree



diameter = height(LST) + height(RST) + 2

very very wrong

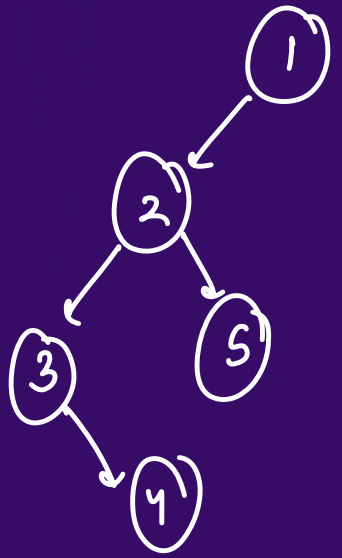
diameter = 7 (edges)

[Leetcode 543]

Ques:

PW SKILLS

Q : Diameter of Binary Tree



$$\text{diameter} = [\text{levels(LST)} + \text{levels(RST)} + 1] - 1$$

[Leetcode 543]

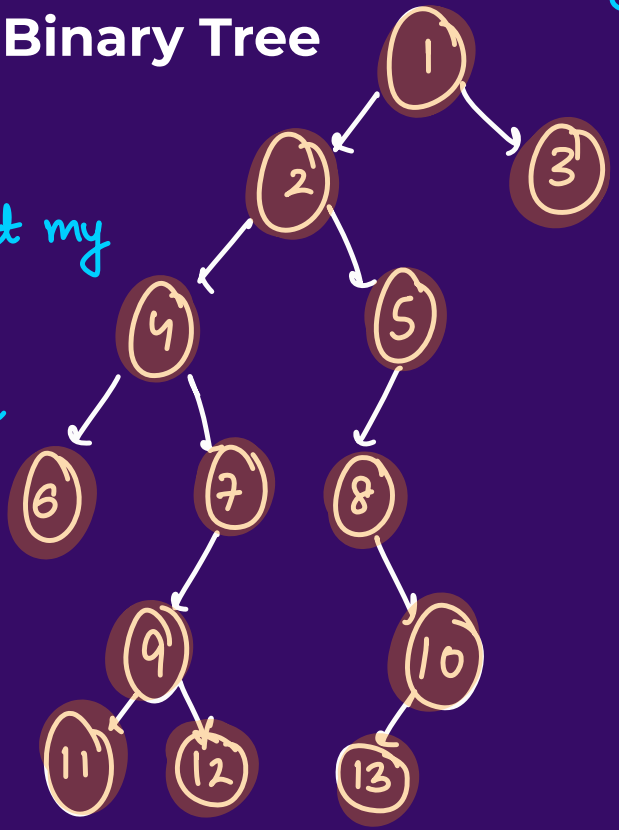
Ques:

dia = 6 ✗
dia = 8 ✓

PW SKILLS

Q : Diameter of Binary Tree

I was trying to get my diameter which is passing through the root.



dia = {6, 8, 0, 4, 0, 2, 2, 0, 0, 3, 2, 1, 0}

T.C. = $O(n^2)$

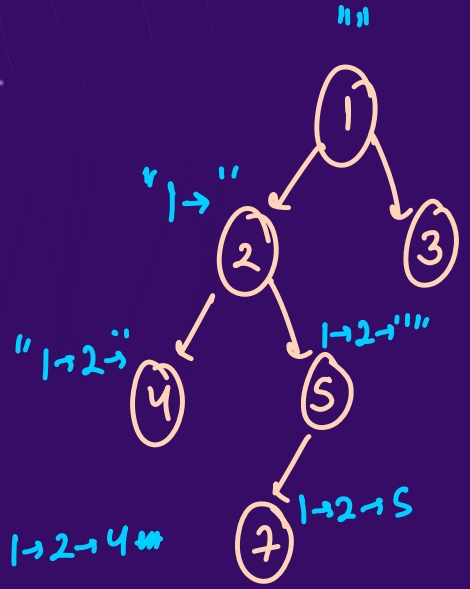
S.C. = $O(n)$ → rec

[Leetcode 543]

Ques:

PW SKILLS

Q : Binary Tree Paths



Root to leaf paths:

1) "1→2→4"

2) "1→2→5→7"

3) "1→3"

T.C. = O(n)

S.C. = O(n·h)

h → height of tree

[Leetcode 257]

Homework:

Q : Path Sum

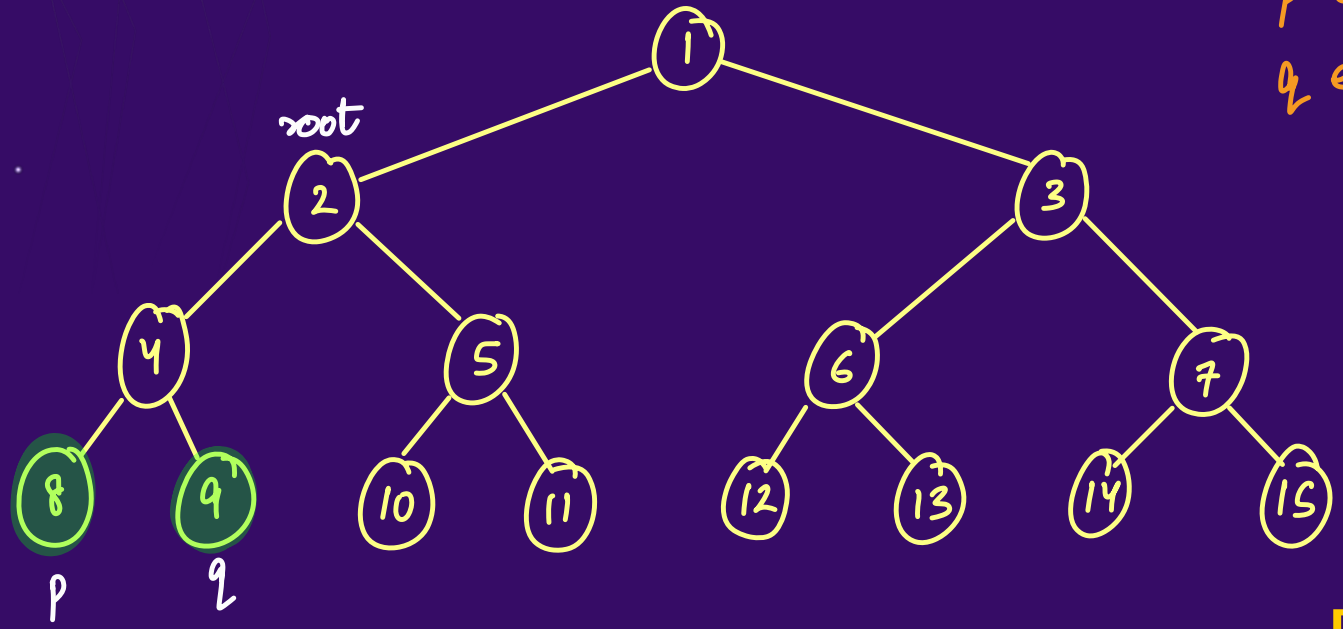
[Leetcode 112]

Very very very Famous

Ques:

LCA is of 2 or more given nodes

Q : Lowest Common Ancestor of a Binary Tree



p exists in LST ?

q exists in LST ?

