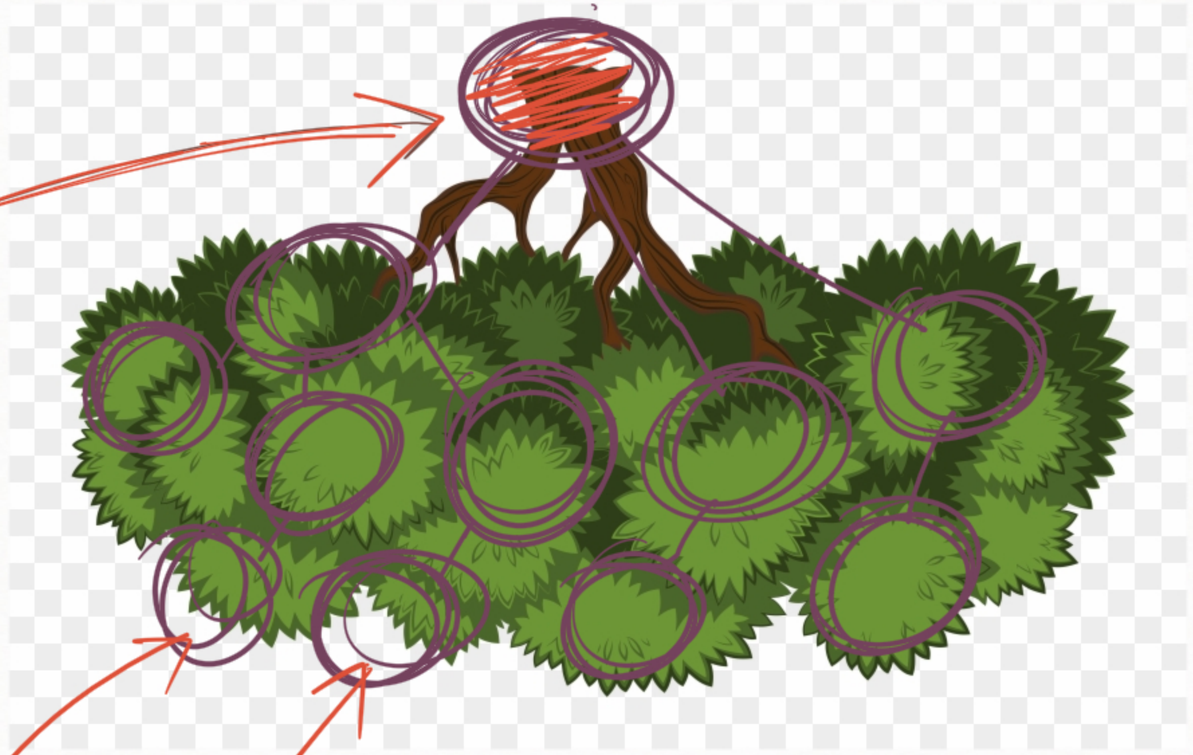
Actual DSA Tree

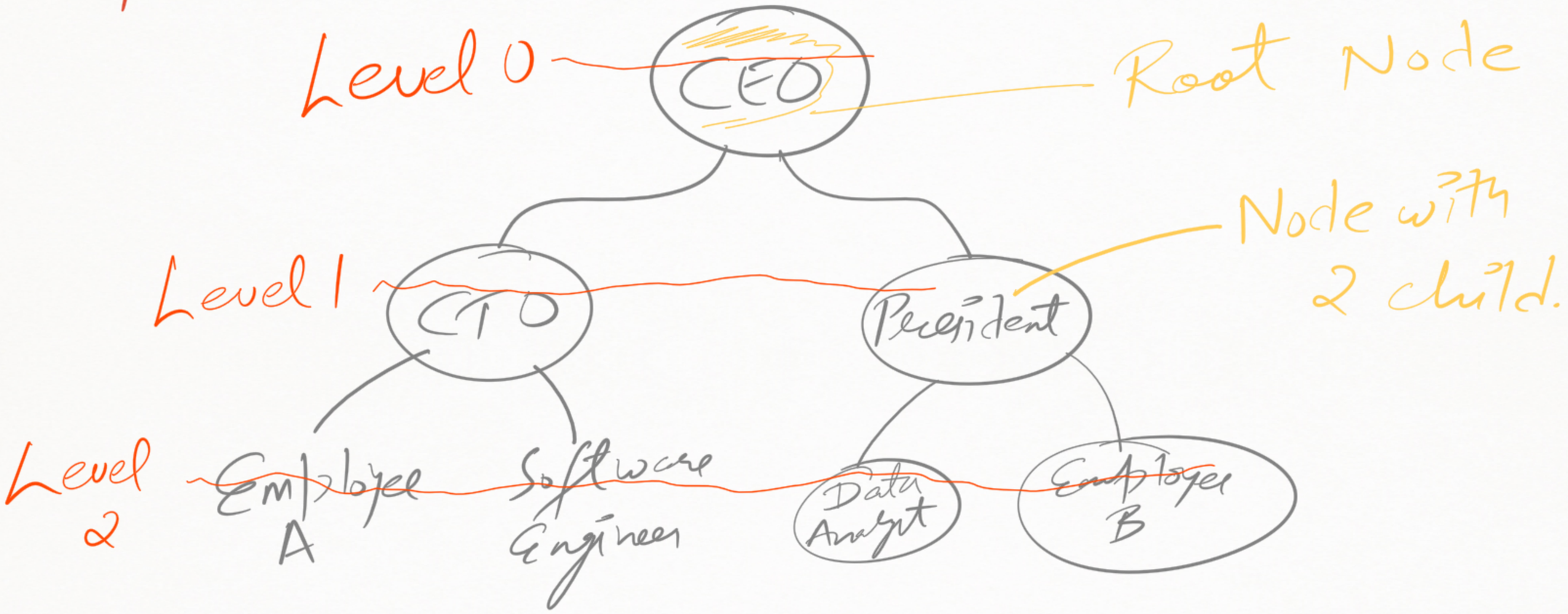


Root Nodes

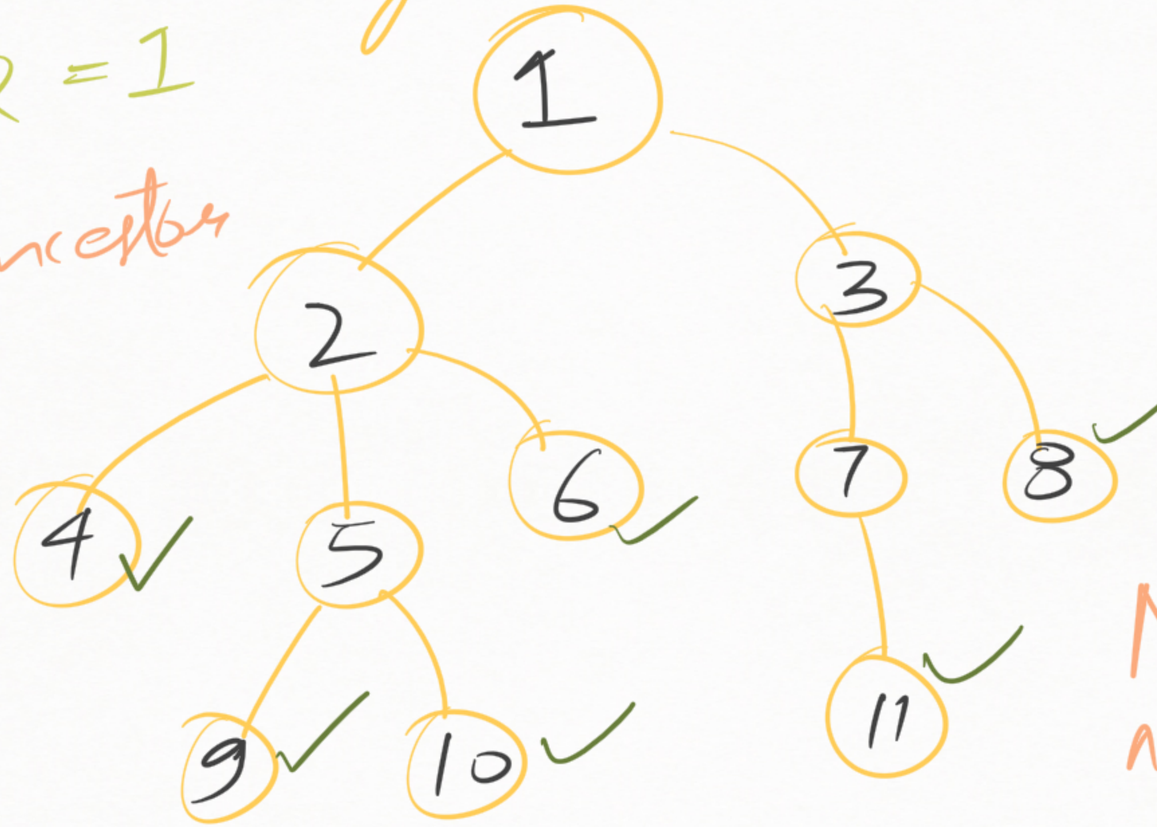
Root is at the top

Leaf Nodes

To Represent Hierarchical Data



