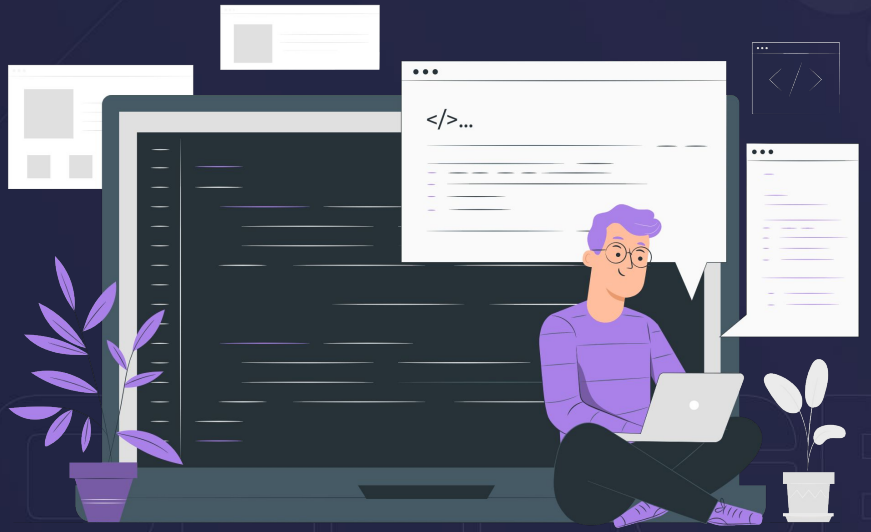
PW SKILLS

PW SKILLS

Lecture – 52

Binary Trees

[Basics]



Prerequisites

- Recursion → must
- Linked List

↓

trees will become easy

```
void pip(int n){
    Kaam → pre
    pip(n-1);
    Kaam → in
    pip(n-1);
    Kaam → post
}
```

3

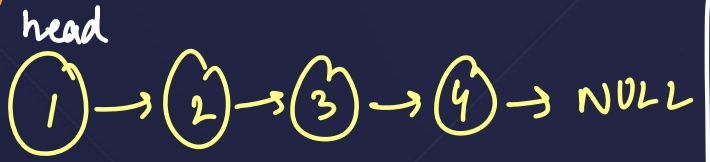
Today's Checklist

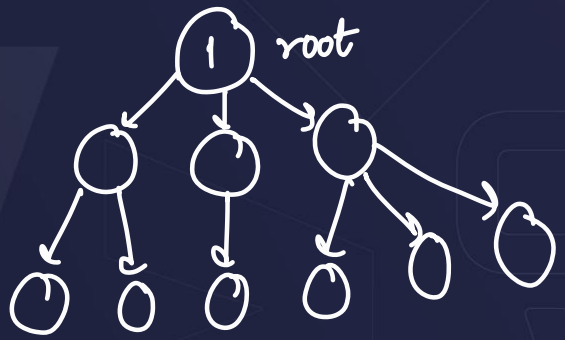
- What is a Tree data structure
- Representation
- Terminology
- Important properties of trees
- Types of Trees
- Applications of tree data structure
- What is a Binary Tree?
- Implementation
- Traversals → Next Lecture
- Problems
- Types of Binary Trees

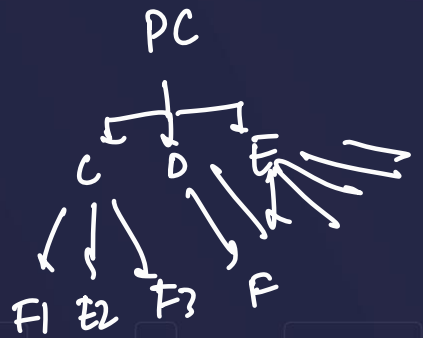
What is a Tree data structure

Array , LL, stack, queues → Linear Data Structure

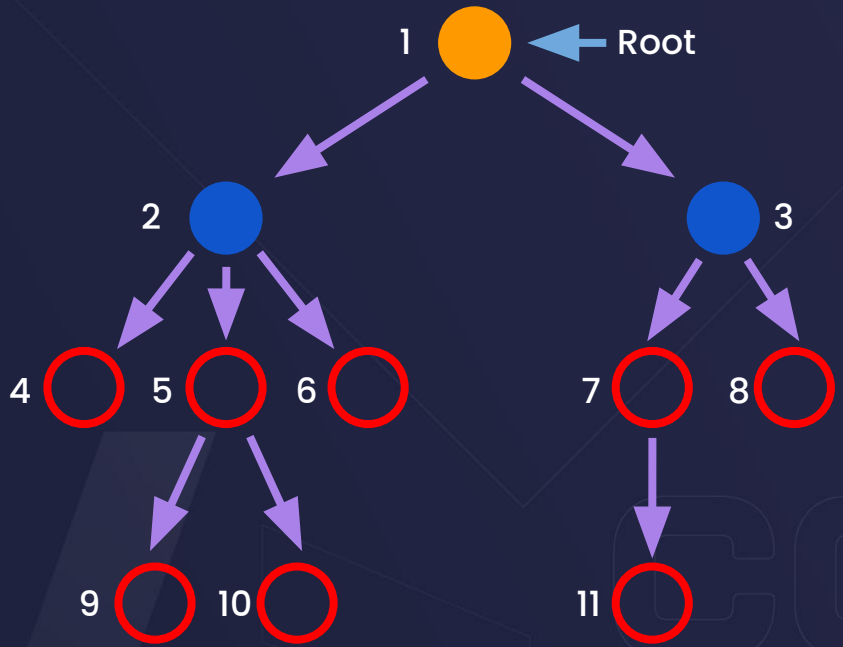
Trees, graphs → non linear D.S.





