Tree Data Structure

Tree → 1) Non-linear Data Structure, Hierarchy based

2) Application - Database (Indexing, file system)

↓

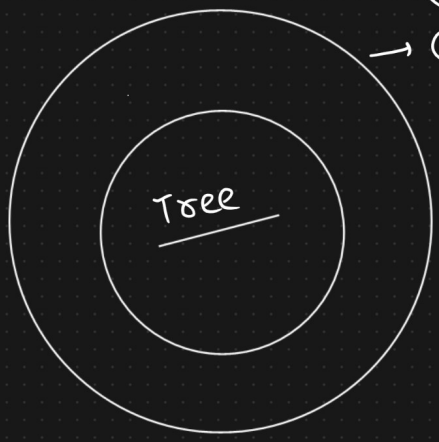
acyclic (No cycle)

- Networking
- Google Maps

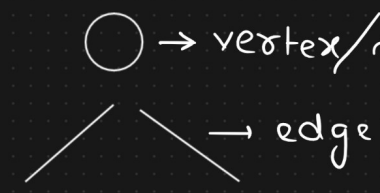
Graph → cycle as well

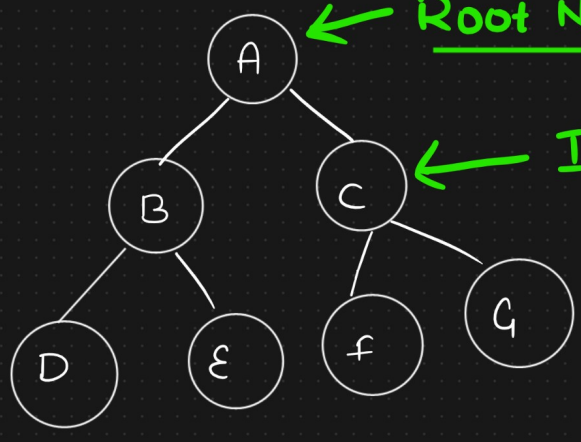
cycle (acyclic)

→ Graph



↳ Binary Tree → atmost 2 child nodes (0, 1, 2)





← Root Node

← Internal Nodes

← Leaf Node (0 child)

Traversal → Traverse every node inside the tree

1) Preorder Traversal → 1) Root Node
2) Inorder Traversal → 2) Left subtree (Recursion)
3) Postorder Traversal → 3) Right subtree (Recursion)