Family Tree Relationships



Parent of 2 = 1

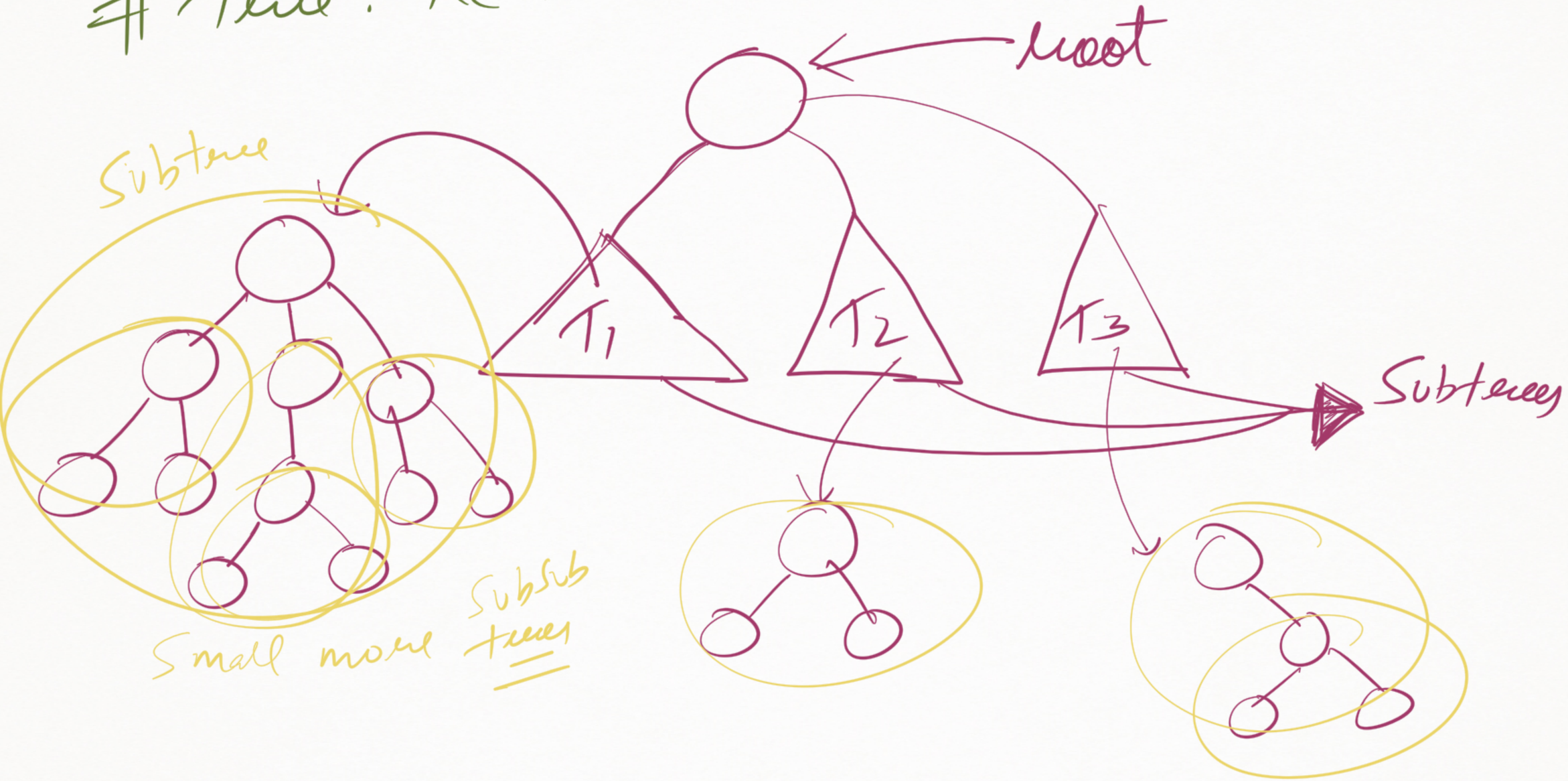
Common Ancestor of 10 & 4 => 2

Siblings of 6 => 4 & 5

Cousins of 6 => 7 & 8

No child => Leaf node => 4, 9, 10, 6, 11