$n + \frac{n}{2} + \frac{n}{4} \cdots$

$\sim 2n$

T.C. = O(n)

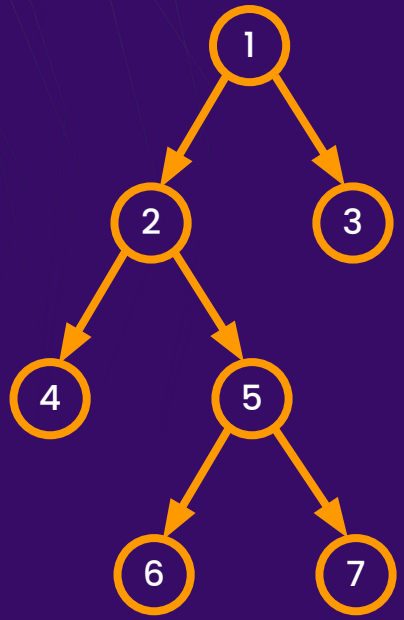
or $O(n^2)$

[Leetcode 236]

Types of Binary Trees

PW SKILLS

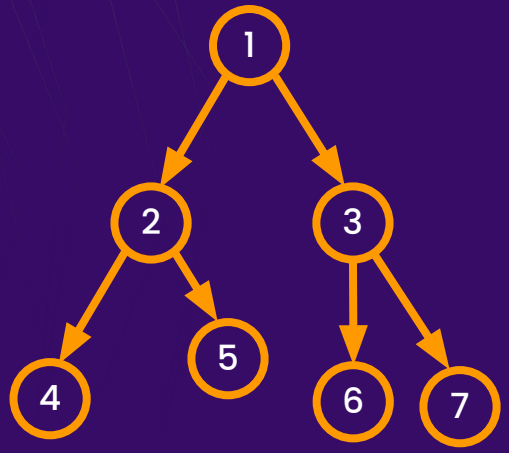
1. Full Binary Tree (0 or 2)



1

Child nodes

2. Perfect Binary Tree

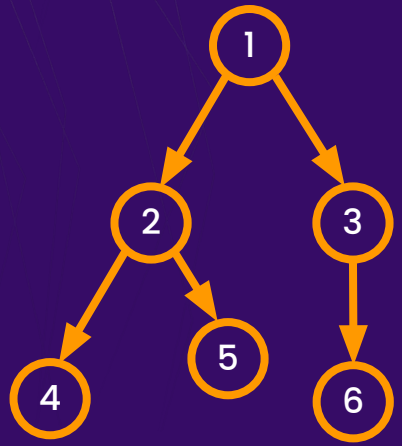


1

Every node has 2 child nodes except the nodes of the level which have 0 child node

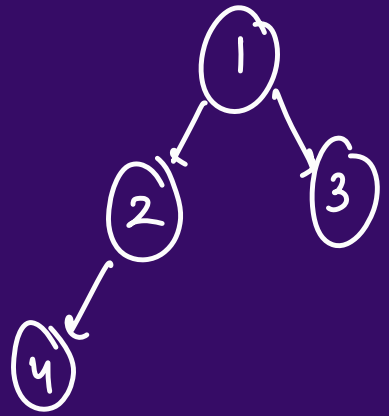
3. Complete Binary Tree

heaps

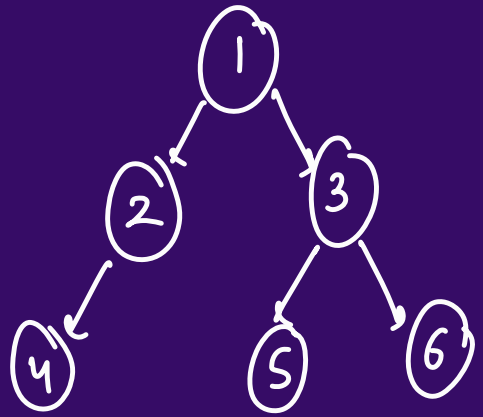


↓

Is a perfect binary tree but its last level can be incomplete



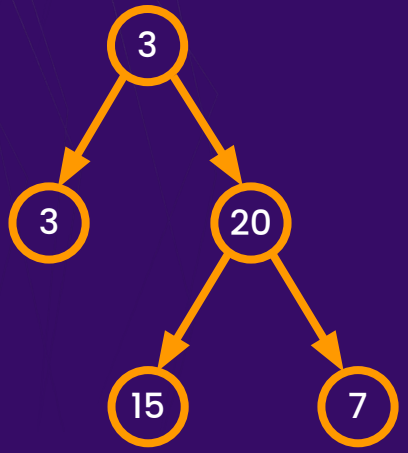
Yes



No

4. Balanced Binary Tree → Very Important

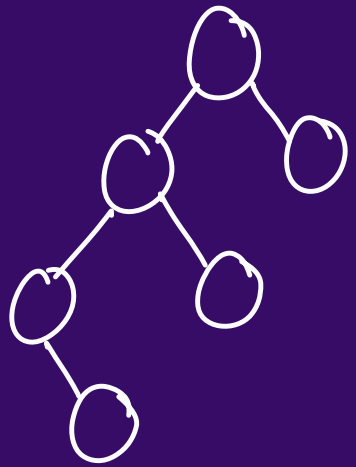
PW SKILLS



↓

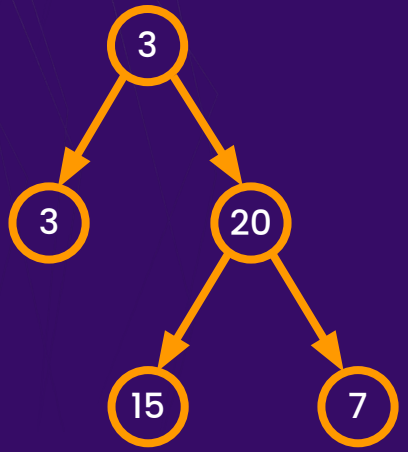
where diff. b/w levels (LST) & levels (RST) <= 1

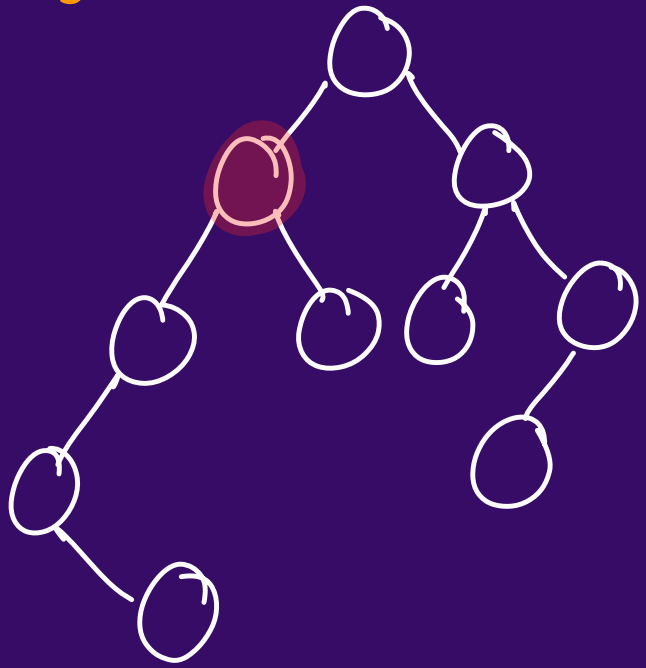
$$|\text{levels(LST)} - \text{levels(RST)}| <= 1$$



