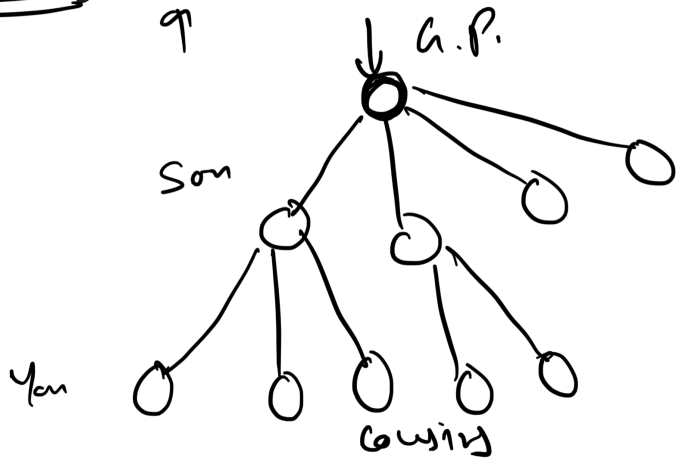
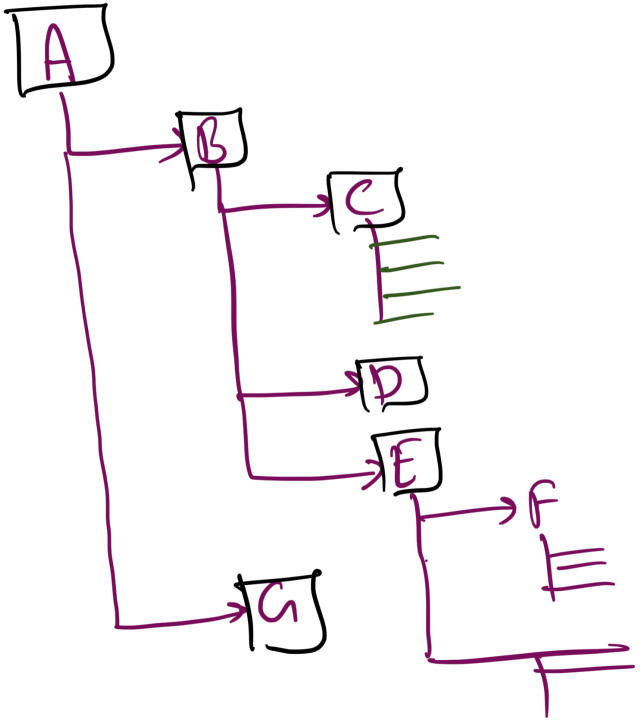
Binary Tree Basics

Binary Tree Data Structure





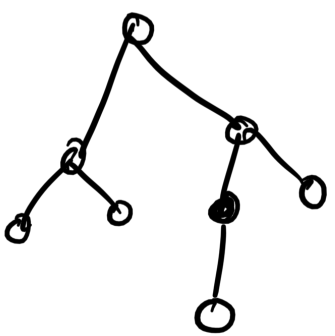
→ Root Node.

Parent ↔ child , Leaf Node

Binary Tree: Every Node has at most 2 child nodes

No closed loop inside a Tree.

Non-linear data structure.