Tree: Recursive Data Structure



N Nodes

N-1 Edges

1 Node

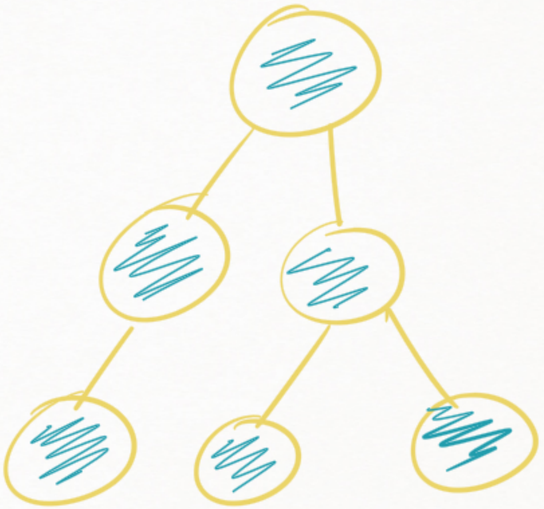
0 edge

2 Nodes

1 edge

3 Nodes

2 edges



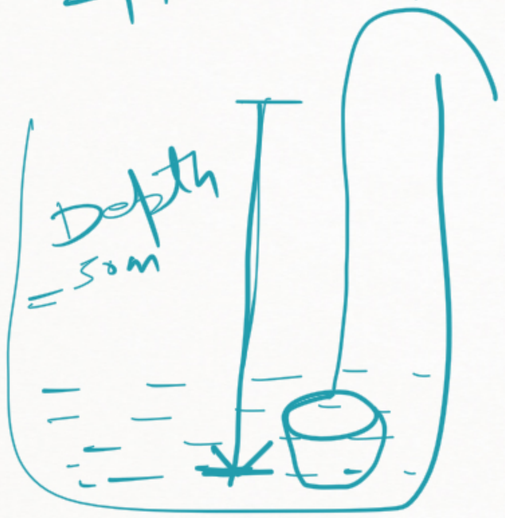
6 Nodes