Unbalanced

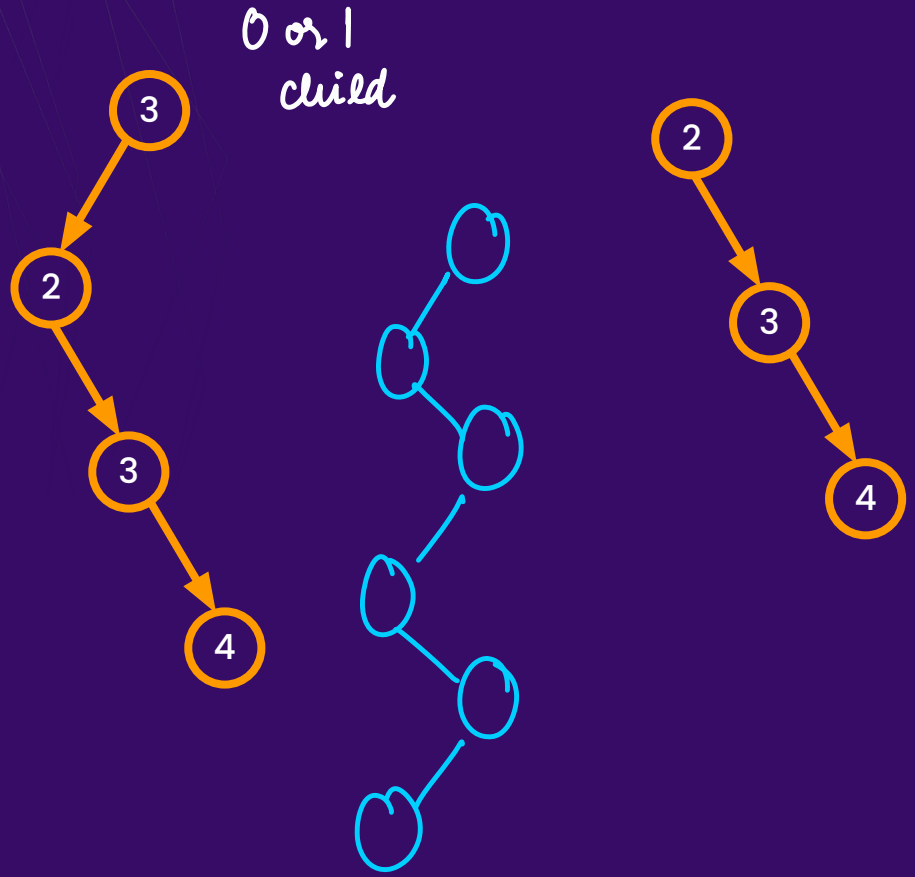
4. Balanced Binary Tree





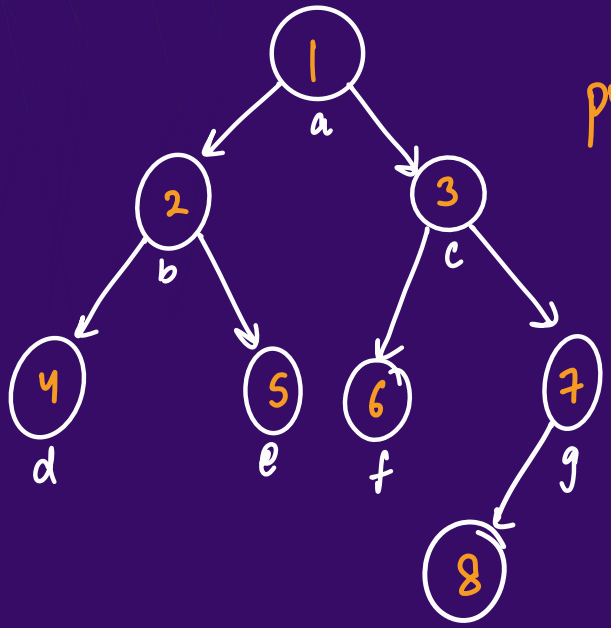
Not Balanced

5. Degenerate and Skewed Binary Trees



Level order traversal (Using Queue)

BFS → breadth first search , level wise search

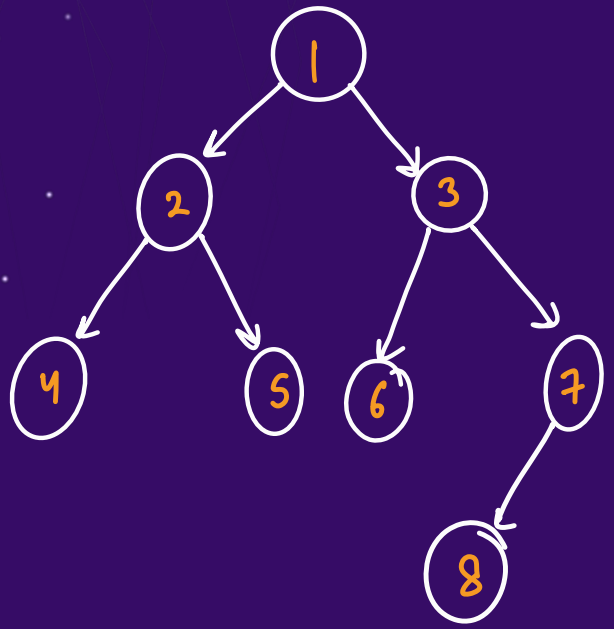


1
2 3
4 5 6 7
8

BFS → 1 2 3 4 5 6 7 8

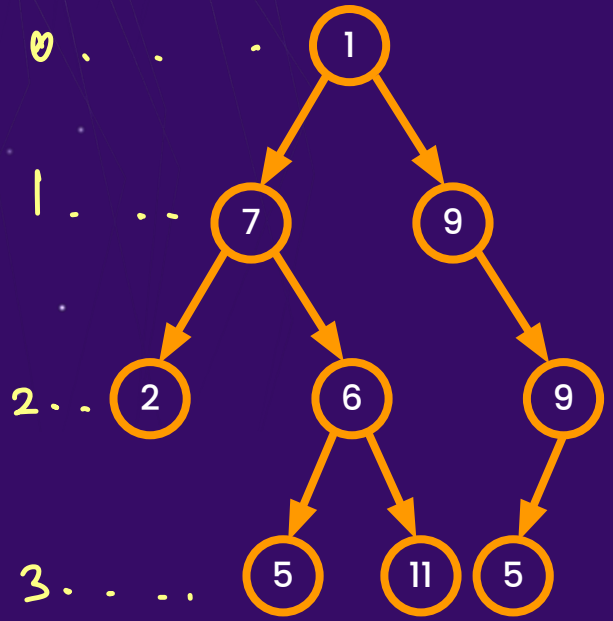
Level order traversal (Using Queue)

Right to Left



1 3 2 7 6 5 4 8

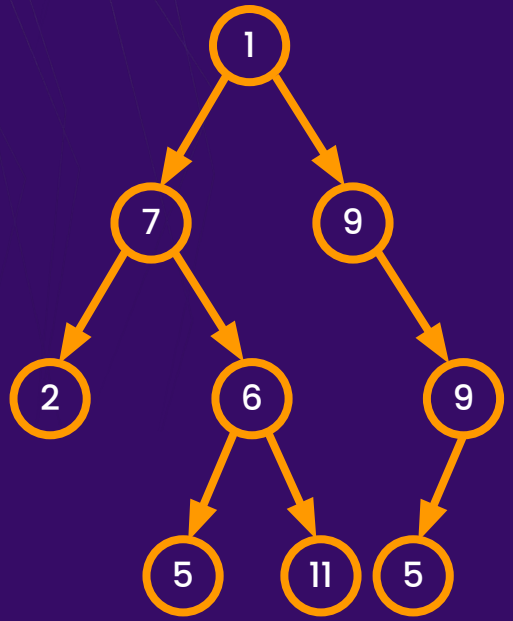
Print elements of nth level (left to right)



