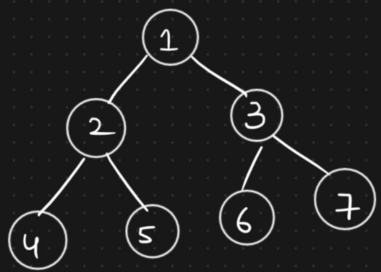
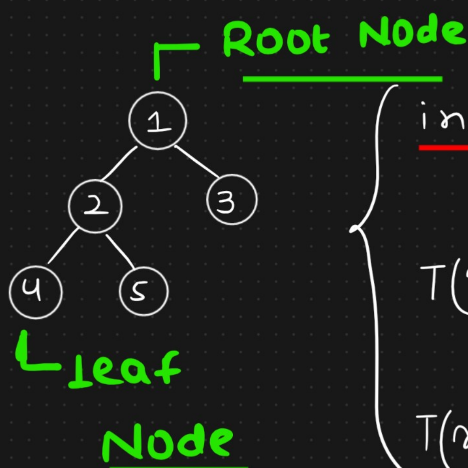
Tree Traversal Algorithms



1) Inorder
2) Preorder
3) Postorder

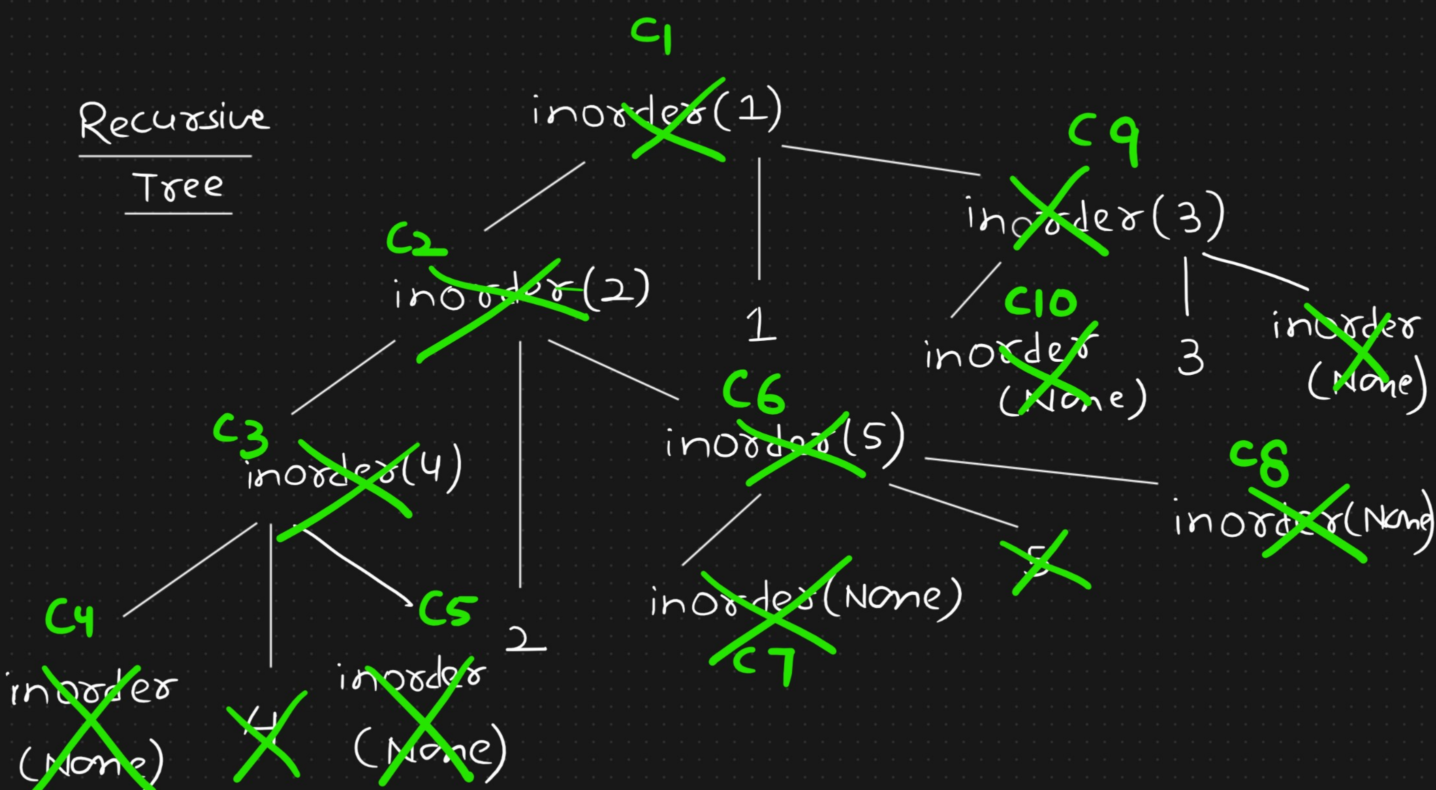
Recursion concept

Inorder
- Left side of the tree
- print root node
- Right side of the tree

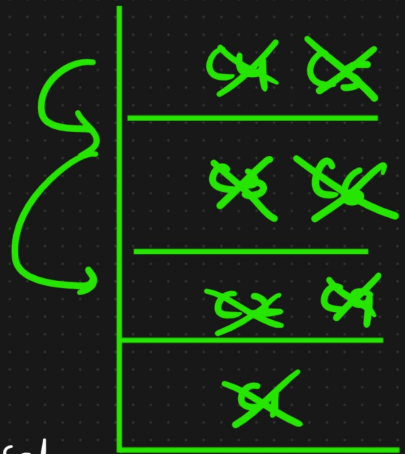



```
inorder(root):            T(n)
    if root:              Recursion
        inorder(root.left)     T(n/2)
        print(root.data)       c
        inorder(root.right)    T(n/2)
```

Recursive Tree



4 2 5 1 3



Stack

↳ Balanced Tree

for every traversal algorithm $\boxed{\text{Recurrence Relation}}$ (Best/average case scenario)