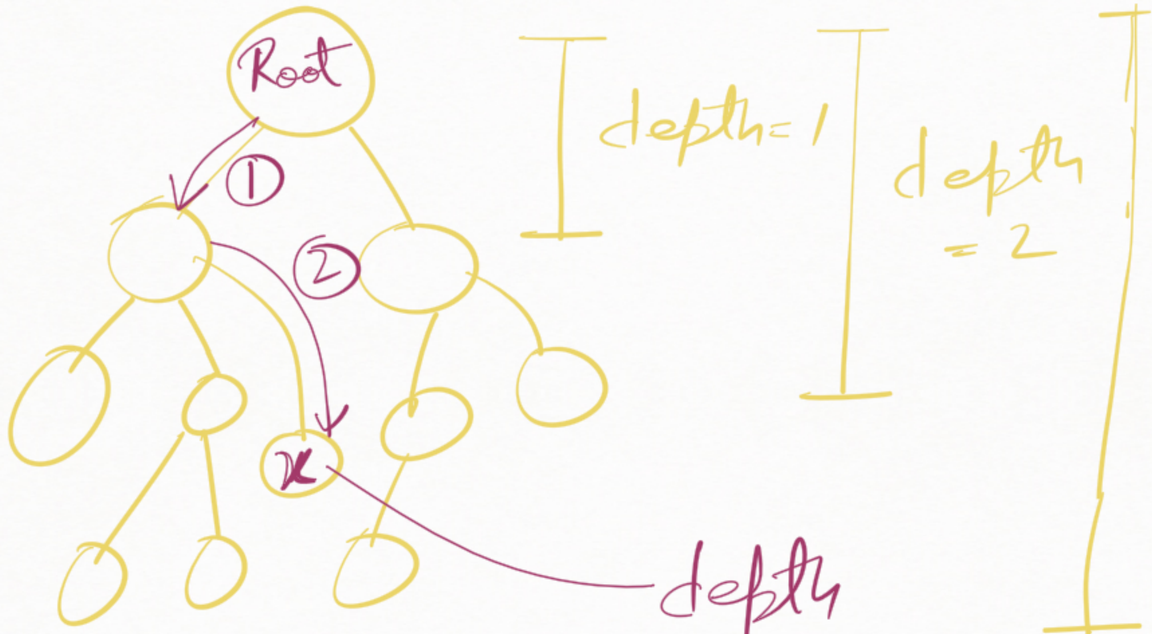
5 edges

edges = nodes - 1

Depth of x: Length of (Path) from root to x

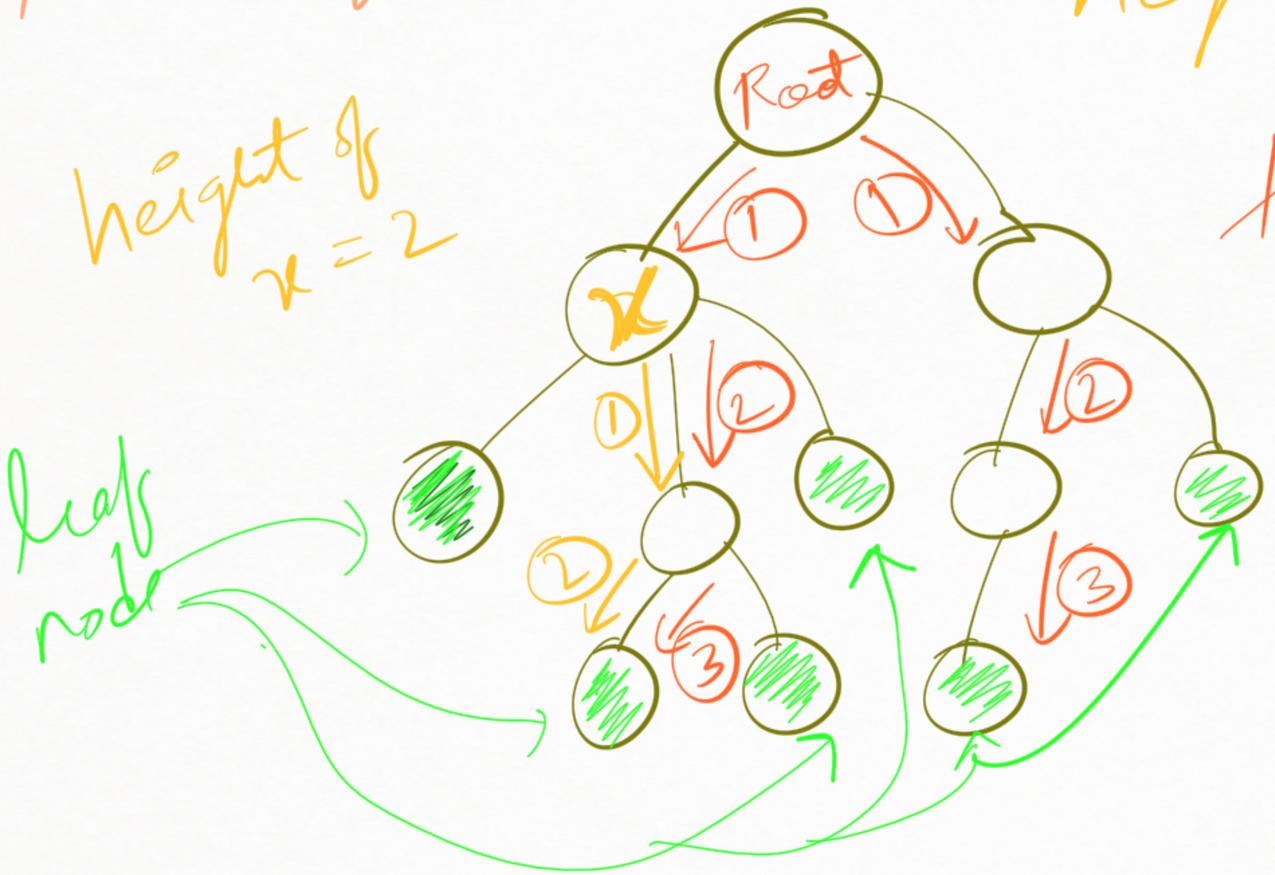
edges





Height of x : No of edges in longest path from x to a leaf