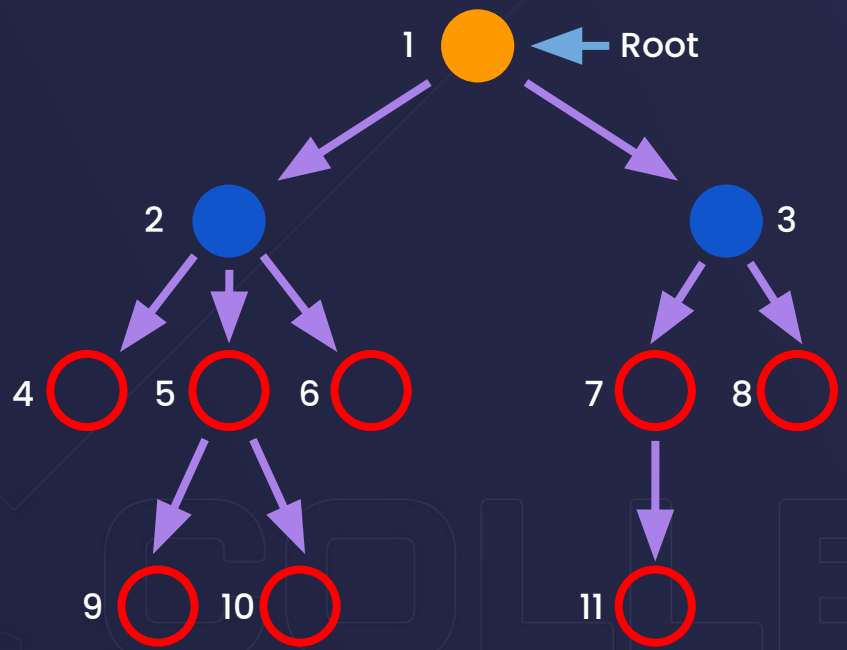


Representation



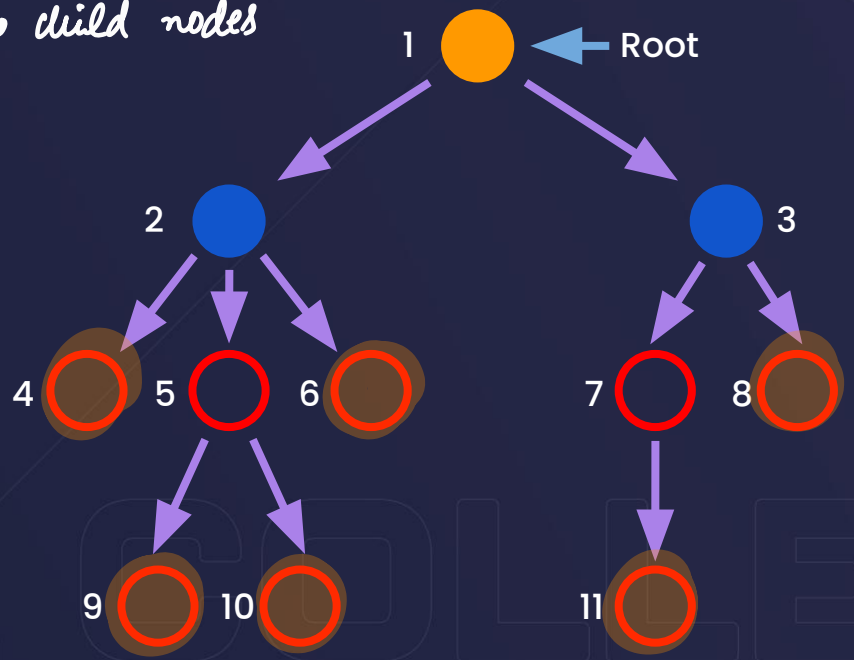
Terminology

1. Root
2. Child Node
3. Parent Node
4. Sibling Nodes



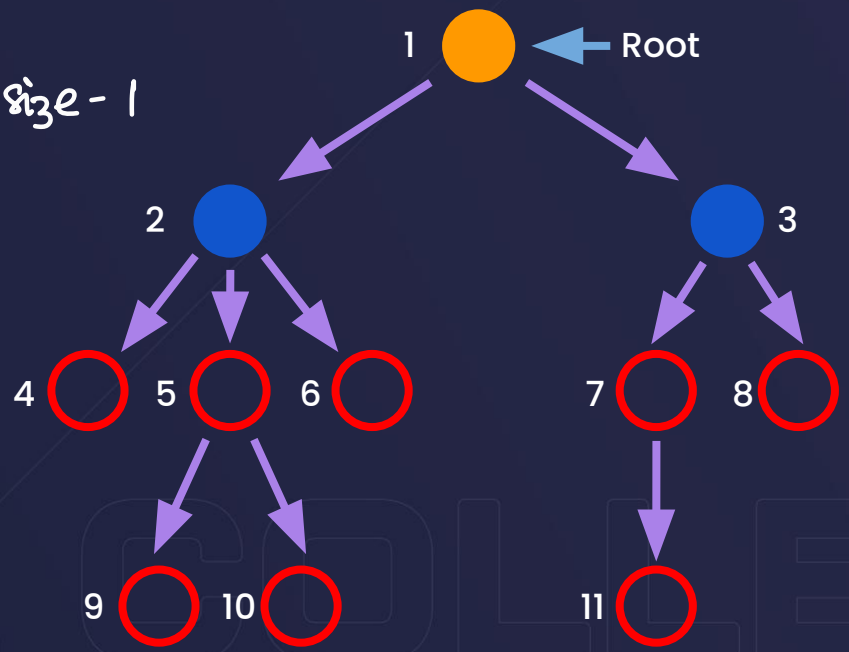
Terminology

5. Leaf Node → with no child nodes
6. Internal Node
7. Ancestor Node
8. Descendant Node



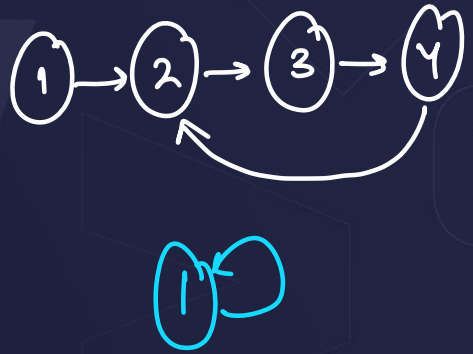
Terminology

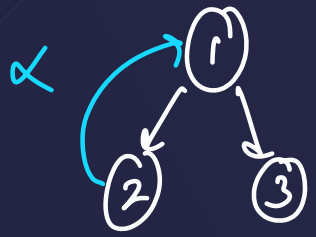
9. Level
10. Number of edges = size - 1
11. Height = level - 1
12. Size = no. of nodes