Left Sub
Root Node
Right Sub

Left, Right, Root

Preorder Traversal

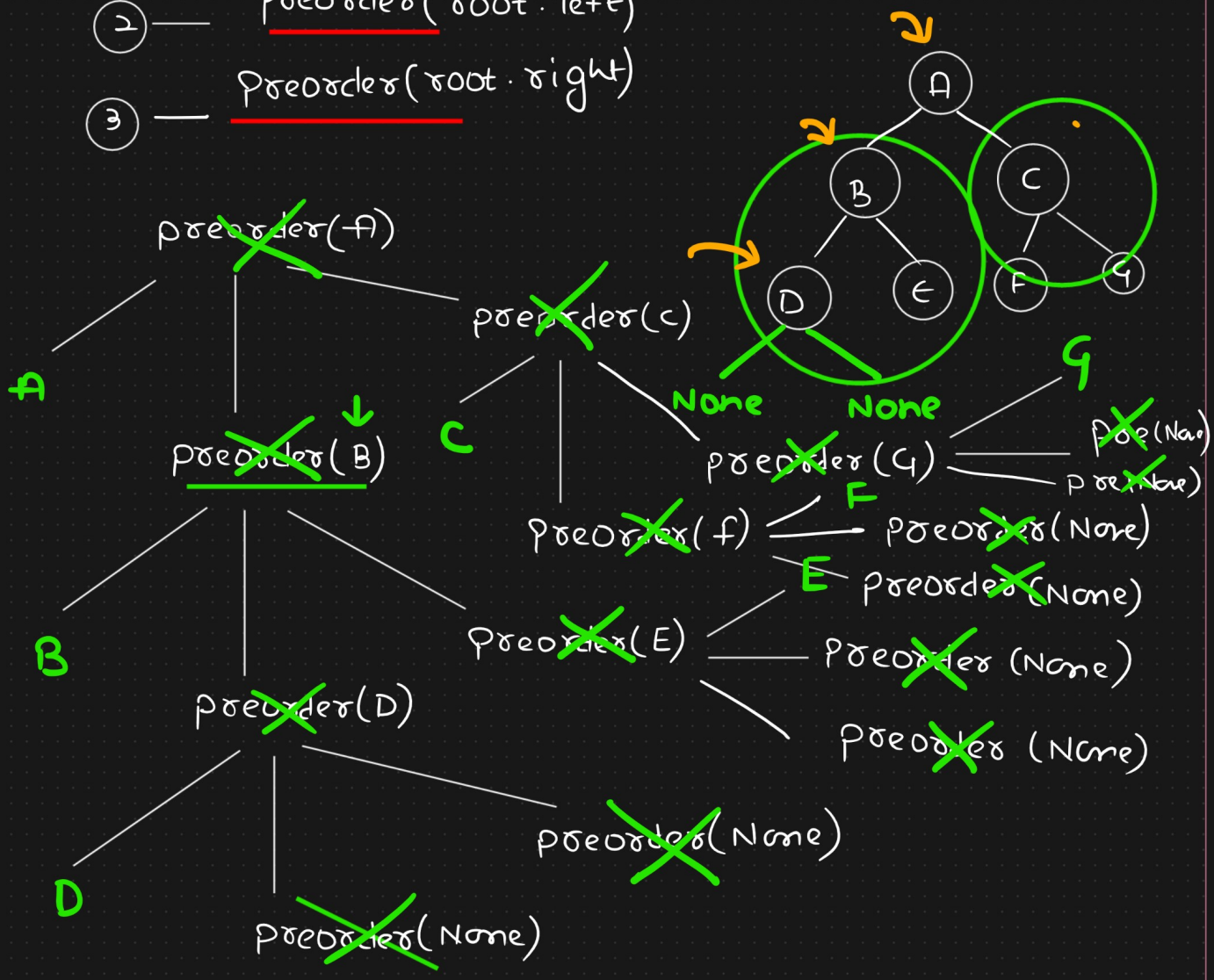
A B D E C F G

root is not None:

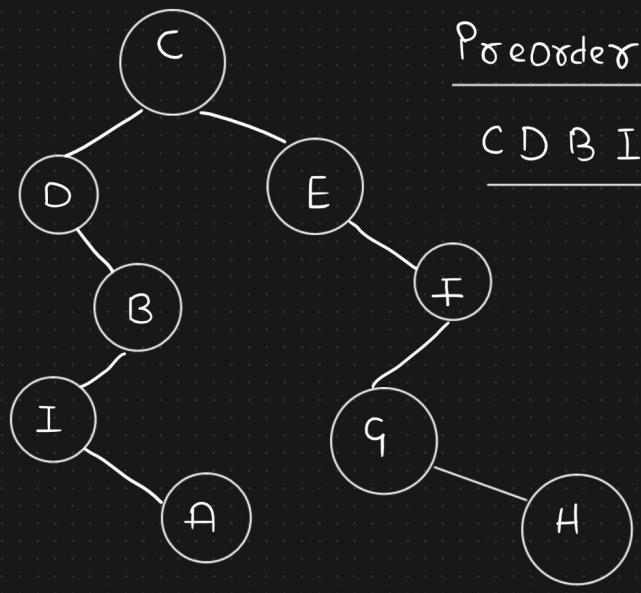
Pseudocode

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

Recursion



Practice Problem



Preorder

C D B I A E F G H

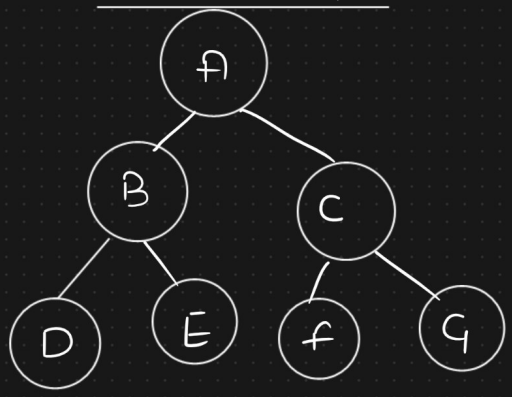
Inorder

D I A B C E G H F

Inorder Traversal

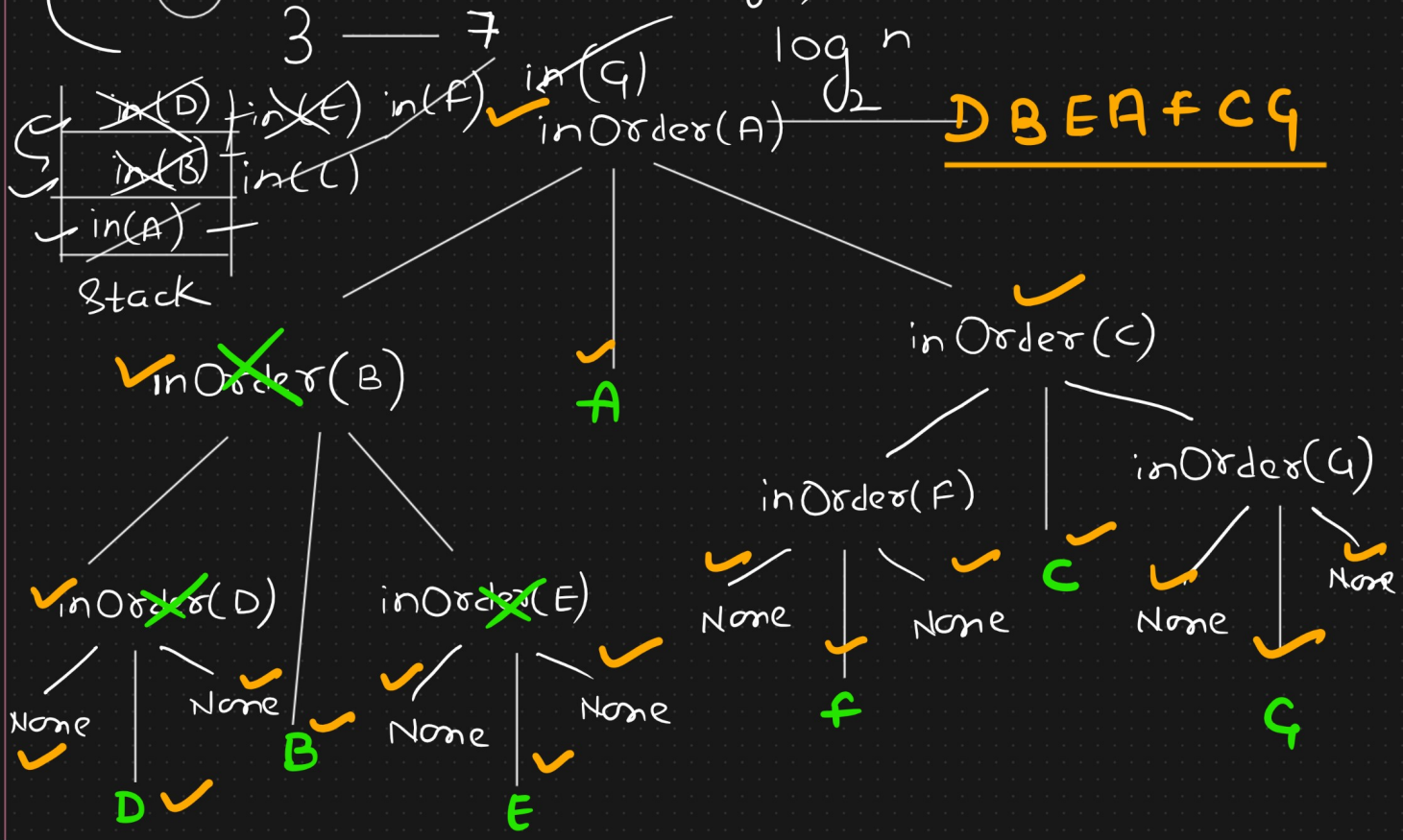
```
inOrder(root):
    if root :
        1 ——— inOrder(root.left)
        2 ——— print(root.data)
        3 ——— inOrder(root.right)
```