$$\boxed{T(n) = 2T(n/2) + c}$$

By master's theorem

$a = 2 \qquad k = 0$

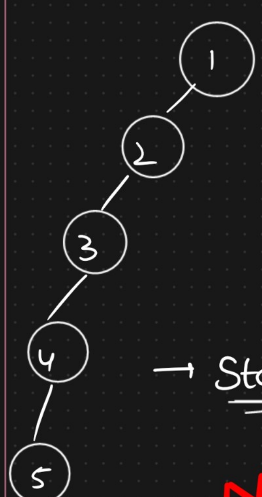
$b = 2 \qquad p = 0$

$\log_b a = 1 > k$

↳ $O(n)$

Stack space

↳ $\log_2 n$



worst case scenario → unbalanced tree or skewed tree

$$\boxed{T(n) = T(n-1) + c}$$

$\Rightarrow O(n)$

→ Stack Space

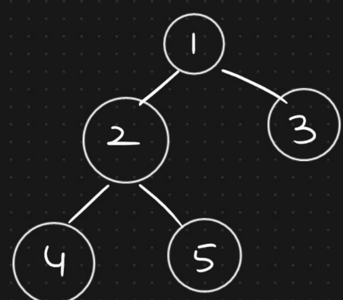
↳ $O(n)$

Note: for every case, Overall time complexity $= O(n)$

Preorder Traversal

```
preorder(root):
    if root:
        print(root.data)      — ①
        preorder(root.left)   — ②
        preorder(root.right)  — ③
```