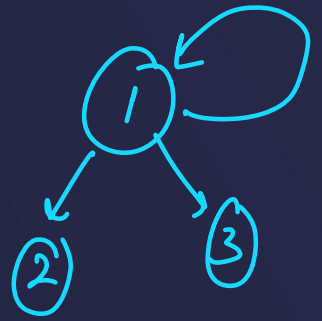


Important Properties of trees

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

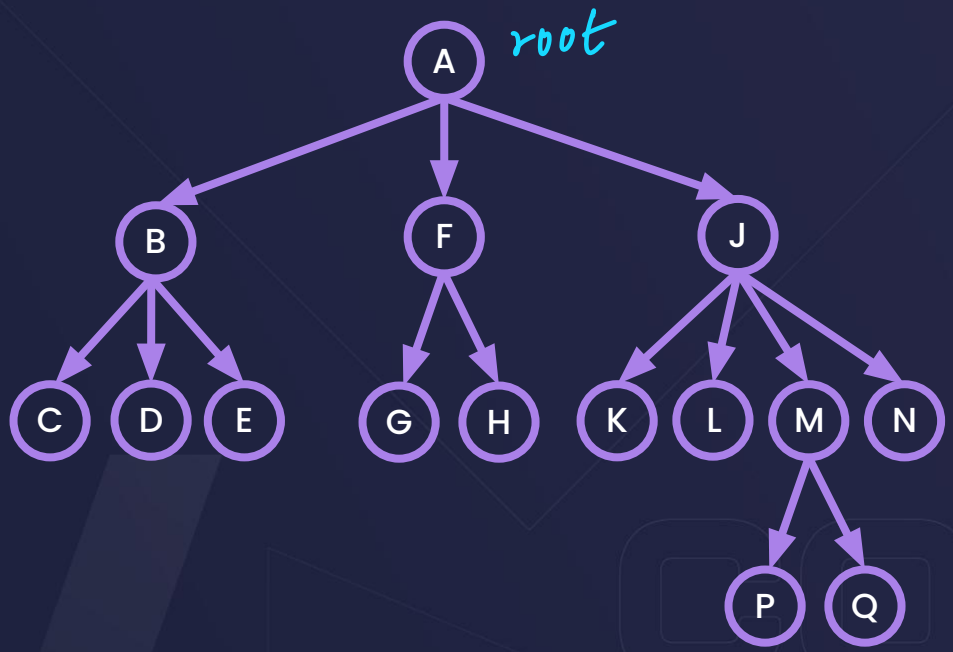




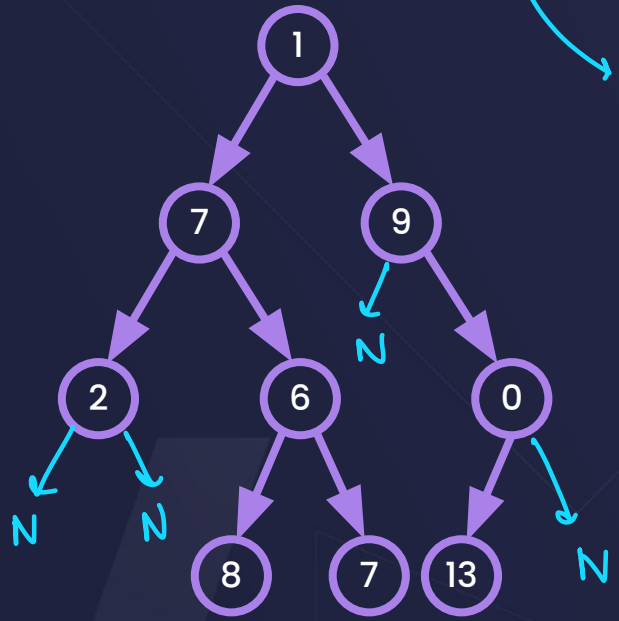


Types of Trees

1. Generic Trees → multiple children

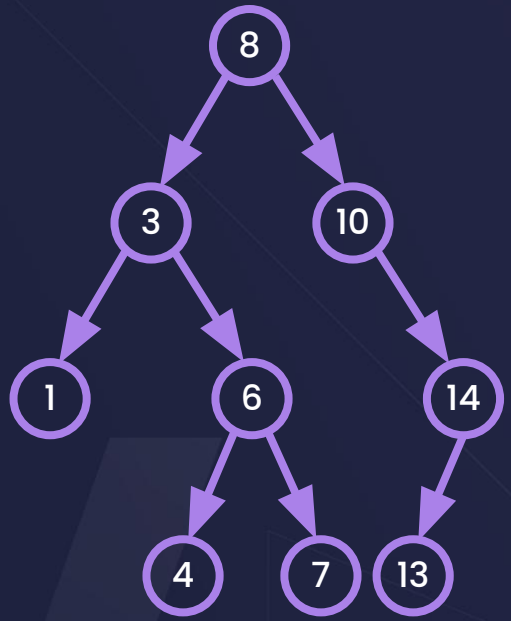


2. Binary Trees → 0, 1, 2 children [upto 2 child nodes]



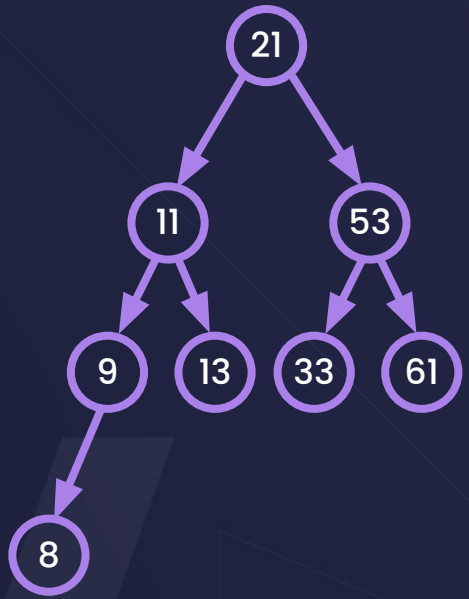
→ Every node has a value, & its left child's address, right child's address