height of leaf node = 0

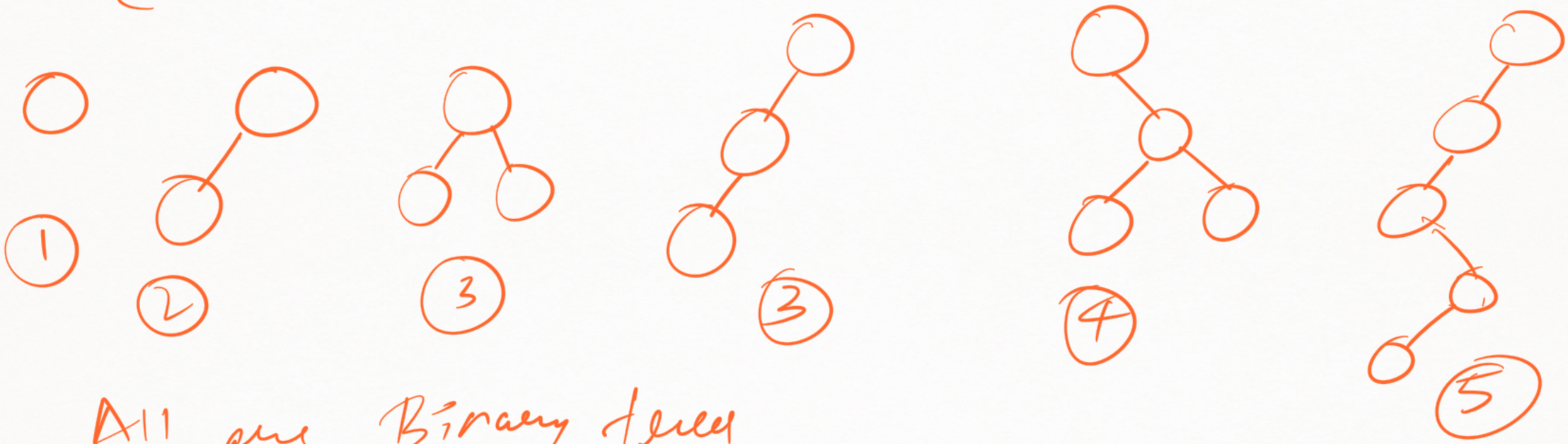


height of x = 2

Height of Tree = height of root = 3

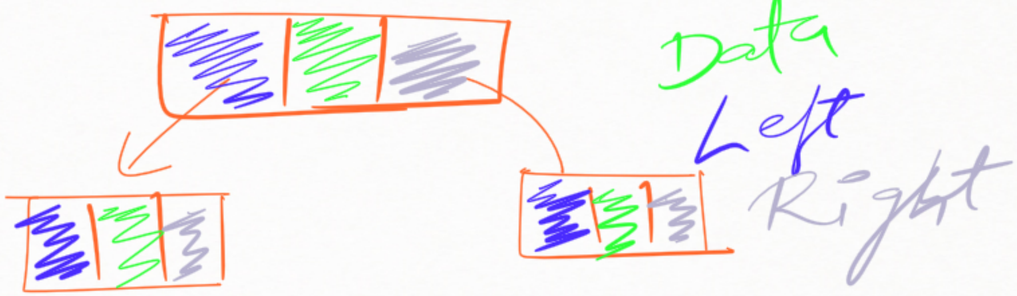
Types of Trees: Binary trees.