Leaf Nodes : Do Not have children

Node class

- Data
- Node left, right

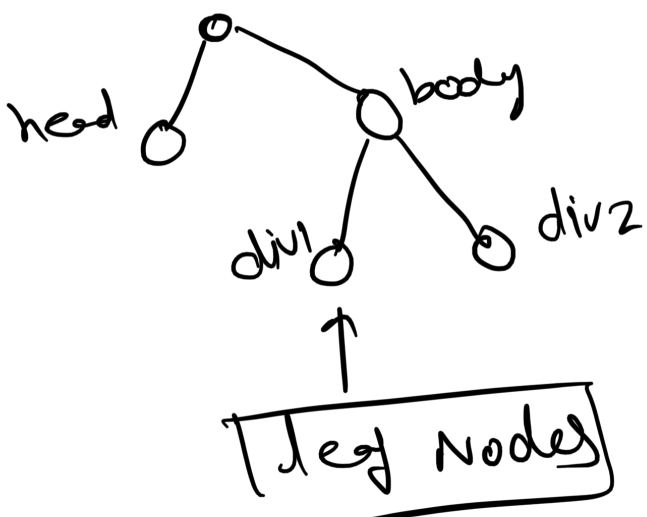
Node

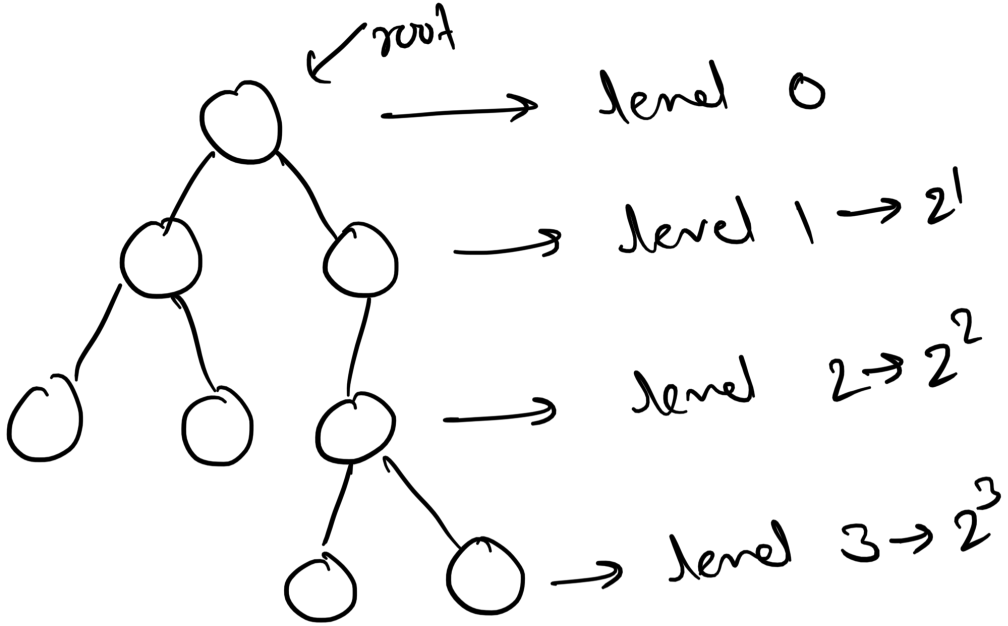
- data
- next

```
<html>
    <head>

    </head>
    <body>
        <div>
            <div>

            </div>
        </div>
    </body>
</html>
```





k

Maximum number of Nodes at any level 'k'

2^k

Maximum number of nodes in a binary tree with height h.

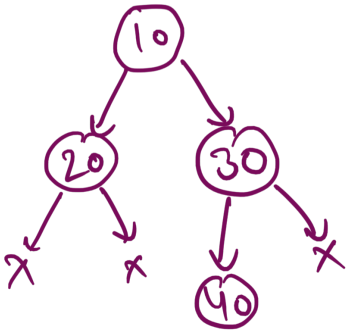
$2^h - 1$

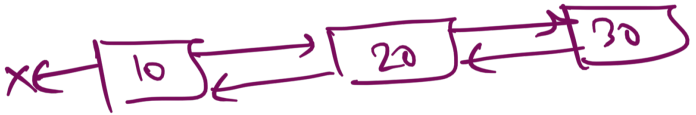
$2^0 + 2^1 + 2^2 + \cdots 2^k = 2^{k+1}$