3. Binary Search Trees : Binary trees with some special property



Baad me

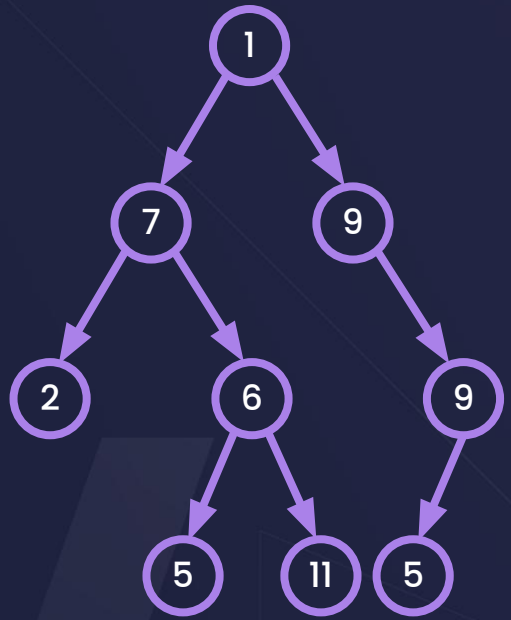
4. AVL Trees : Balanced BST's



Applications of tree Data Structure

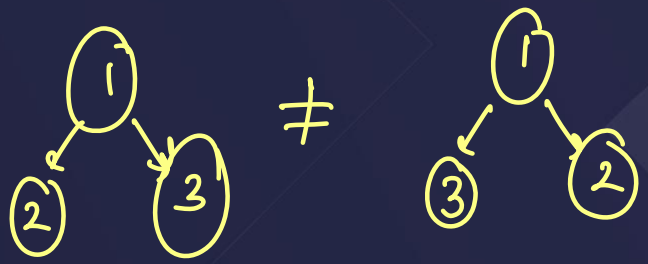
1. Hierarchical data structure
2. Searching efficiency
3. Sorting
4. Dynamic Data
5. Efficient Insertion and Deletion
6. Easy to implement

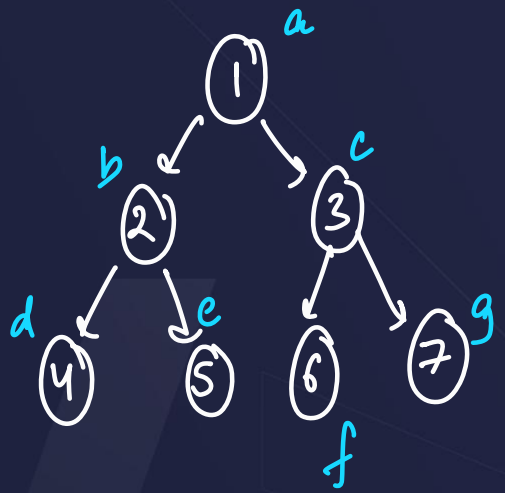
What are Binary Trees? → 0, 1, 2 child nodes



Implementation

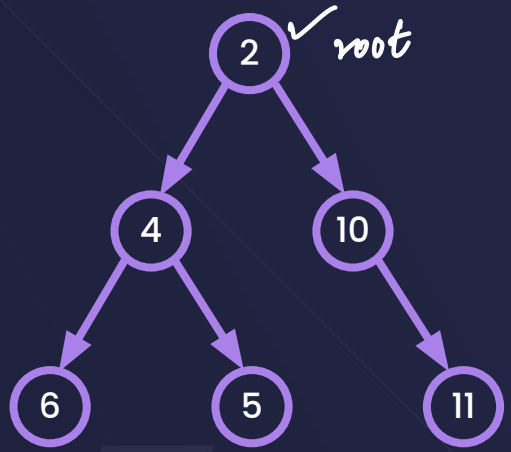
Creating a Node class





1 2 4 5 3 6 7

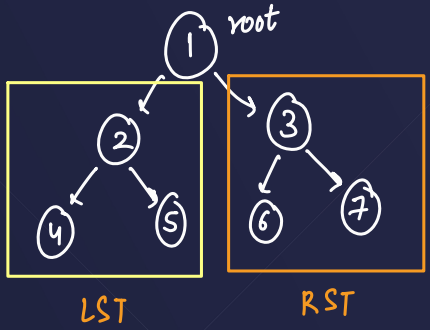
Display (root of tree) (just like in LL → head only)



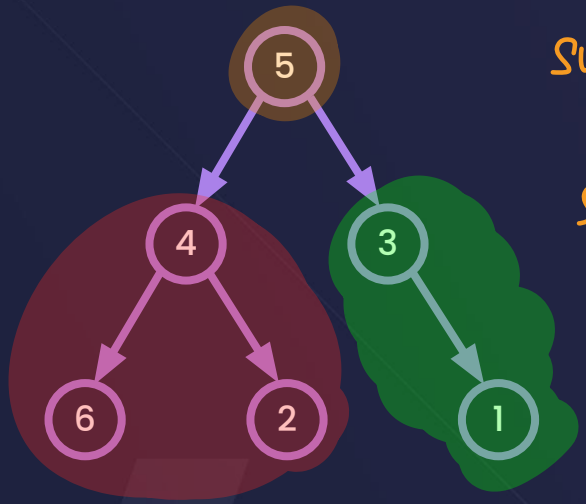

```
display (head) {
    if (root == NULL) —
    cout << head -> val;
    display (head -> next);
}
```