3 —— 7



DBEAfCG

$\log_2 n$



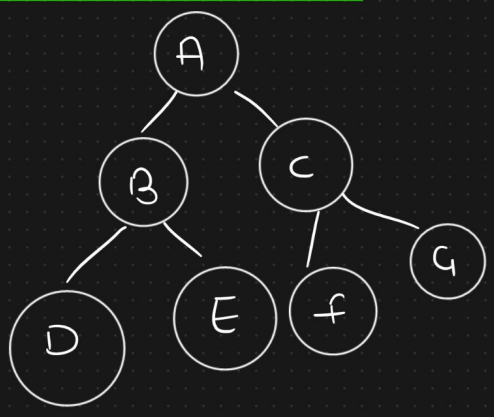
Recursive Tree

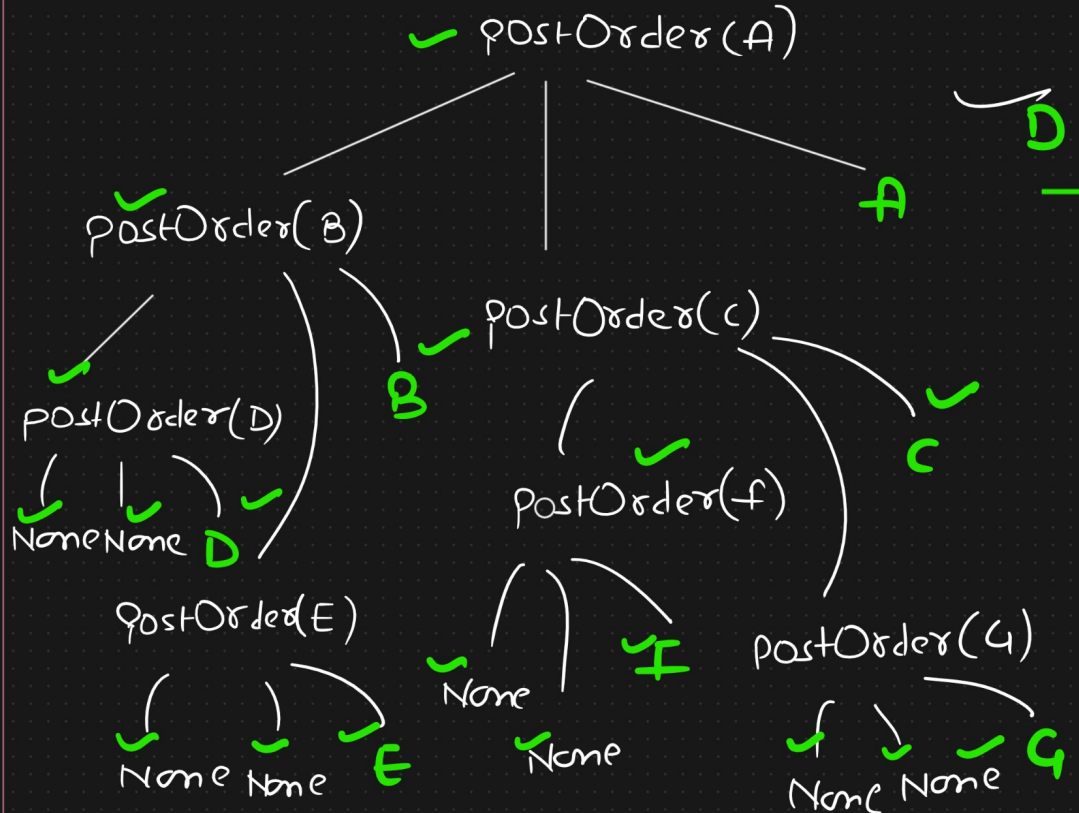
↳ Preorder fashion