Each node must have atmost 2 children



All are Binary trees

Node Implementation { for Binary Trees }

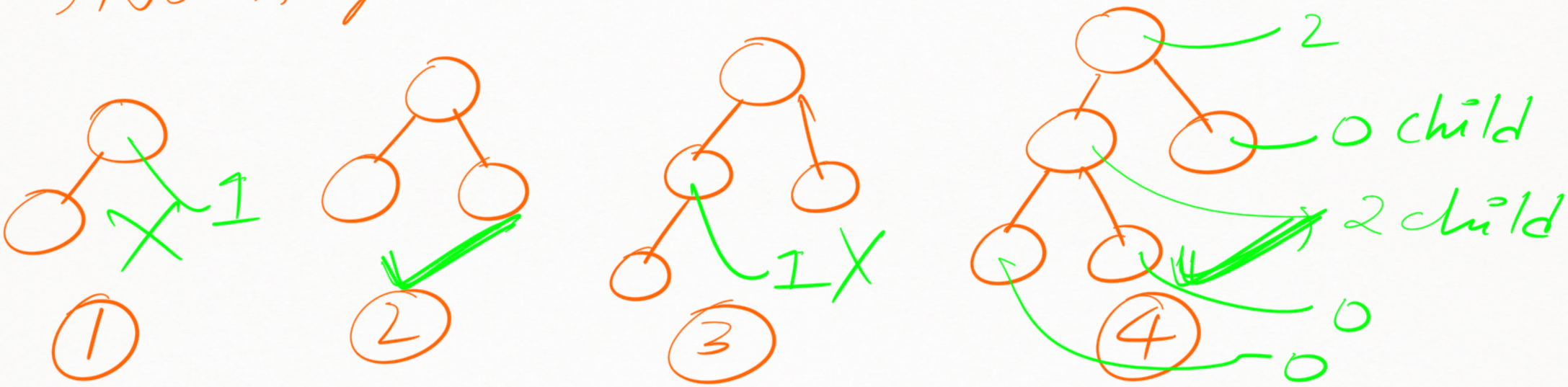



```
class node
{   init / constructor (self, data)        only for Py.
        self/this.data = data
        self/this.left = left
        self/this.right = right
}
```