2^{nd} level → 2 6 9

preorder (root, level)

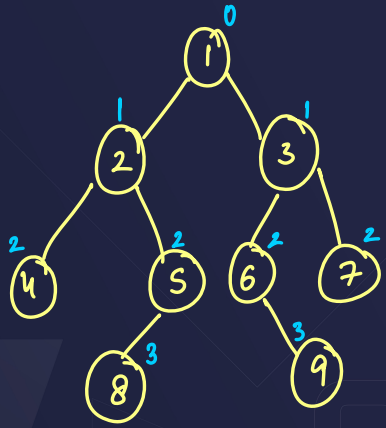
Print elements of nth level



Preorder ✓

Inorder ✓

Postorder ✓


0
1
1
2
1
3
2
4
2
5
2
6
2
7
3
8
3
9

Level order traversal (using nth level)

Kuchh nahi karna

aapko bas 0 to n-1 sabko call lagado

*Level order traversal (Right to Left)

Homework

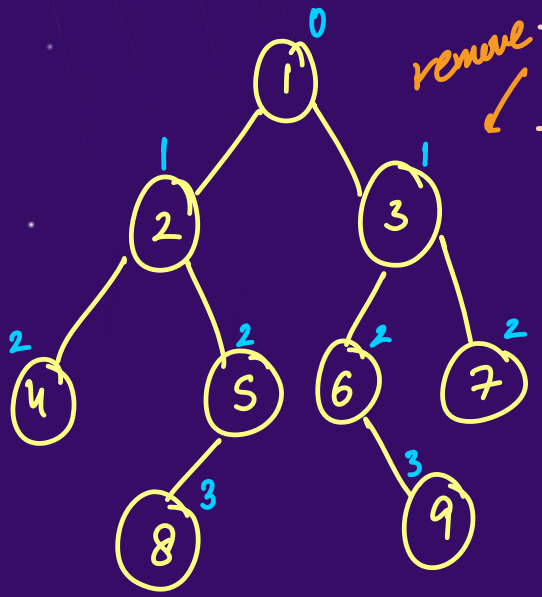
Ques:

Queue < Pair > q = new LinkedList<>();

node lvl

PW SKILLS

Q : Binary Tree Level Order Traversal (Queues)



remove

add

{1,0} (2,1) (3,1) (4,2) (5,2) (6,2) ,(7,2)
(8,3)(9,3)

1 2 3 4 5 6 7 8 9

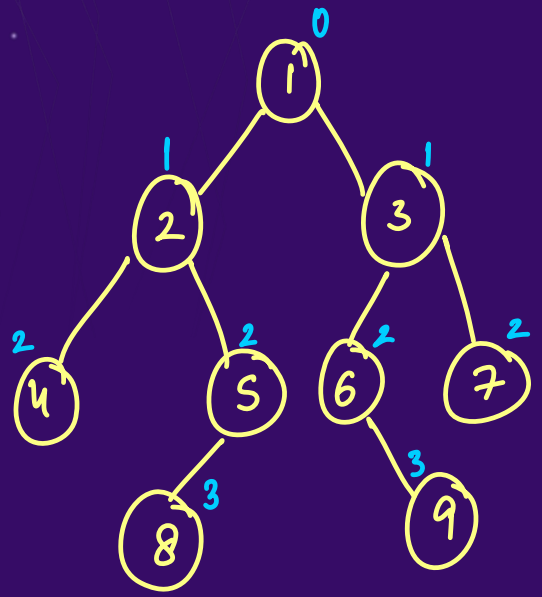
[Leetcode 102]

Ques:

T.C. = O(n) S.C. = O(n)

PW SKILLS

Q : Binary Tree Level Order Traversal



List < List < Integer > > ans

0	1	2	3

{ { 3 , { 3 , { 3 , { 3 } }

For Ex:

you have to add 4

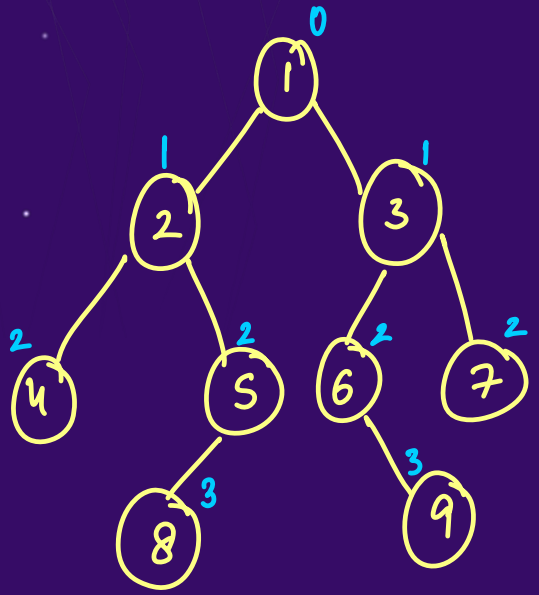
ans.get(lvl).add(4)

[Leetcode 102]

Ques: Homework

PW SKILLS

Q : Binary Tree Level Order Traversal (zigzag)



{ {1} {3 2} {4 5 6 7} {9 8} }

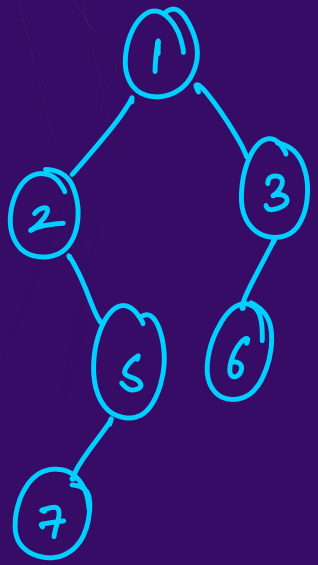
Do it

1) BFS → queue <Pair>

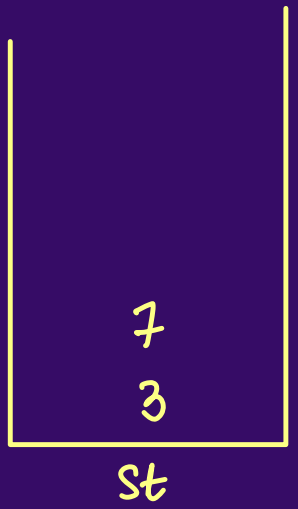
2) DFS → preorder (n^{th} level)

[Leetcode 103]

1. Preorder Traversal (Iterative)



Root Left Right



Recursive
↓
Stack
↓
Auxiallary Space

Time Complexity : O(n)

Space Complexity : O(n)