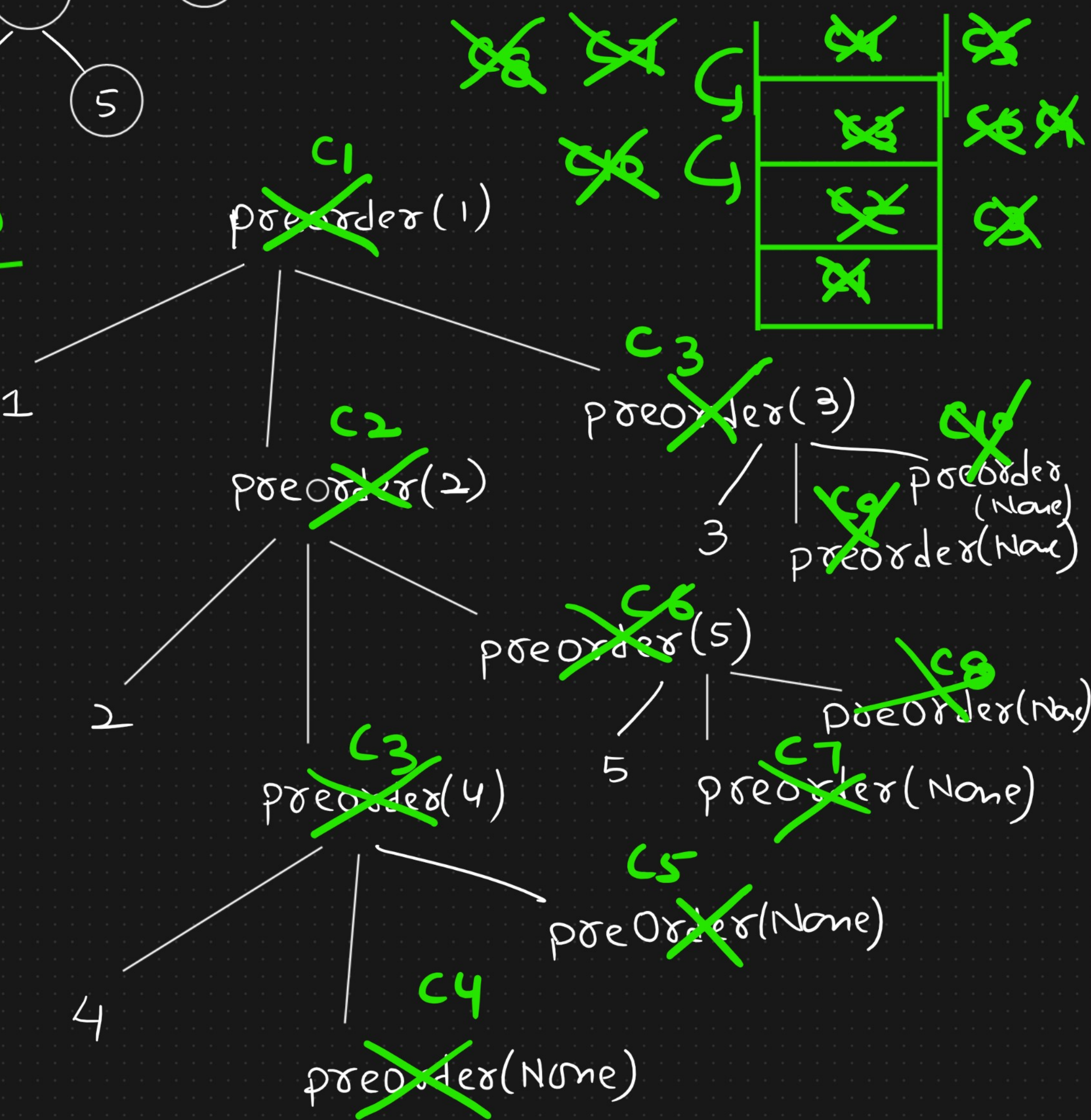
↳ **Recursion**



→ 1 2 4 5 3

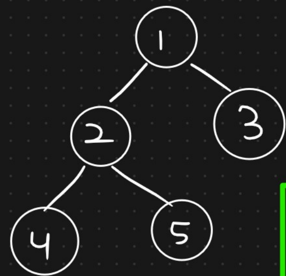
1 2 4 5 3

Recursive Tree



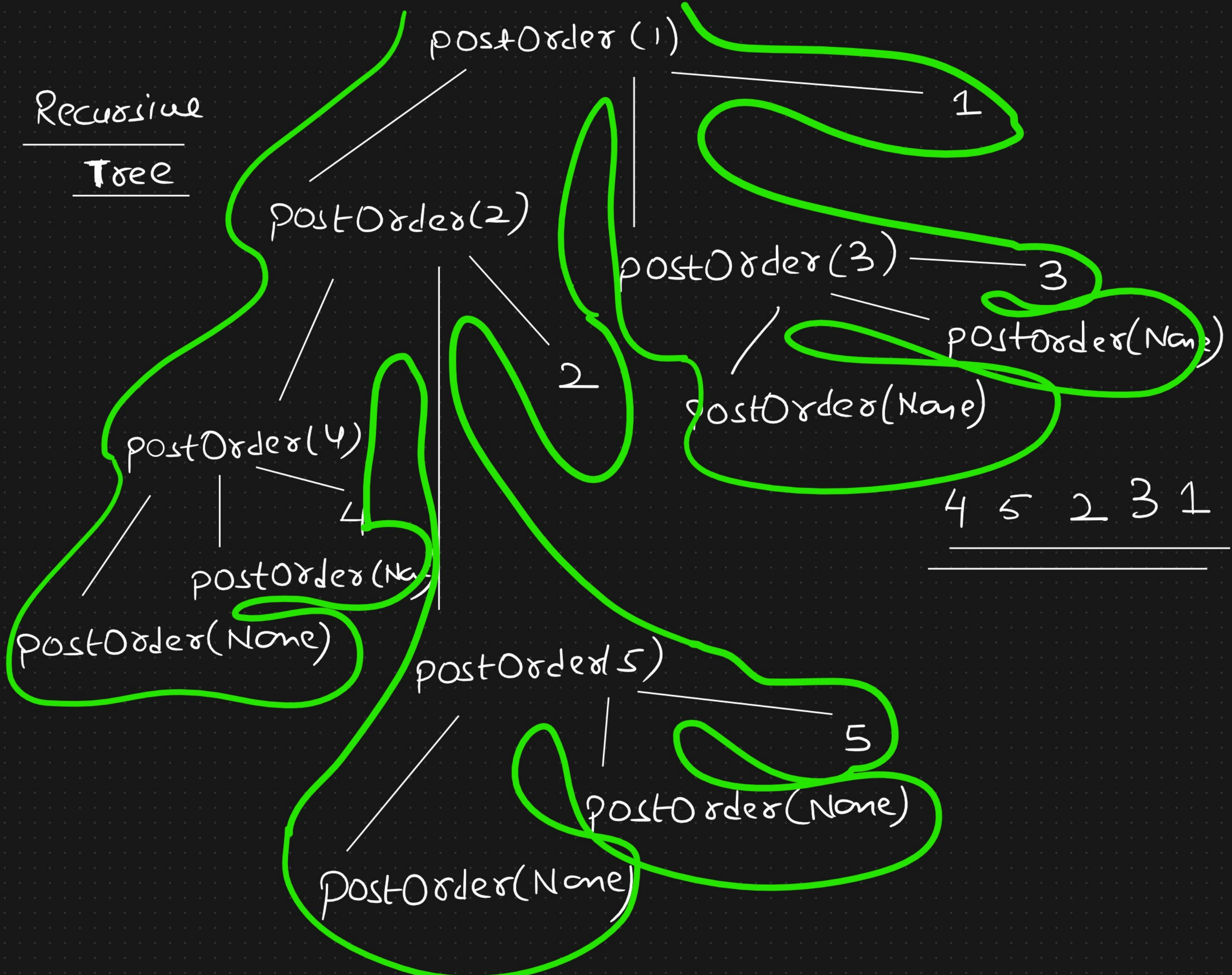
Postorder

```
postOrder(root):
    if root:
        postOrder(root.left)
        postOrder(root.right)
        print(root.data)
```



$\boxed{4\ 5\ 2\ 3\ 1}$

Recursive Tree



Note: Root Node of a Tree

- Inorder (Middle)
- Preorder (first)
- PostOrder (Last)