Postorder

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

DEBFGCA



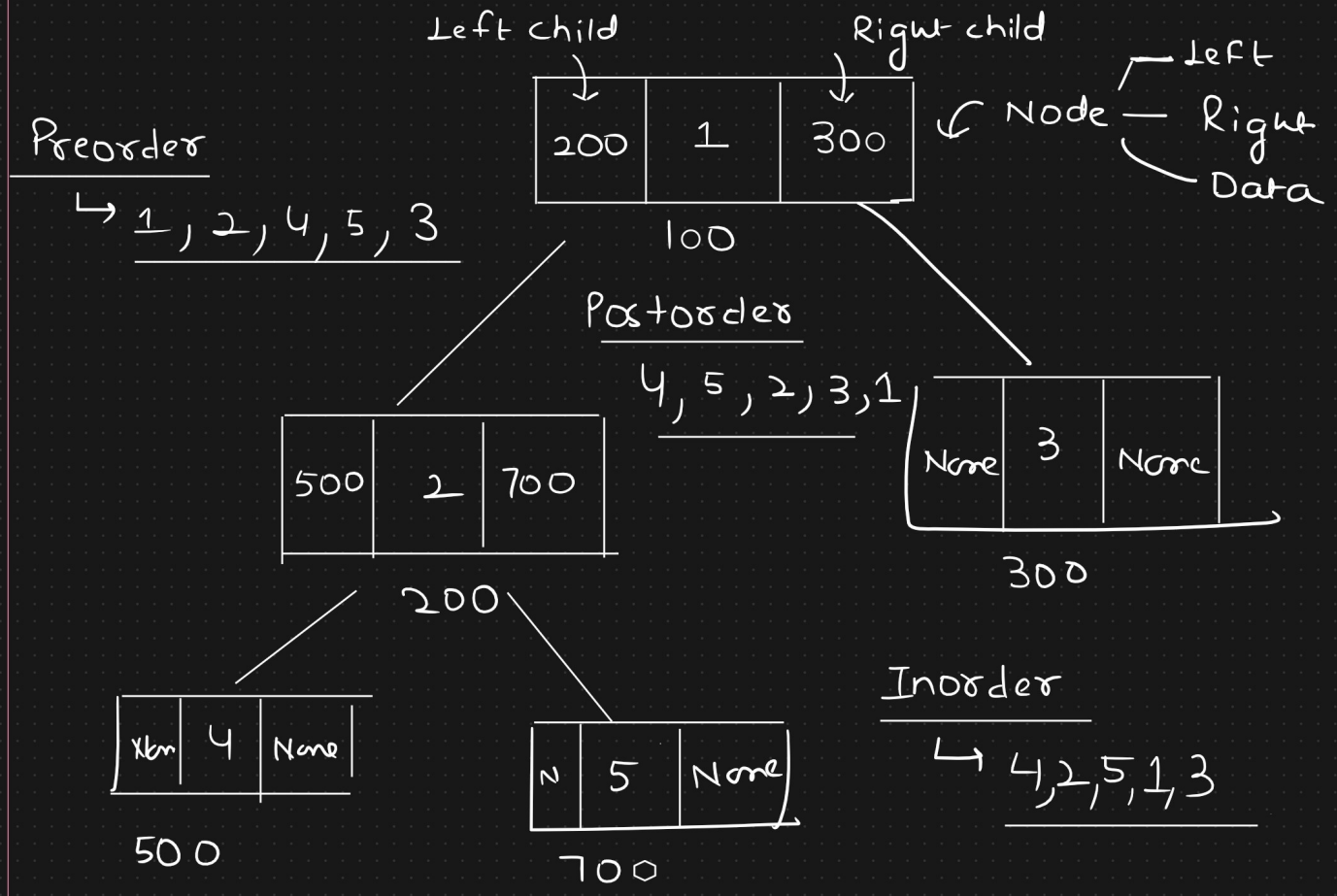


DEBFGCA

PostOrder → Root Node (End)