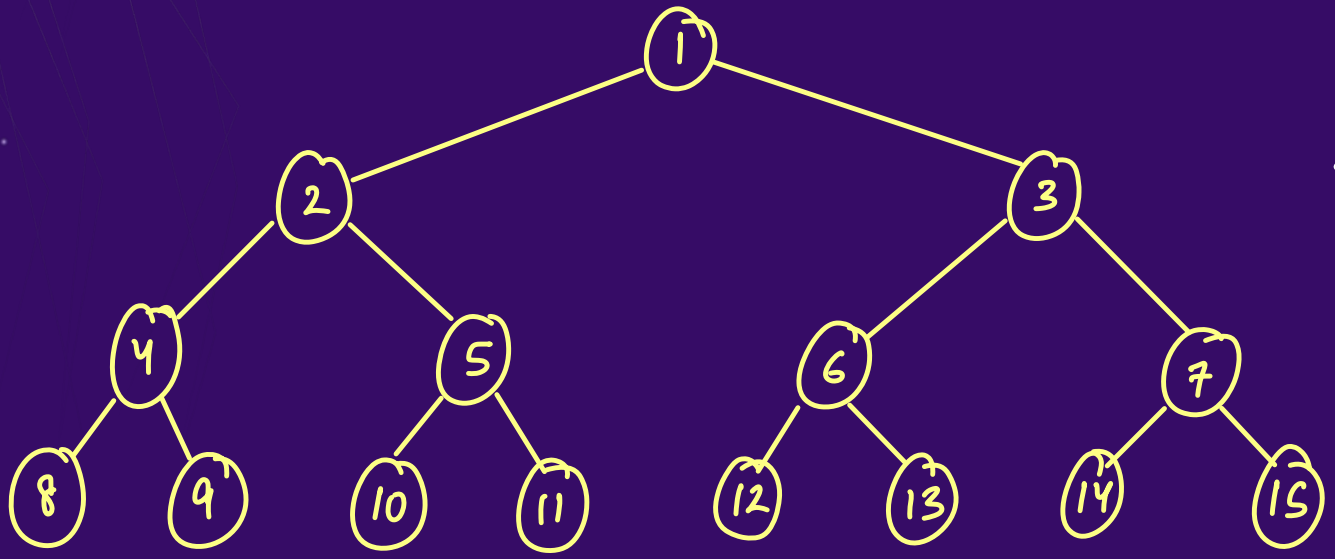
Auxillary Space : $O(\log_2 n)$

where h is level of tree

ans = { 1 2 5 }

Best Case = O(1)

1. Preorder Traversal (Iterative)



$n = 15$

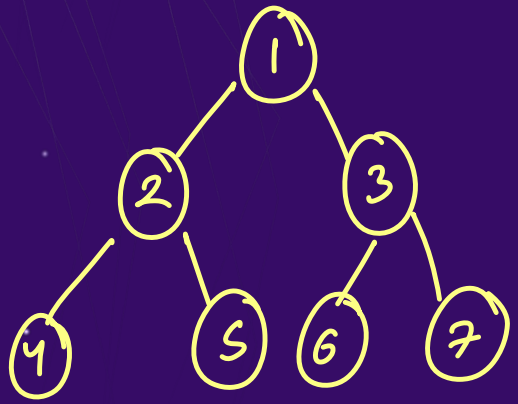
$\log_2 n \cong$ level $= 4$

8
9
5
3

For a balanced B.T. Pre: 1 2 4

if n are nodes then height/levels $\cong \log_2 n$

1. Preorder Traversal (Iterative)



PreOrder : Root left Right

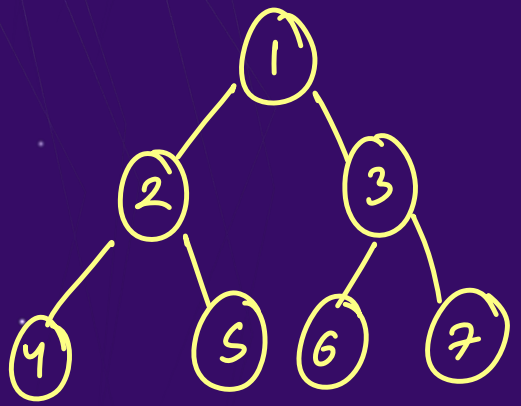
1 2 4 5 3 6 7

Reverse PreOrder : Root Right left

1 (3 6 7) (2 4 5)

1 3 7 6 2 5 4

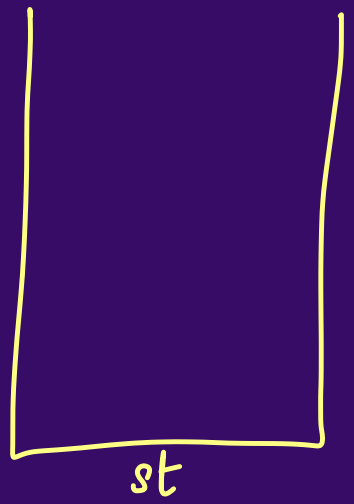
2. Inorder Traversal (Iterative)



t

Inorder : left root right

4 2 5 1 6 3 7




```
while(true){
    if(temp != null)
        st.push(temp)
        temp = temp.left
    else{
        Node top = st.pop()
        print
        temp = top.right
    }
}
```

break

3 3

Stack & a variable (Node)

temp = 3

ans = { 4 2 5 1 6 3 7 }

2. Inorder Traversal (Iterative)

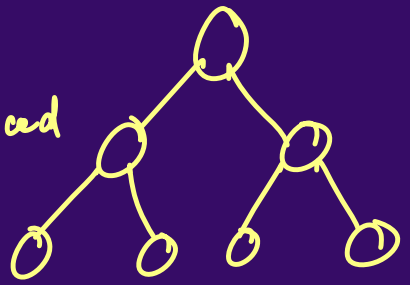
PW SKILLS

Time Complexity : $O(n)$

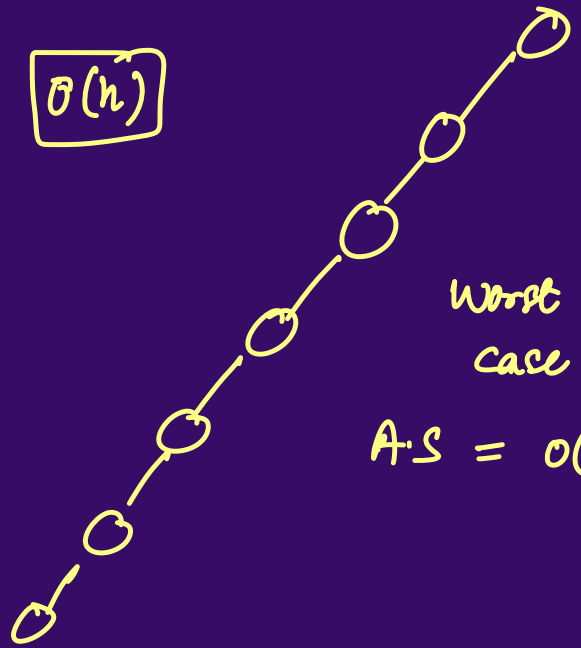
Space Complexity : $O(n)$

Auxillary Space : Max Space of stack $\boxed{O(n)}$

Best Case : Balanced B.T.