3

Left - Subtree & Right - Subtree



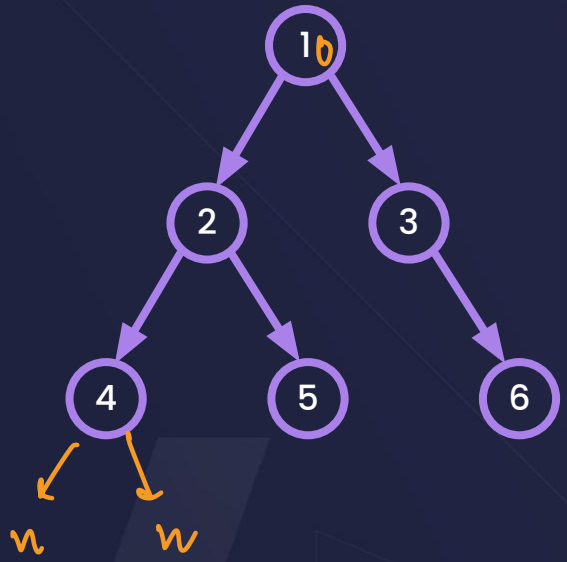
Find sum of tree nodes



Sum = root→val + LST + RST

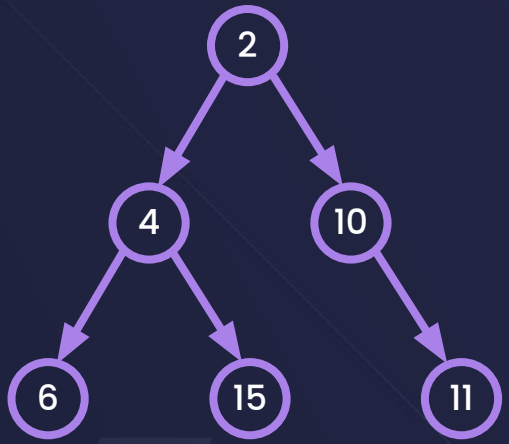
Sum(root) = root→val + Sum(root→left)
+ Sum(root→right);

Find size of Binary Tree → Try it yourself first



Size(root) = 1 + size(root→left) + size(root→right)

Find node with max value



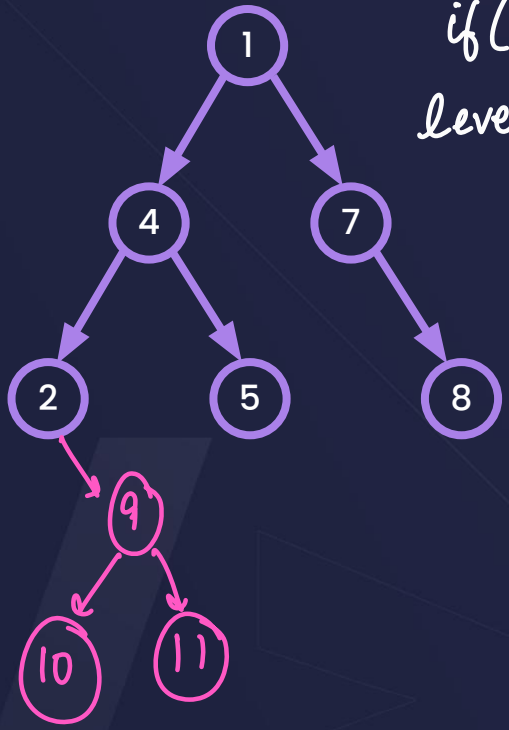

```
if (root == NULL) return INT_MIN;
max(root) = max(root->val, max(left), max(right))
```

a, b, c

```
max(a, max(b,c));
```

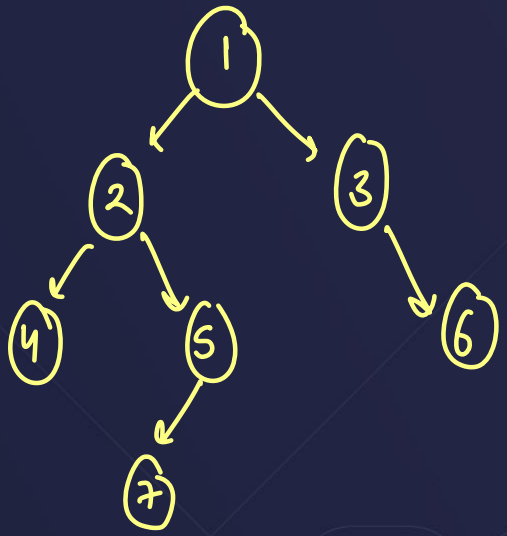
no. of levels

Find ~~height~~ of Binary Tree




```
if (root == NULL) return 0;
levels(root) = 1 + max(levels(root->left), levels(root->right))
```

