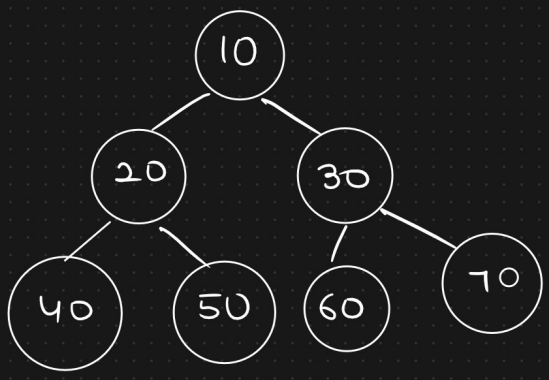
TREE Data Structure

1) Preorder
2) Postorder
3) Inorder

→ Implementation using Recursion

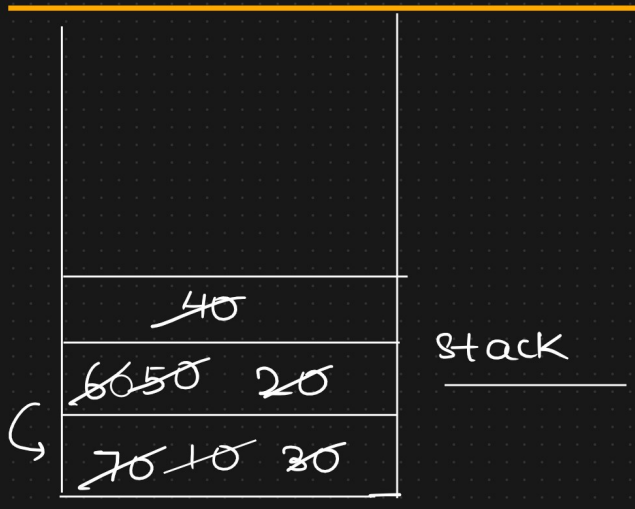
↳ Without using Recursion

Preorder Traversal



Expected O/P: 10, 20, 40, 50, 30, 60, 70

root = 10



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

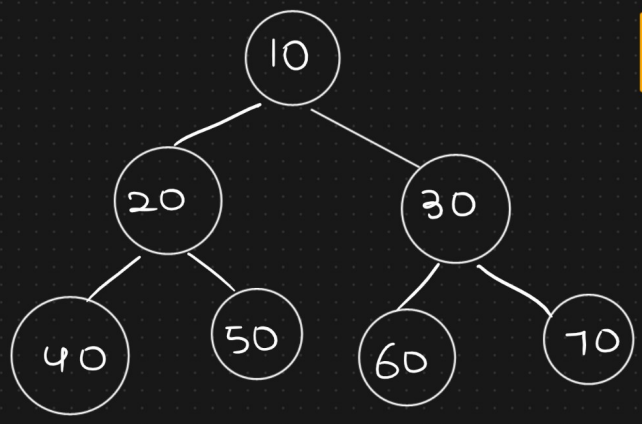
↳ Print → 10, 20, 40, 50, 30, 60, 70

Stack → LIFO

```
stack.append(root)                    ①

while len(stack)>0:                   ②
    popped = stack.pop()
    print(popped.data)
    if popped.right:
        stack.append(popped.right)
    if popped.left:
        stack.append(popped.left)
```

Stack → LIFO



Postorder

→ 40, 50, 20, 60, 70, 30, 10

current = ~~10~~ ~~30~~ ~~70~~ ~~60~~ ~~20~~ ~~50~~ 40

Stack 1

~~70~~

~~50~~ ~~30~~ ~~60~~

40 ~~10~~ ~~20~~

Stack 2

~~40~~

~~50~~

~~20~~

~~60~~

~~70~~

~~30~~

~~10~~

current = root

While stack1:

→ current = stack1.pop()

→ stack2.append(current.data)

→ if current.left :

stack1.append(current.left)

if current.right:

stack1.append(current.right)

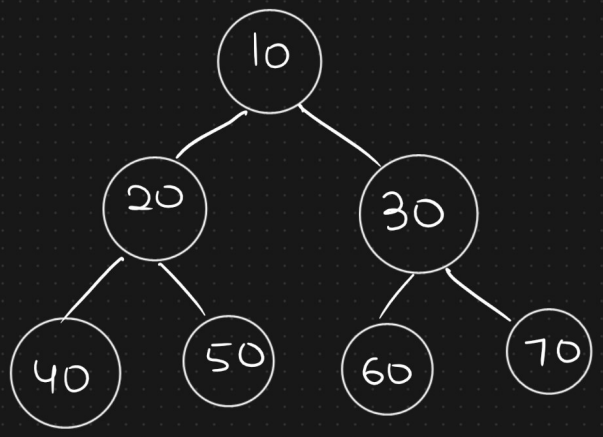
While stack2:

data = stack2.pop()

print(data)