Creation of a Binary Tree

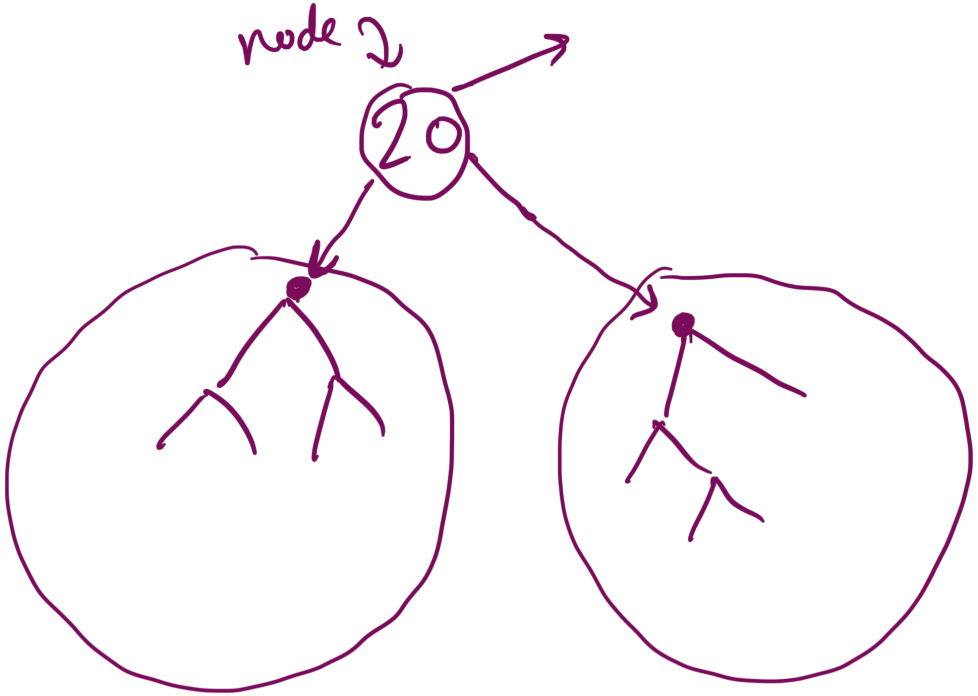
```
class Node {
    int data;
    Node left, right;
}
```

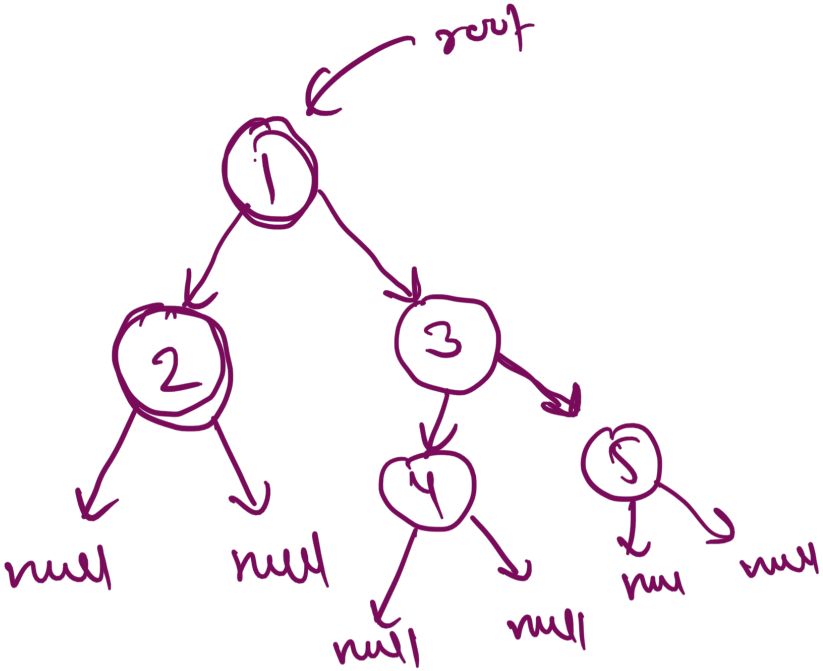
```
Node root = new Node(10);
n1 = 20
root.left = n1
n2 = 30
root.right = n2
n3 = 40
n2.left = n3
```

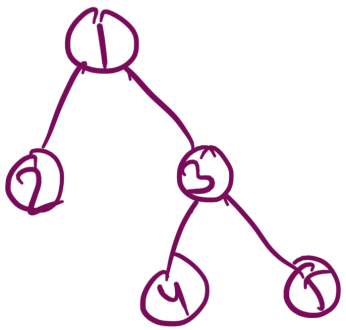




Convert a Binary Tree into a DLL. / flatten a binary tree.


