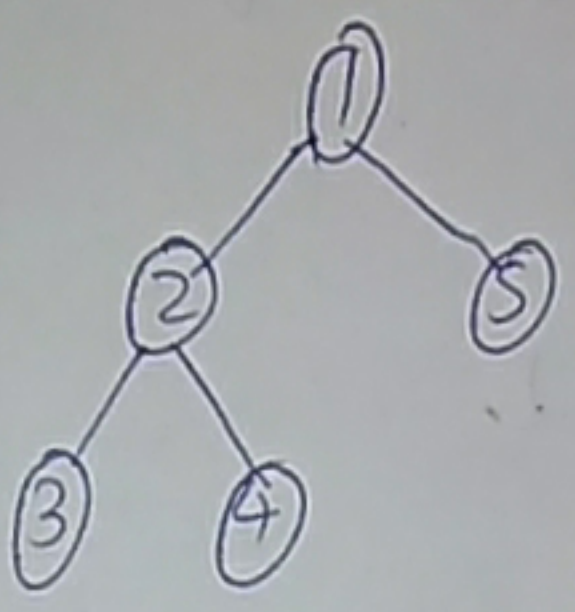
day 19

Maximum depth or hight of a tree

* Hight of empty tree is zero



⇒ hwight of this tree = 3

Algorithm :

* recursively call to get maxdepth of both left and right subtree.

Implemenation :

```
Class Node :
    def __init__(self, data):
        self.val = data
        self.left = None
        self.right = None
```

```
def maxDepth(node):
    if node is None:
        return 0
    else:
        ldepth = maxdepth(node.left)
        Rdepth = maxdepth(node.right)

        if(ldepth > rdepth):
            return ldepth + 1.
        else:
            return rdepth + 1
```

Input:

```
root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)
print(maxdepth(root))
```

Output:

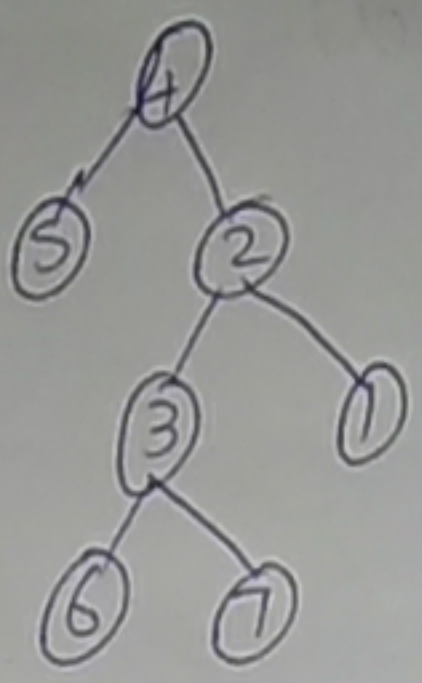
3

Time traversal:

O(n)

Print left view of binary tree

left view of binary tree is set of nodes visible when tree is visited from left side.



left view

⟹ 4 5 3 6

Implementation :

```
Class Node:

    def __init__(self,data):
        self.val = data
        self.right = None
        self.left = None

def leftviewutil(root, level, max_level):

    If root is None:
        return

    If (max_level[0] < level):
        print("%d\t" %(root.data)),
        max_level[0] = level
```

```
leftviewutill (root, level+1, max-level)
leftviewutill (root, level+1, max-level)

def leftview(root):
    max-level = [0]
    leftviewutill (root, 1, max-level)

    root = Node[1]
    root.left = Node(10)
    root.right = Node(20)
    root.right.left = Node(25)
    root.right.right = Node(40)
    leftview(root)
```

Output :

12 10 25

Time Complexity :

O(n)

Find minimum and maximum in binary tree

```
class Node:

    def __init__(self,data):
        self.data  = data
        self.right = None
        self.left  = None

def findmaxm(root):
    if(root == None):
        return float('-inf')
    res = root.data
    lres = findmax(root.left)
    Rres = findmax(root.right)
    if(lres > res):
        res = lres
    if(Rres > res):
        res = Rres
    return res
```

```
def findmin(root):
    if (root == None):
        return float('inf')
    res = root.data
    lres = findmin(root.left)
    Rres = findmin(root.right)
    if(lres < res):
        return res = lres
    it (lres < res):
        res = Rres
    return res
```

Input:

```
root = Node(4)
root.left = Node(7)
root.right = Node(5)
root.left.right = Node(6)
root.left.right.left = Node(1)
root.left.right.right = Node(11)
print("maximum is:", findmax(root))
print("minimum is", findmin(root))
```

Output:

```
maximum is 11
minimum is 1
```