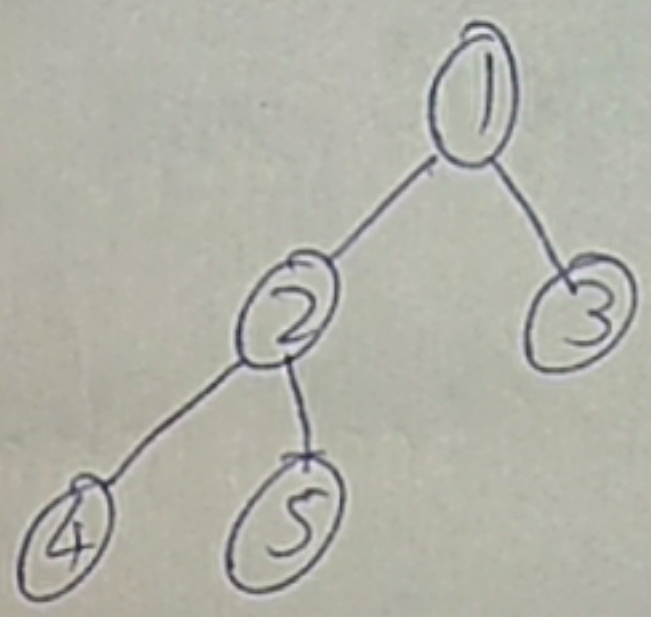
Day 18

Tree traversal

Depth first traversals:

* Inorder
* Preorder
* Postorder

Example:



Inorder : 4 2 5 1 3

preorder : 1 2 4 5 3

postorder : 4 5 2 3 1

Inorder traversal:

* Traverse the left tree
* visit root
* Traverse the right tree

preorder traversal:

* visit root
* traverse the left tree
* traverse the right tree

postorder traversal:

* traverse the left subtree
* traverse the right subtree
* Visit the node root

Program for tree traversal;

```
Class Node:

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

```
def Inorder (root):
    if root:
        Inorder (root.left)
        print(root.val)
        Inorder (root.right)

def postorder (root):
    if root:
        ~~Inorder(root)~~.
        postorder (root.left)
        postorder (root.right)
        print(root.val)

def preorder (root):
    if root:
        print(root.val)
        preorder (root.left)
        preorder(root.right)
```

Input:

```
root    = Node(1)
root.left  = Node(2)
root.right  = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)
print("preorder traversal:")
preorder(root)
print("postorder traversal:")
postorder(root)
print("Inorder traversal:")
Inorder(root)
```

Output:

preorder traversal:

1 2 4 5 3

postorder traversal:

4 5 2 3 1

Inorder traversal:

4 2 5 1 3

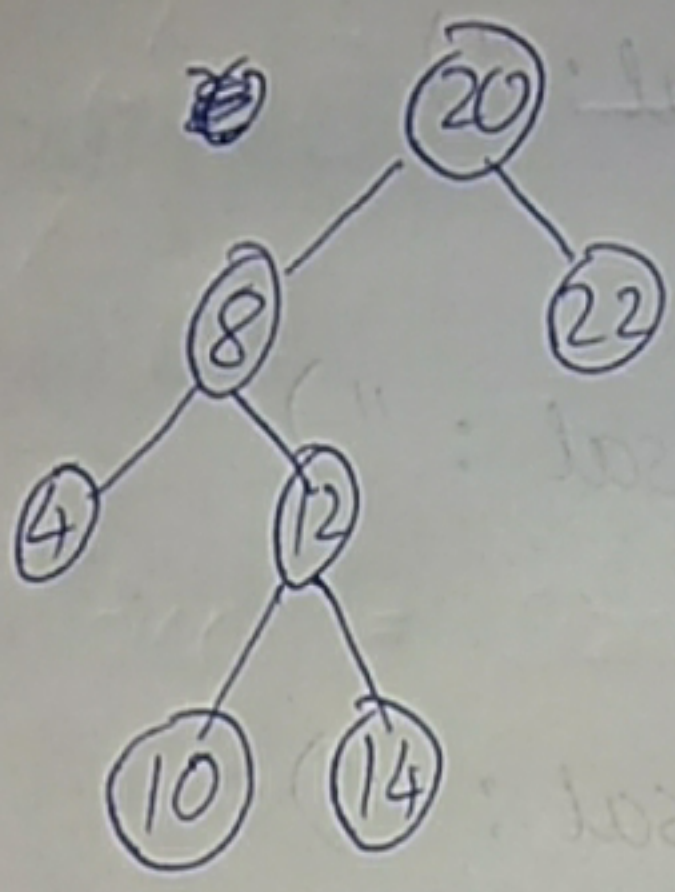
Time Compleity:

O(n)

Print level order traversal (line by line)

All levels are printed in seprate lines.

Example:



→ output

20

8 22

4 12

10 14

Programme:

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

```
def printlevelorder(root):
    if root is None:
        return
    q = []
    q.append(root)
    while q:
        count = len(q)
        while count > 0:
            temp = q.pop(0)
            print(temp.val, end = " ")
            if (temp.left):
                q.append(temp.left)
            if (temp.right):
                q.append(temp.right)
            count -= 1
        print('')
```

Input:

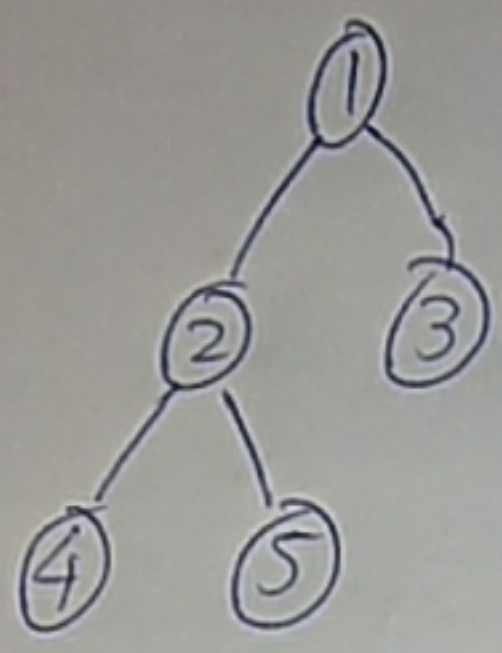
```
root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)
printlevelorder(root)
```

output

```
1
2 3
4 5 6
```

Size of tree

Size of tree is the number of elements present in the tree.



Size of tree → 5

Size_of tree = Sizeof leftree + 1
+ size of rightt

Programme;

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

```
def size(node):
    if node is None:
        return 0
    else:
        return (size(node.left) + 1 + size(node.right))
```

Input :-

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

output:

5