node 2
20

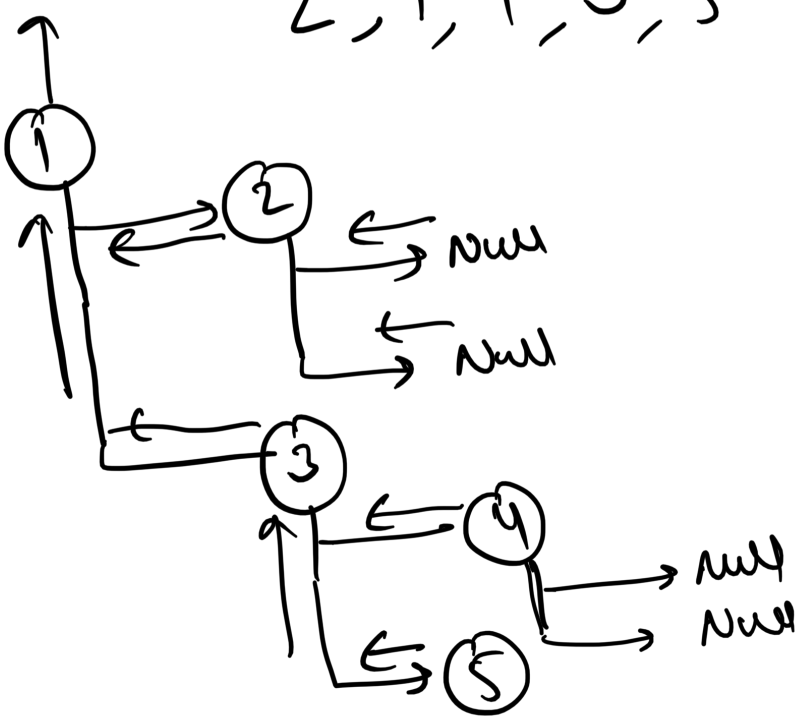




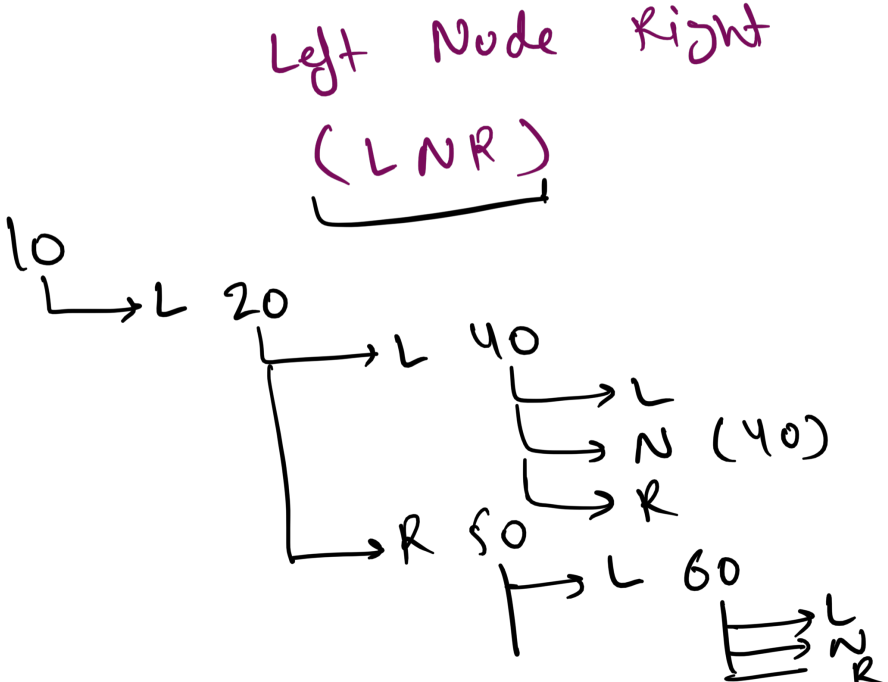
2, 1, 4, 3, 5

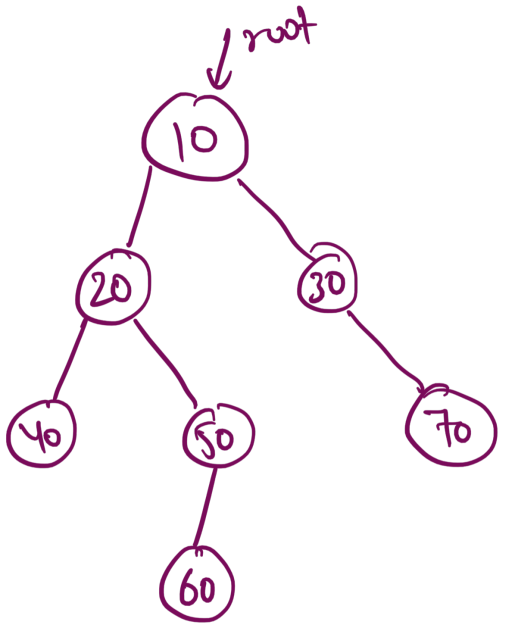
2, 1, 4, 3, 5

LNR

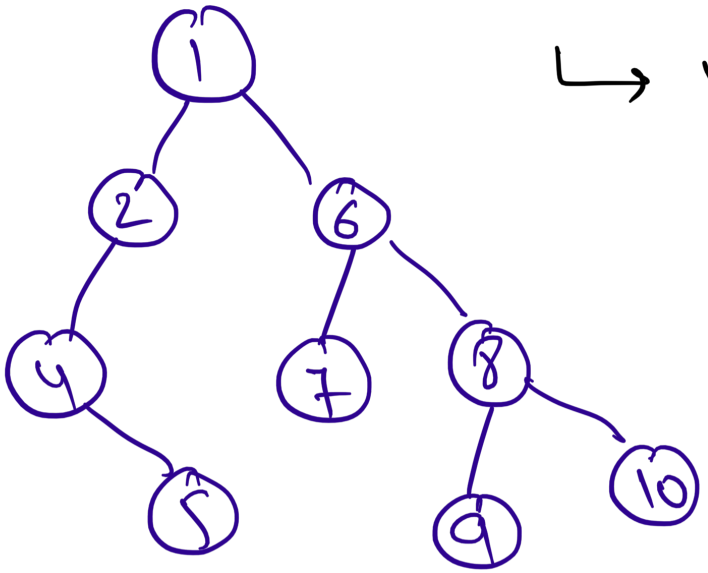


Inorder Traversal in a Binary Tree

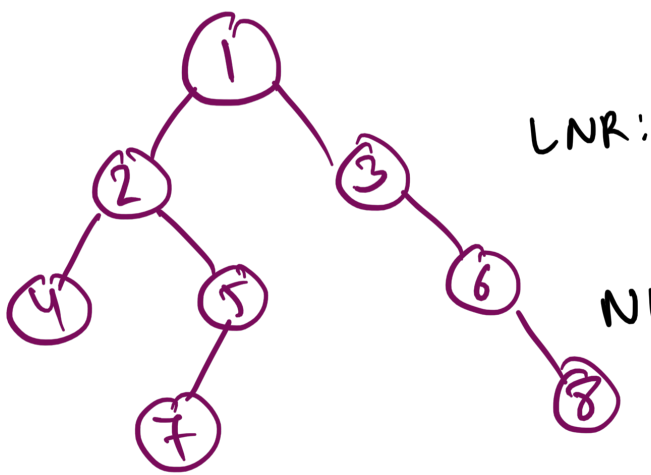




40, 20, 60, 50, 10, 30, 70