Strict/Proper Binary Tree

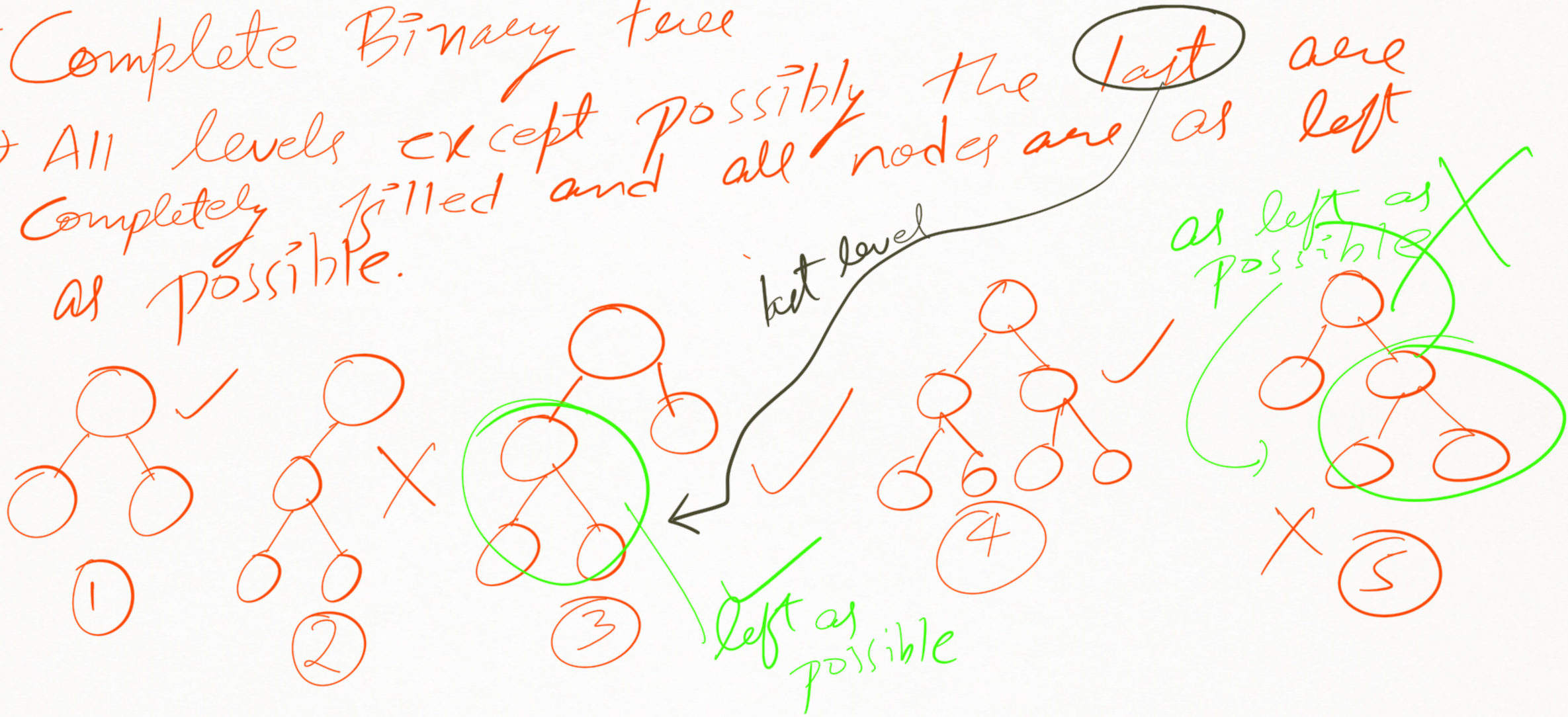
→ either 0 or 2 children

→ No single child allowed



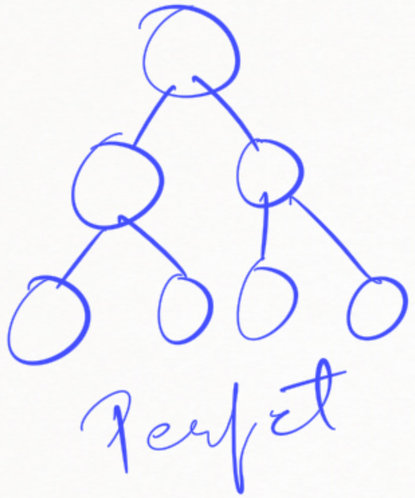
Complete Binary tree

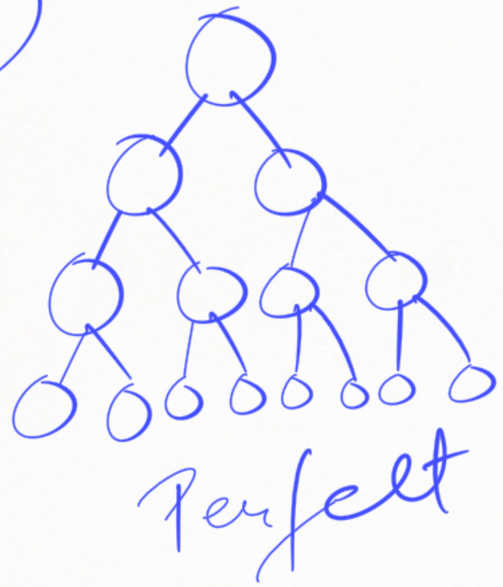
↳ All levels except possibly the last are completely filled and all nodes are as left as possible.

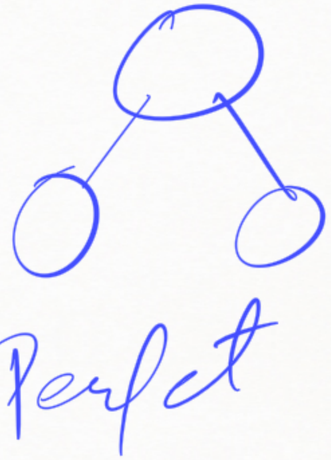


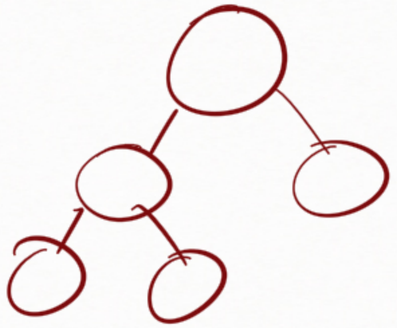
Perfect Binary Tree

→ ≠ all levels are completely filled









Not Perfect

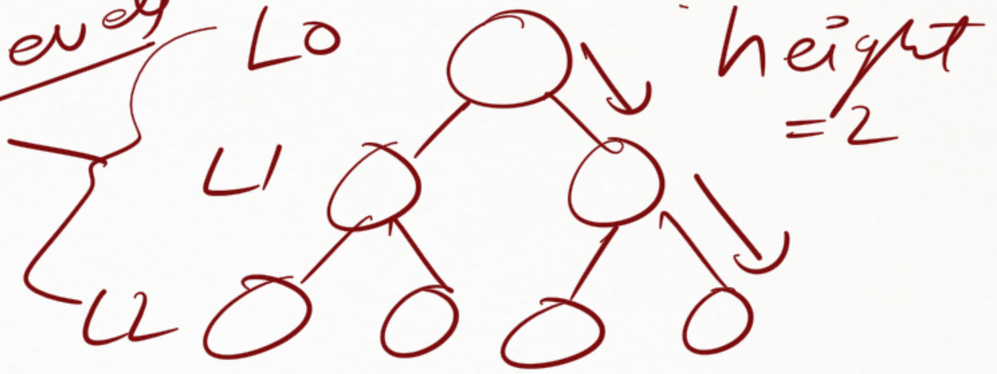
Max No. of nodes in a Binary Search tree with height h

$$= 2^0 + 2^1 + 2^2 + \cdots\cdots 2^h$$

$$= 2^{h+1} - 1$$

$$= 2^{\text{no of levels}} - 1$$

3 levels



N nodes, what is the height of perfect Binary Tree

$$n = 2^{h+1} - 1$$

$$n+1 = 2^{h+1}$$

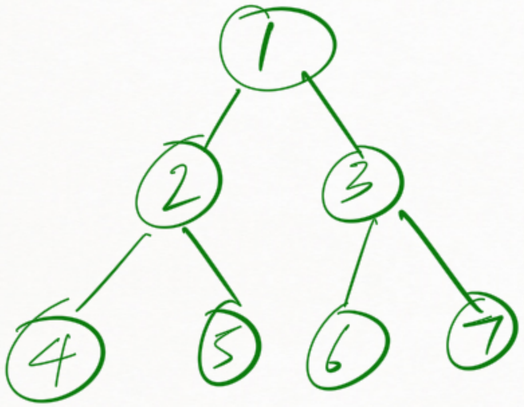
$$\log_2(n+1) = \log_2 2^{h+1}$$

$$h+1 = \log(n+1)$$

$$\boxed{h = \log_2(n+1) - 1}$$

Time Complexity = O(height)

Best case



O(height)