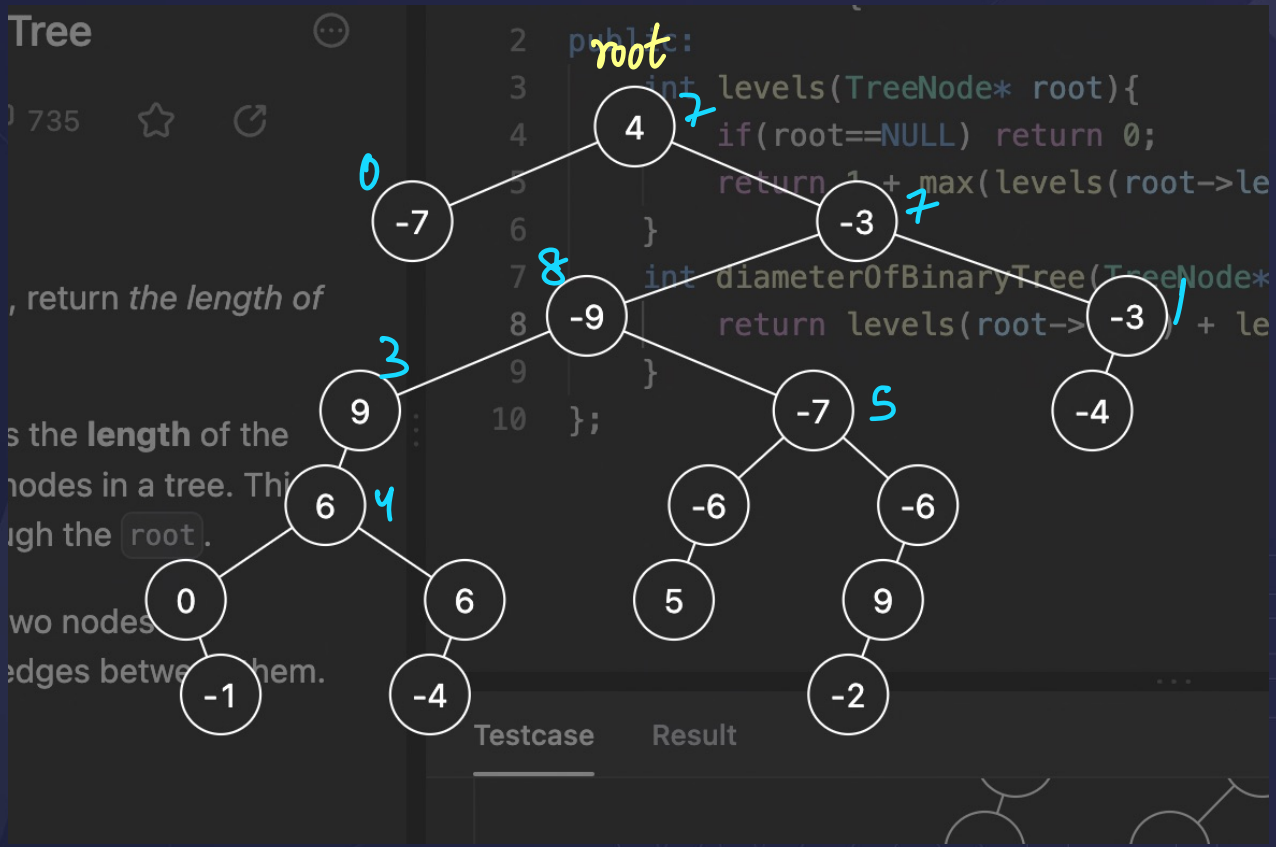
Ques: Diameter of Binary Tree [LeetCode 543]



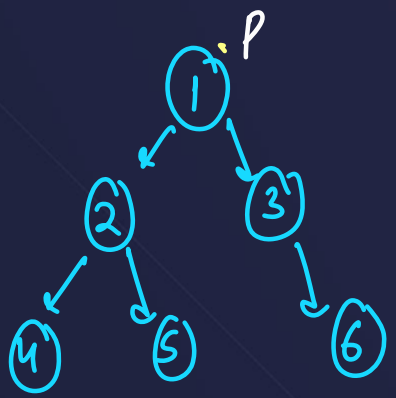
Approach 1: Diameter (longest path b/w any 2 nodes)

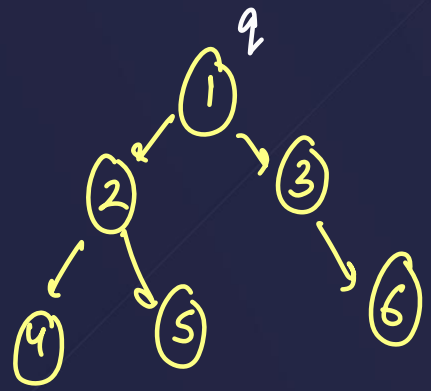
= no. of levels in LST + levels (RST)

Ques: Diameter of Binary Tree [LeetCode 543]



Ques: Same Tree [LeetCode 100]

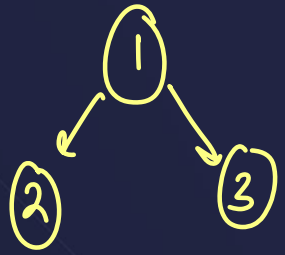


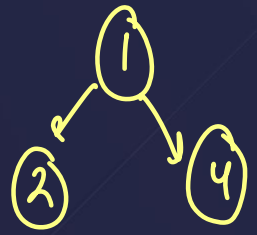


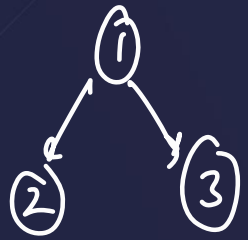
1) if (p != q) return false;
2) bool LSTans = isSameTree(p->left, q->left);
3) bool RSTans = isSameTree(p->right, q->right);
4) return true

Ques: Same Tree

[LeetCode 100]