↳ 4, 5, 2, 1, 7, 6, 9, 8, 10

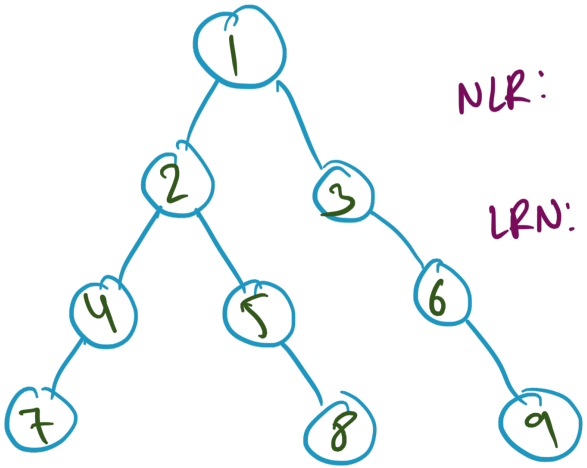


LNR: 4, 2, 7, 5, 1, 3, 6, 8

NLR: 1, 2, 4, 5, 7, 3, 6, 8

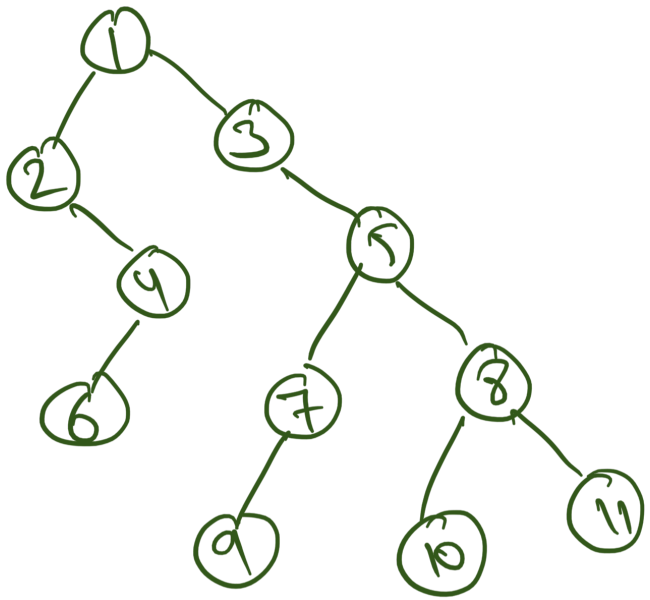
Preorder Traversal in a Binary Tree

Node Left Right



NLR: 1, 2, 4, 7, 5, 8, 3, 6, 9

LRN: 7, 4, 8, 5, 2, 9, 6, 3, 1



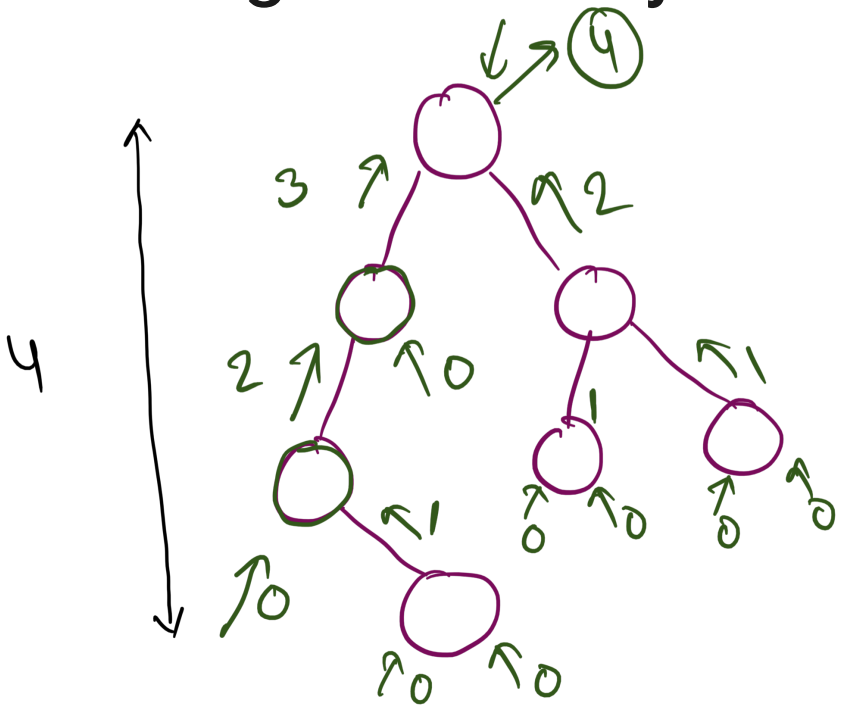