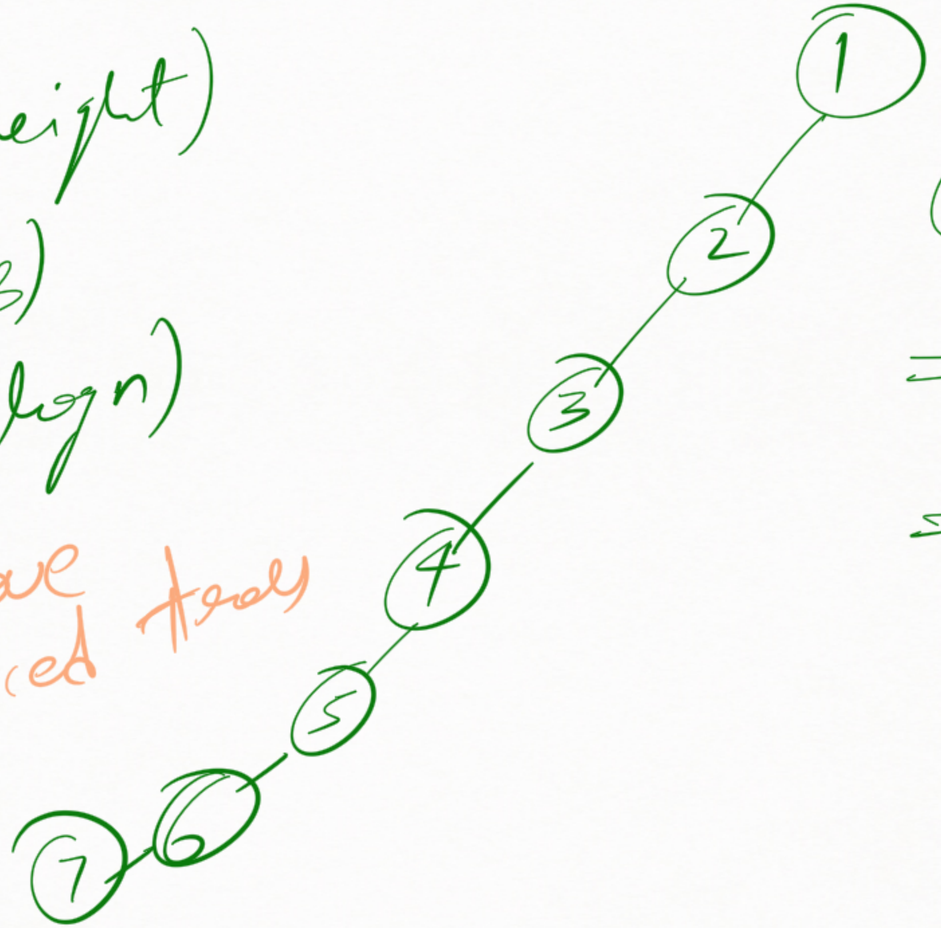
O(3)

O(log n)

Perfect Tree

This is why we love Balanced trees

Worst case



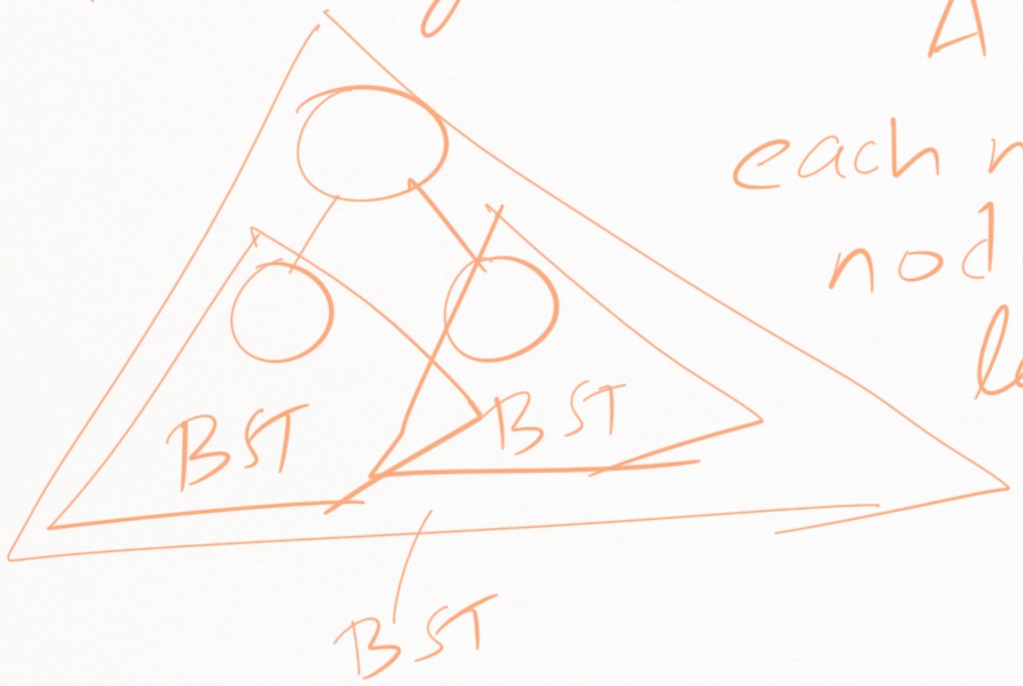
O(height)

= O(7)

= O(n)

Binary Search Trees

A binary tree in which for each node, value of all the nodes in left subtree is lesser or equal, value of all nodes in right subtree is greater.



Search → O(log n)