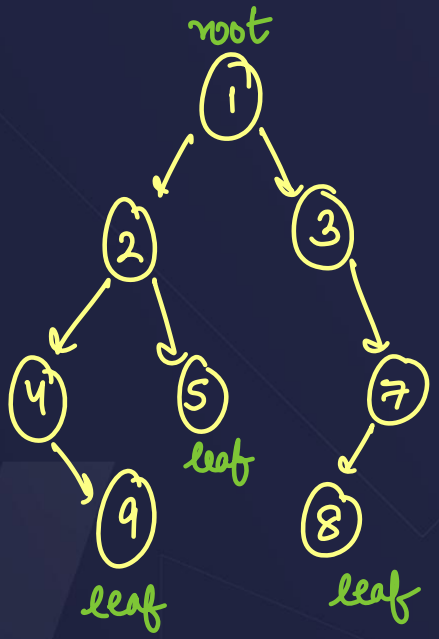
Ques: Invert Binary Tree

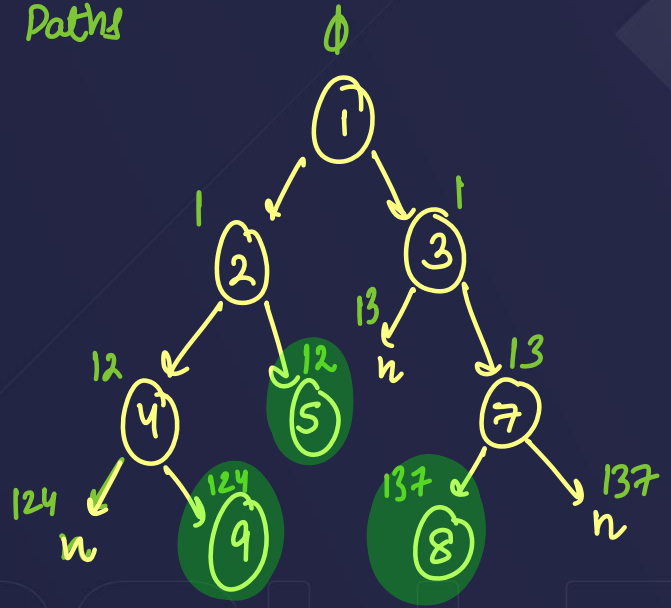
[LeetCode 226]

Ques: Binary Tree Paths [LeetCode 257]

ϕ = ""

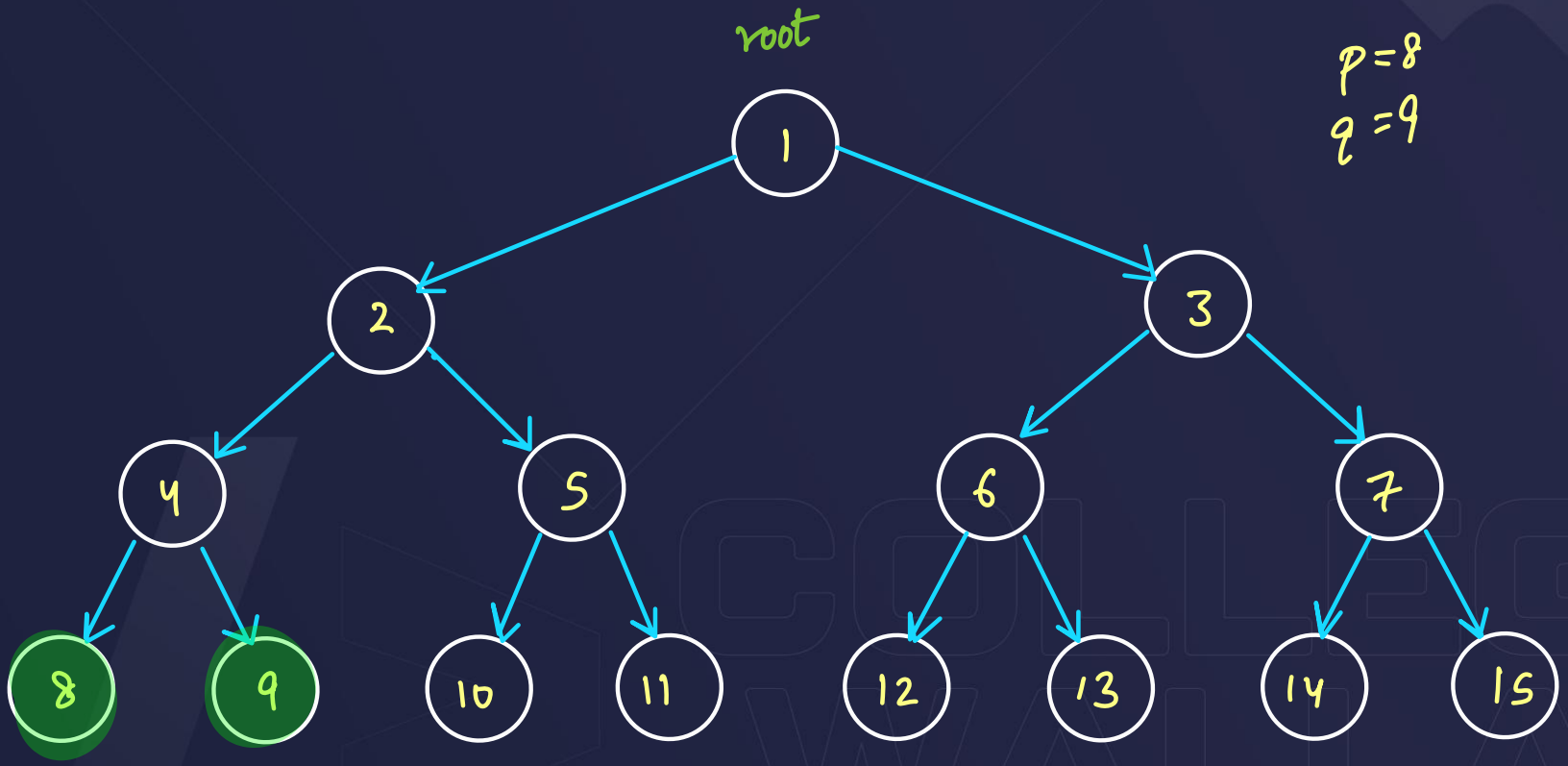
Root to Leaf Paths





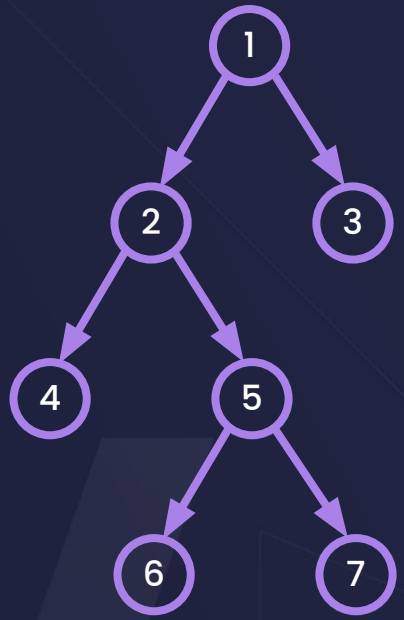
→ V { "1249", "125", "1378" }

Ques: Lowest Common Ancestor of a Binary Tree (LCA) [LeetCode 236]