1, 2, 4, 6, 3, 5, 7, 9, 8, 10, 11

Node Left Right

Postorder Traversal in a Binary Tree

Left, Right, Node

Height of a Binary Tree

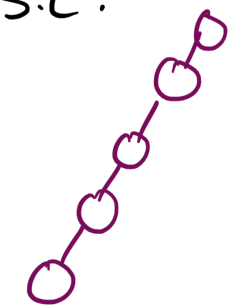



```
height (root) {
    lh = height (root.left);
    rh = height (root.right);
    return max (lh, rh) + 1;
}
```

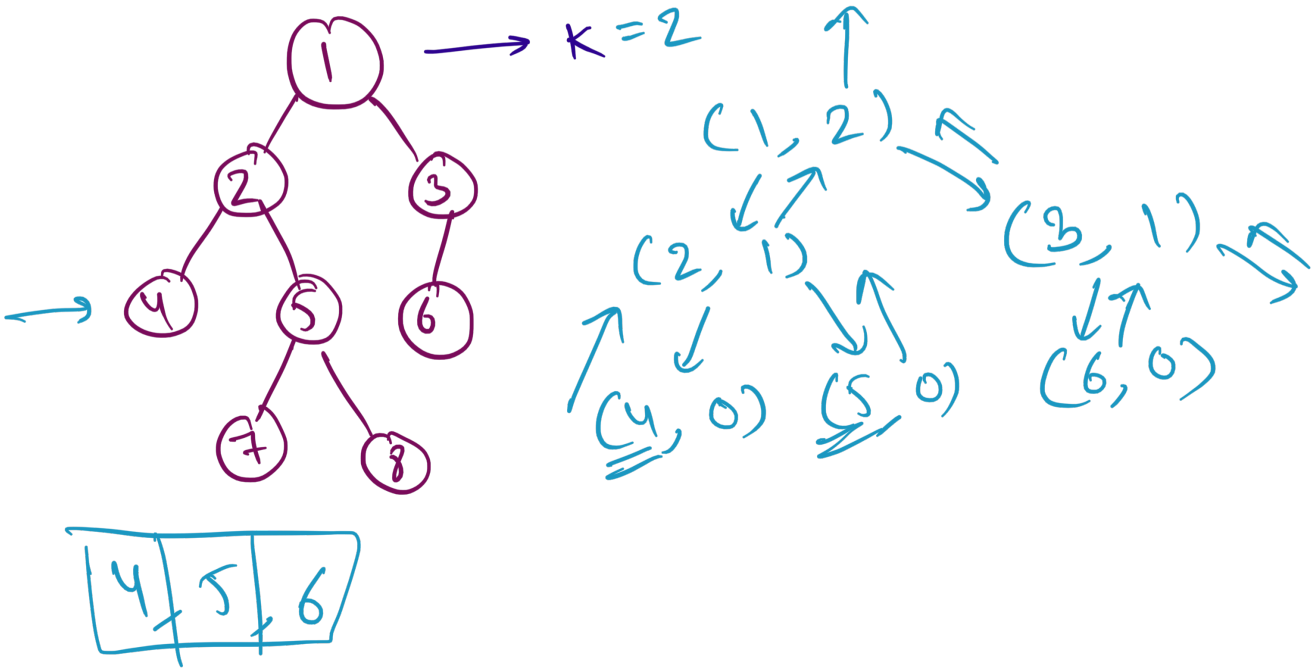
T.C. → O(n)

S.C. → O(height of Tree)

O(n)



Print all the nodes present at a distance k from Root



list.size()

Practice Problems

1. Postorder Binary Tree Traversal
2. Size of a Binary Tree → Number of nodes
3. Maximum / Minimum in a Binary Tree
4. Inorder/Preorder/Postorder without using Recursion