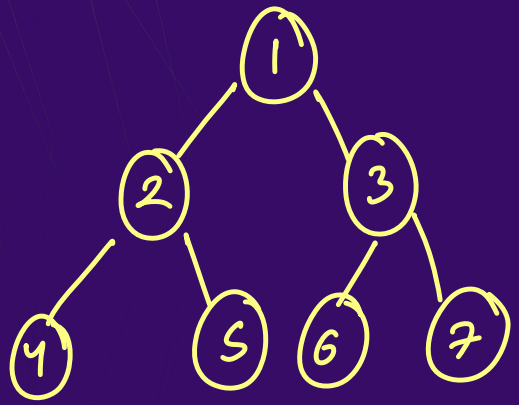
A.S $O(\log_2 n)$



Worst Case

A.S = $O(n)$

3. Postorder Traversal (Iterative)

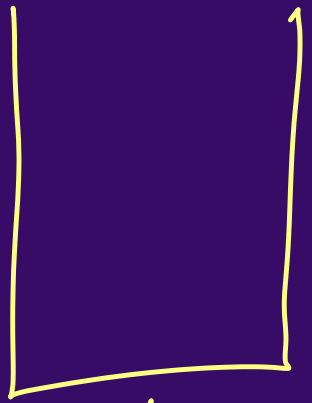


Left Right Root

4 5 2 6 7 3 1

Reverse (Root Right left)

== PostOoder



T.C. = O(n)

S.C. = O(n)

A.S. = O(n)

Reverse PreOrder

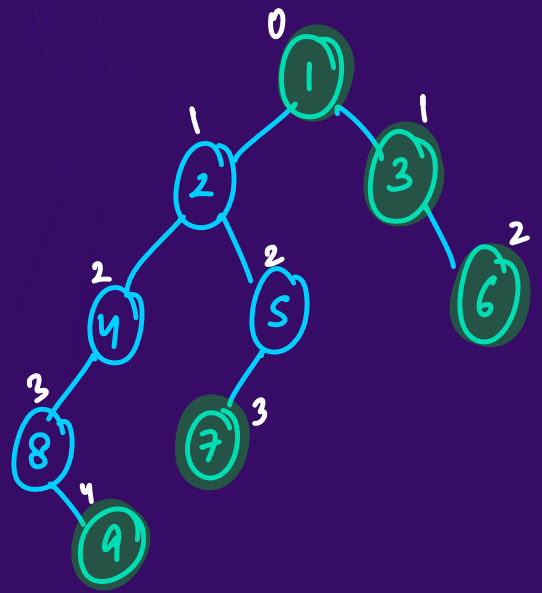
ans | 1 | 3 | 7 | 6 | 2 | 5 | 4 |

Kya hai ?

Ques:

PW SKILLS

Q : Binary Tree Right Side View → All the nodes which are at the rightmost in each level



Hint :

Preorder, InOrder, PostOrder

n^{th} level

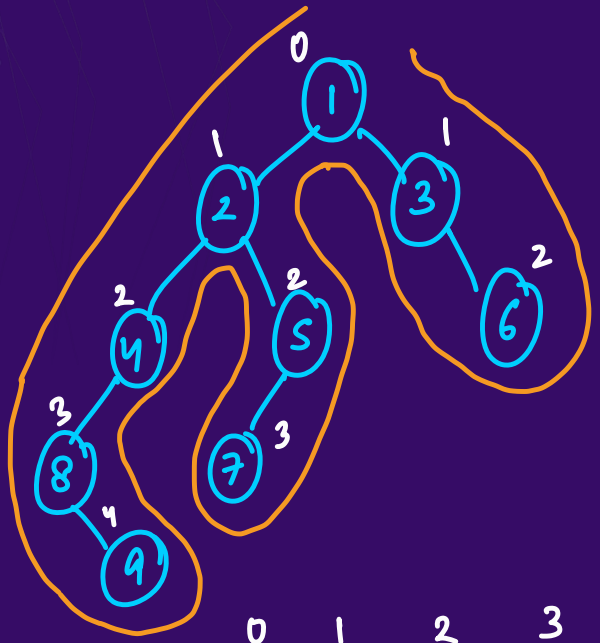
ans = { 1, 3, 6, 7, 9 }

[Leetcode 199]

Ques:

Q : Binary Tree Right Side View

levels = 5



	0	1	2	3	4
ans =	1	3	6	7	9

	3	4	2	1	3	2	0	1	2
inorder =	8	9	4	2	7	5	1	3	6

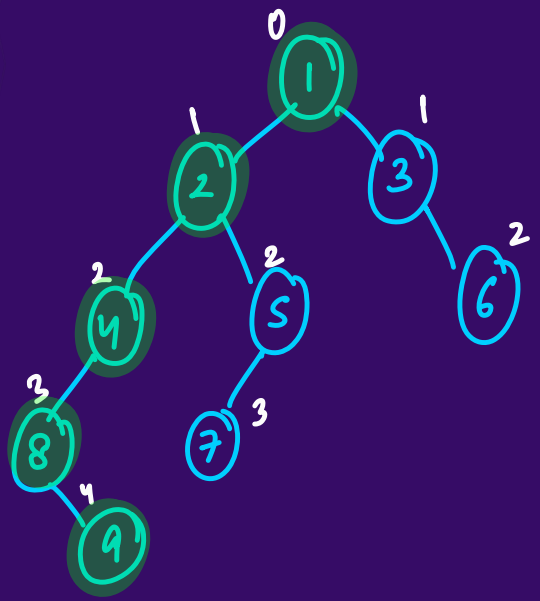
	0	1	2	3	4	2	3	1	2
preorder =	1	2	4	8	9	5	7	3	6

[Leetcode 199]

Homework:

PW SKILLS

Q : Binary Tree left Side View

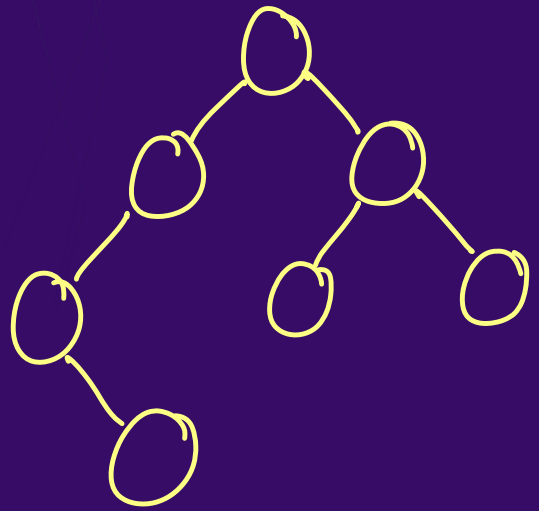


ans = {1, 2, 4, 8, 9}

Ques:

PW SKILLS

Q : Balanced Binary Tree



$|levels(LST) - levels(RST)| <= 1$

$T.n.o. = n + n-1 + n-2 + \cdots 1$

$T.C. = O(n^2)$

[Leetcode 110]

Ques:

PW SKILLS

Q : Diameter of Binary Tree [O(n) Approach]