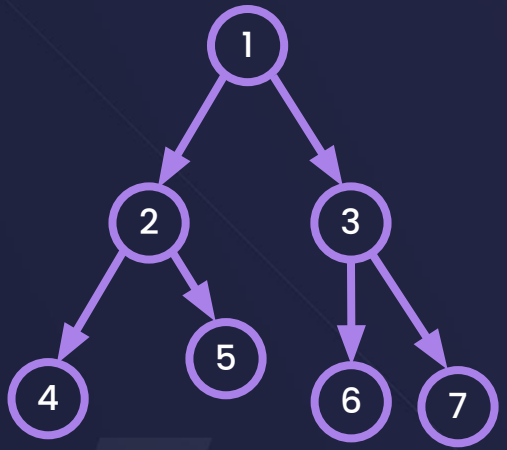


Types of Binary Trees

1. Full Binary Tree (0 or 2)



2. Perfect Binary Tree

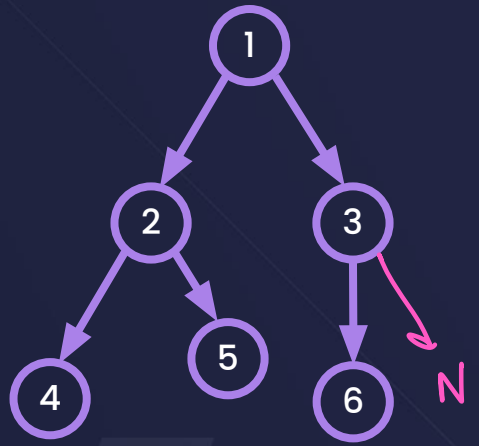


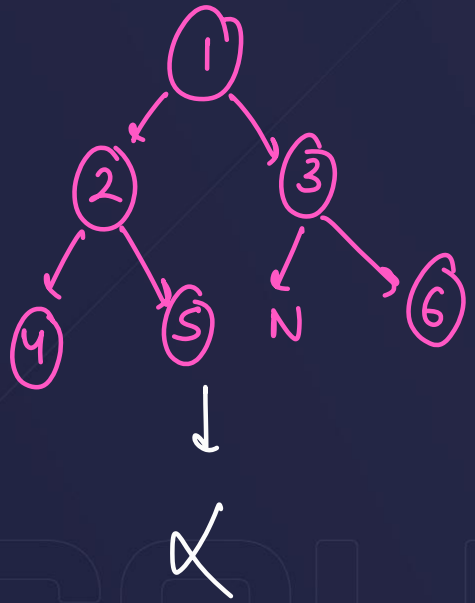
↓

Every Node (except the last level leaf nodes) has 2 child nodes

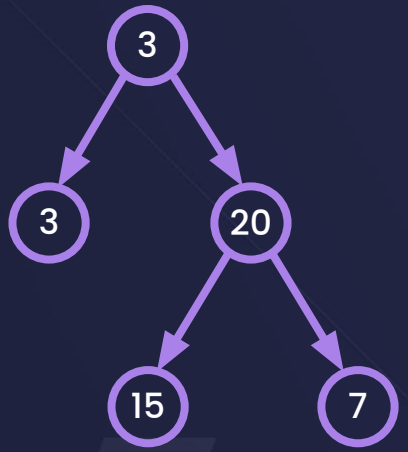
the last level has 0 child

3. Complete Binary Tree



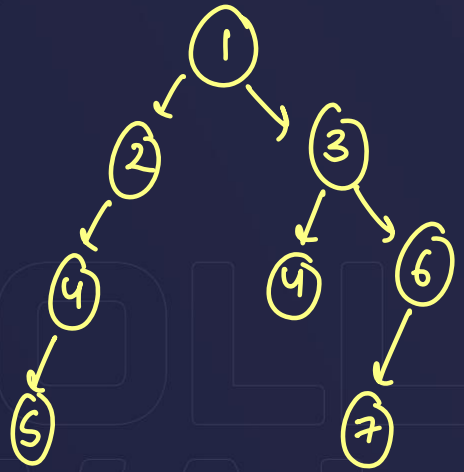


4. Balanced Binary Tree : "Important"

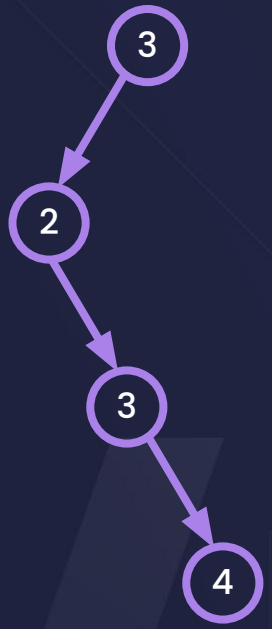


↓

For every node, the difference b/w the levels of LST & RST should be atmost 1.



5. Degenerate and Skewed Binary Trees



0 or 1 child

Summary

- In this lecture , we studied about the Tree data structure.
- We studied its representation.
- We studied about its terminology.
- We studied about various types of Trees.
- We studied about Binary trees
- We studied about its implementation.
- We studied the various traversals on binary trees.
- We solved some problems on binary trees.
- We studied about the various types of Binary trees.

Next Lecture

- ~~Interview problems on binary trees~~

  Traversals in Trees

SKILLS

THANK YOU