PreOrder → Root Node (Start)

InOrder → Root Node (In between)



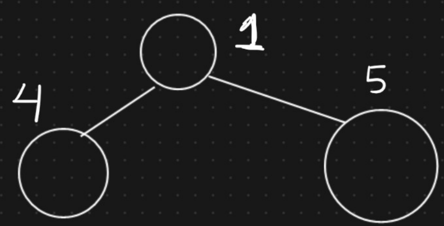
Time complexity

Preorder — if root :
  Print(root.data) — ① (c)
  Preorder(root.left) — ② $T(n/2)$
  Preorder(root.right) — ③ $T(n/2)$

Preorder → $T(n)$

Balanced Tree → Best case scenario

Balanced Tree



Equal distribution of elements on Left & Right subtree

Recurrence Relation

$$T(n) = 2T(n/2) + c$$

$a = 2$

$b = 2$

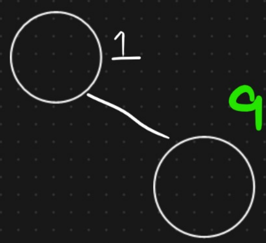
$\log_b a = \log_2 2 = 1$

$$T(n) = \Theta(n)$$

Skewed tree

→ unbalanced tree

Worst case scenario



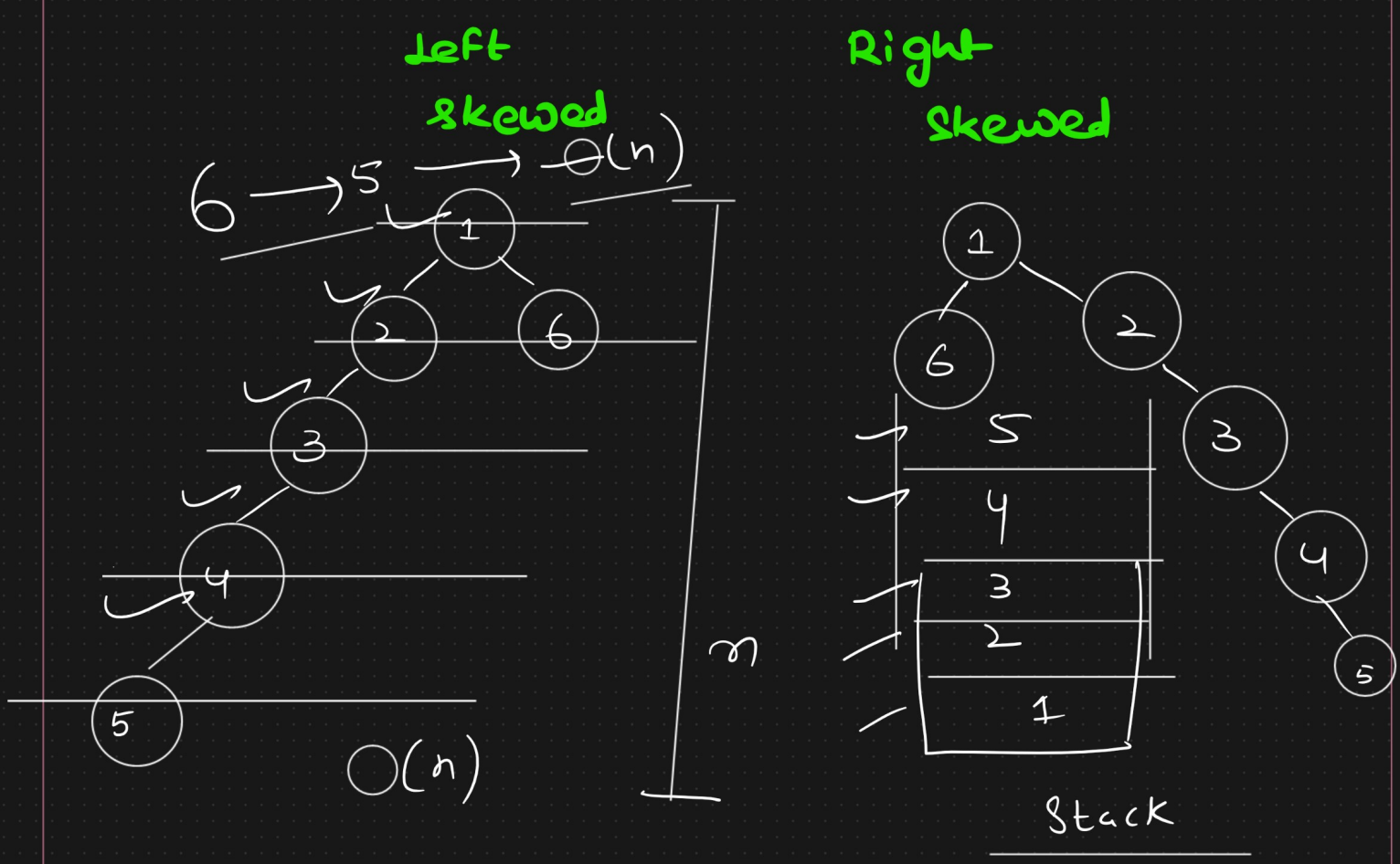
$$T(n) = T(n-1) + c$$

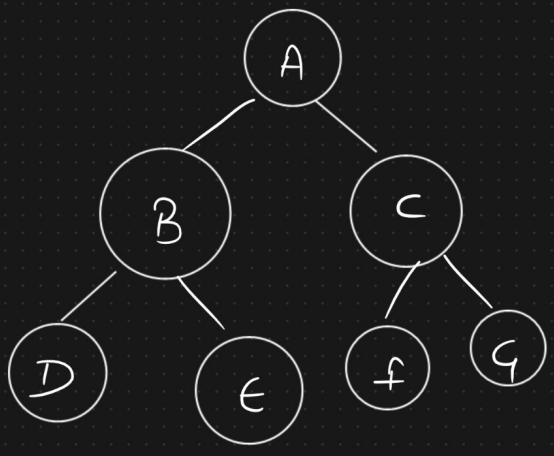
$$T(n) = \Theta(n)$$

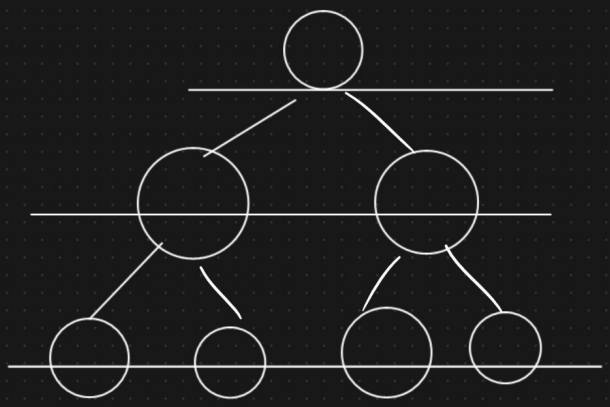
Space complexity → $\Theta(\log n)$ → Best case Scenario

Space complexity → $\Theta(n)$ → Worst case Scenario

Skewed Tree


Left Skewed
Right Skewed
6 → 5 → O(n)
1
2
6
3
4
5
O(n)
n
1
6
2
5
3
4
3
2
1
4
5
Stack


A
B
C
D
E
f
G



$\log_2 n$