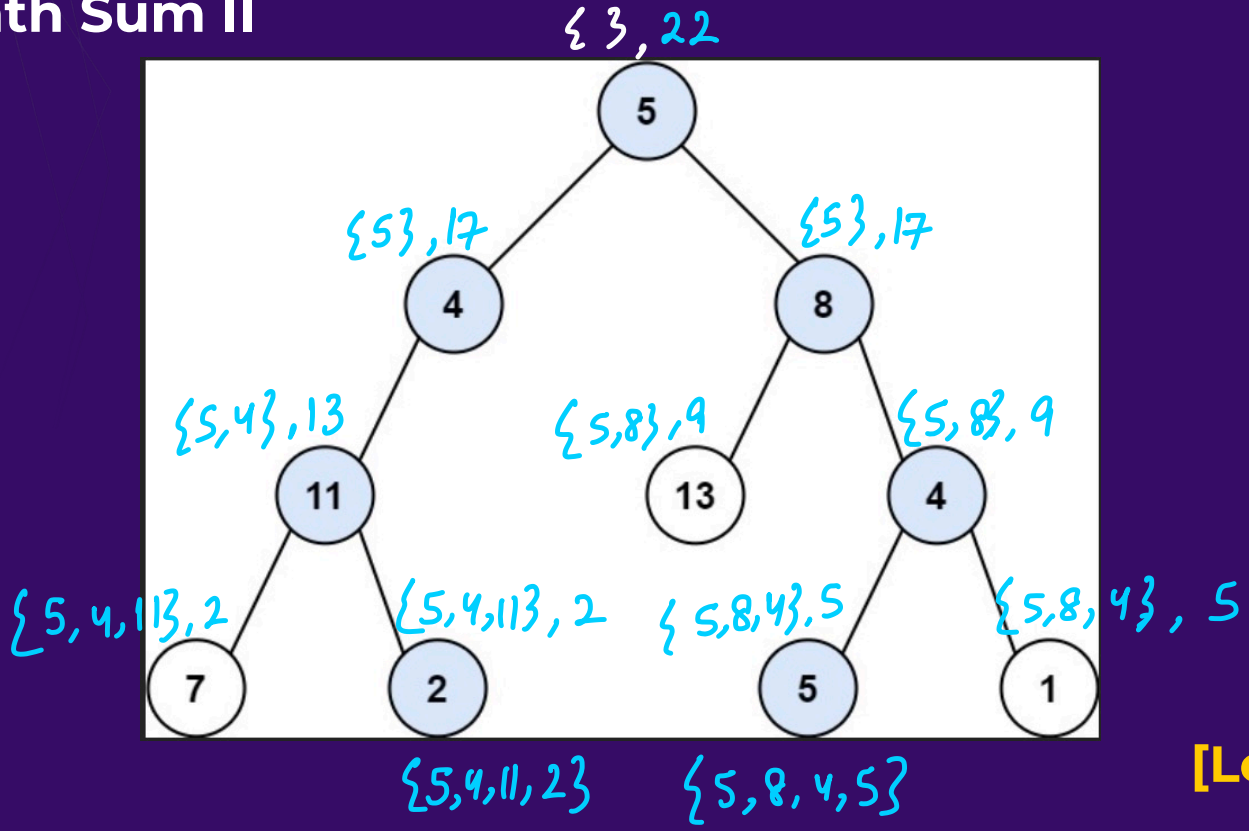
[Leetcode 543]

Ques:

ans = { {5, 4, 11, 2}, {5, 8, 4, 5} }

PW SKILLS

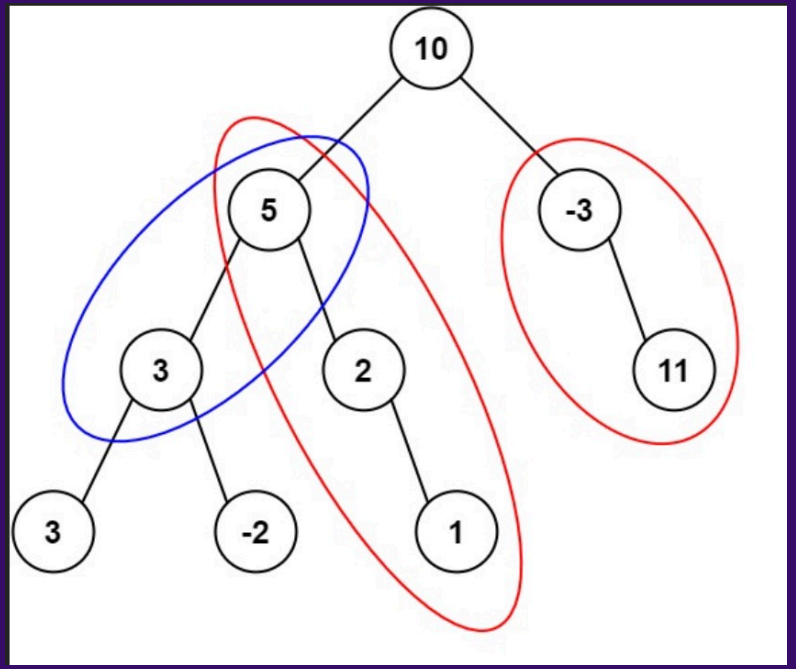
Q : Path Sum II



[Leetcode 113]

Ques:

Q : Path Sum III



target = 8

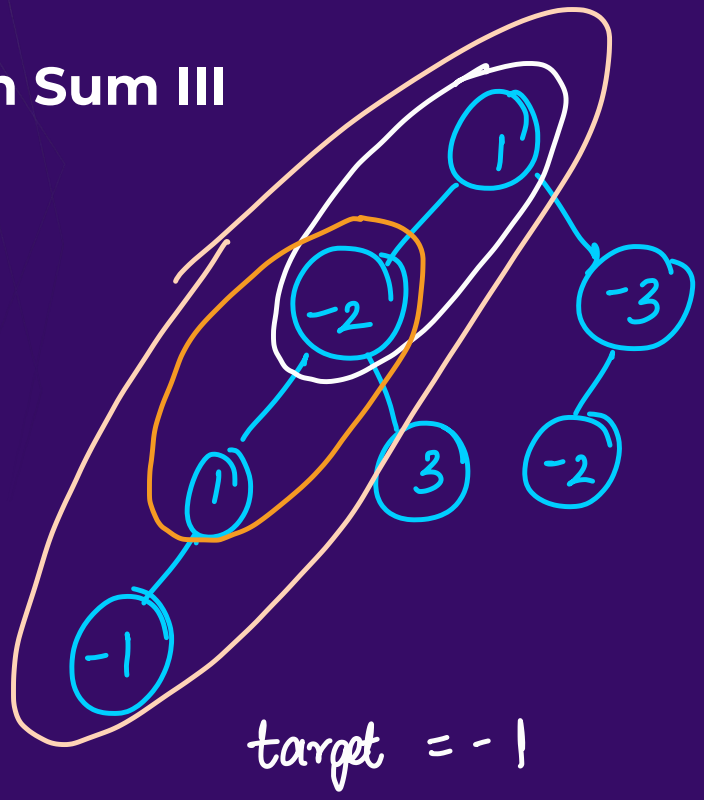
Pehle ye socho.

the path does not need to end at leaf nodes

[Leetcode 437]

Ques:

Q : Path Sum III



[Leetcode 437]

Ques:

PW SKILLS

Q : Path Sum III

```java
public int helper(TreeNode root, long sum){
    if(root==null) return 0;
    int count = 0;
    if(root.val==sum) count++;
    count += helper(root.left,sum-root.val) + helper(root.right,sum-root.val);
    return count;
}
public int pathSum(TreeNode root, int sum) {
    if(root==null) return 0;
    return helper(root,sum) + pathSum(root.left,sum) + pathSum(root.right,sum);
}
```

[Leetcode 437]

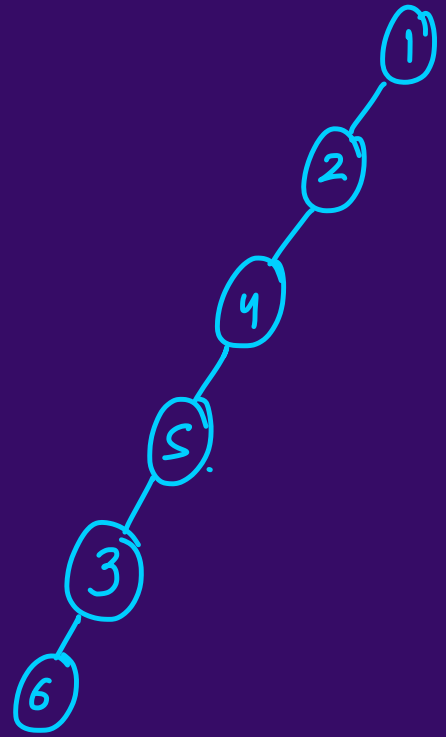
Ques:

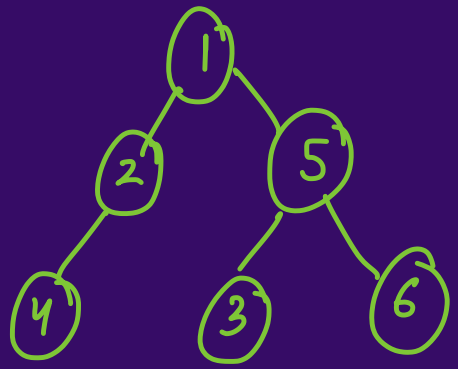
PW SKILLS

Q : Construct Binary Tree from Preorder & Inorder Traversal

Preorder

1 2 4 5 3 6



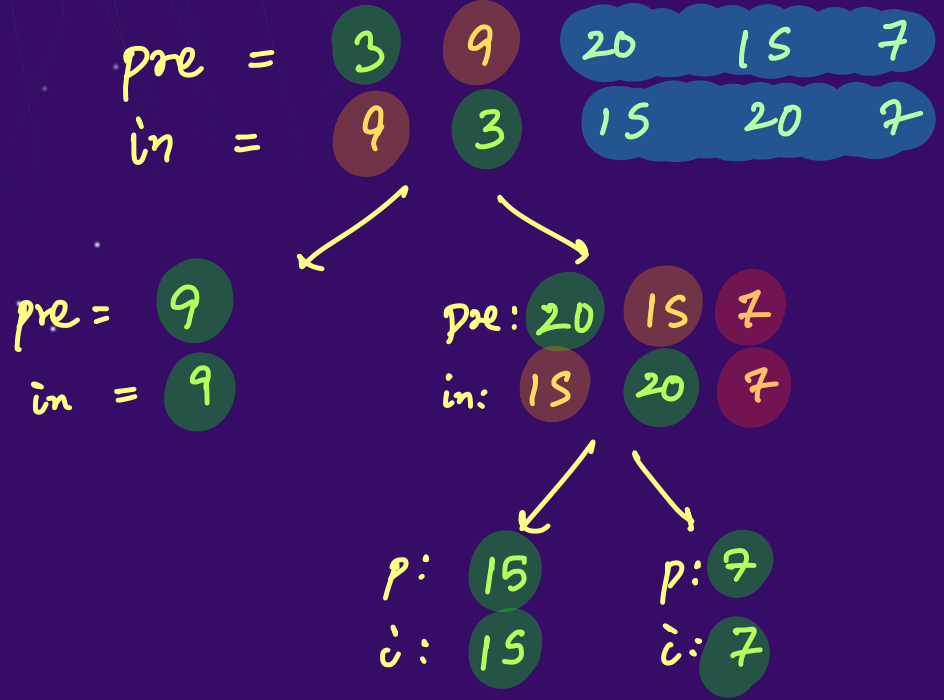


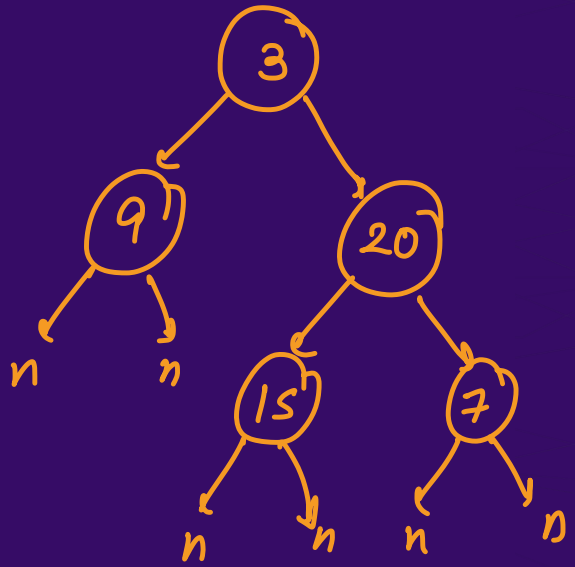
[Leetcode 105]

Ques: Think about root of the tree

PW SKILLS

Q : Construct Binary Tree from Preorder & Inorder Traversal



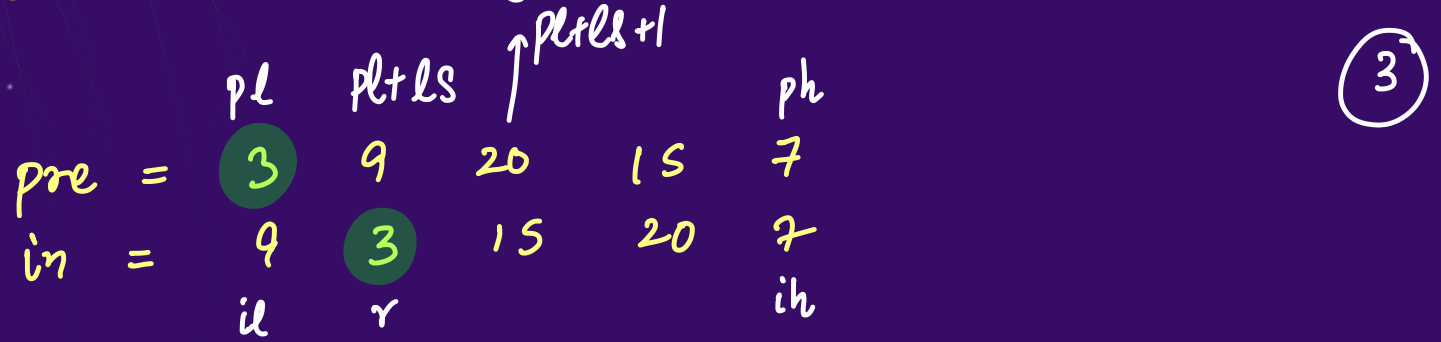


[Leetcode 105]

Ques:

PW SKILLS

Q : Construct Binary Tree from Preorder & Inorder Traversal



LST size = r - il

LST → pl+1 to pl+ls , il to r-1

RST → pl+ls+1 to ph , r+1 to ih

[Leetcode 105]

Ques:

PW SKILLS

Q : Construct Binary Tree from Preorder & Inorder Traversal

```java
public TreeNode helper(int[] preorder, int[] inorder, int prelo, int prehi, int inlo, int inhi) {
    if(prelo>prehi || inlo>inhi) return null;
    TreeNode root = new TreeNode(preorder[prelo]);
    int r = 0;
    while(inorder[r]!=preorder[prelo]) r++;
    int leftsize = r - inlo;
    root.left = helper(preorder, inorder, prelo+1, prelo+leftsize, inlo, r-1);
    root.right = helper(preorder, inorder, prelo+leftsize+1, prehi, r+1, inhi);
    return root;
}
public TreeNode buildTree(int[] preorder, int[] inorder) {
    int n = preorder.length;
    return helper(preorder,inorder,0,n-1,0,n-1);
}
```

[Leetcode 105]

Homework:

PW SKILLS

Q : Construct Binary Tree from Inorder & Postorder Traversal

[Leetcode 106]

THANK YOU