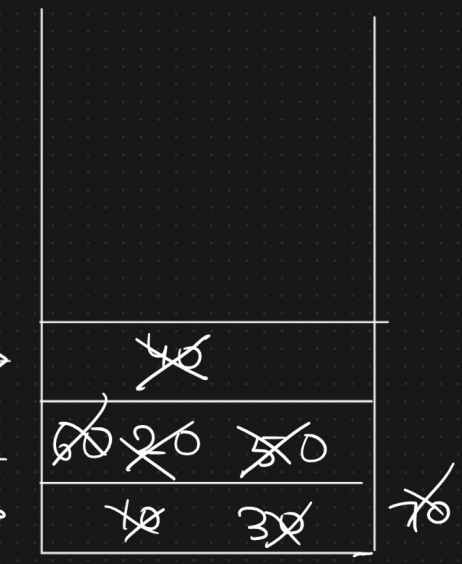
Inorder Traversal

Left → Root → Right



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

current = ~~10~~ ~~20~~ ~~40~~ 20

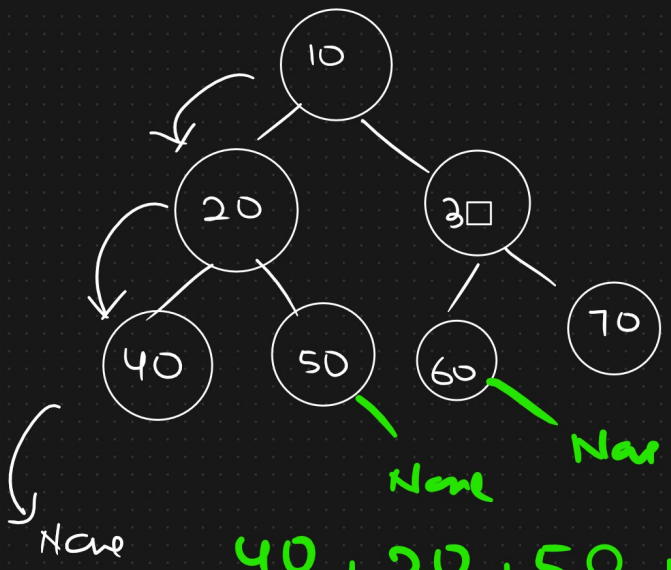


While True:

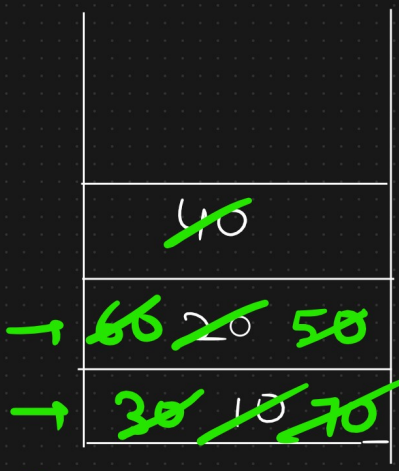
```
if current:
    stack.append(current.data)
    curr = curr.left

elif len(stack)>0:
    curr = stack.pop()
    print(curr.data)
    curr = curr.right
```

40, 20



current = 10 20 40 None



40 None
20 50
None 30
None 10
30 60
60 None
None 70
70 None

40, 20, 50, 10, 60, 30, 70 →

Unique Binary Tree { Inorder + Preorder ; Inorder + Postorder }

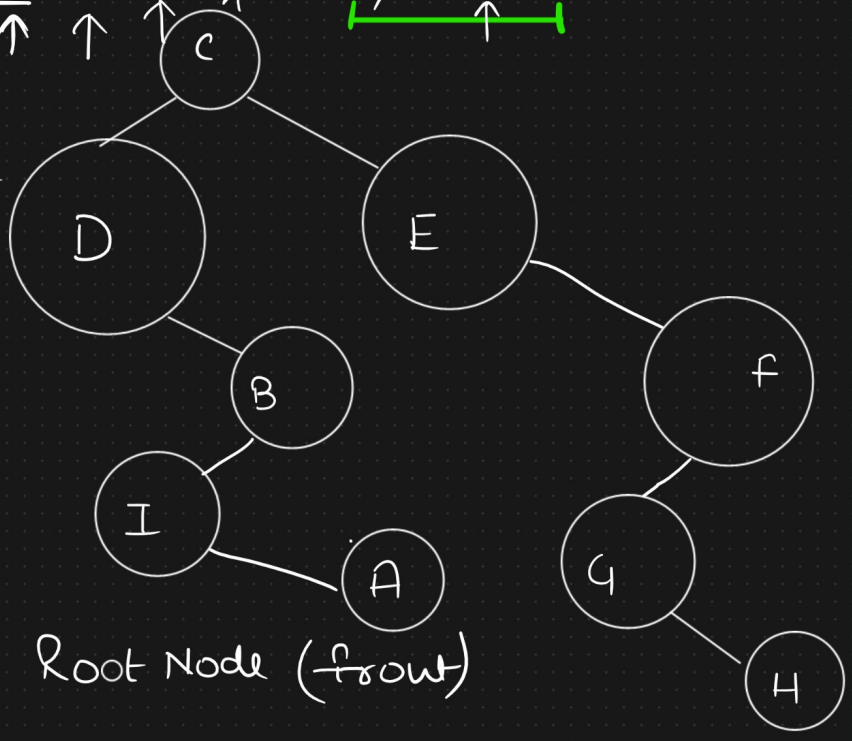
Left subtree — Right subtree

Inorder → D, I, A, B, C, E, G, H, f

Preorder → C, D, B, I, A, E, f, G, H

Postorder → Left subtree, Right subtree, Root

Left Subtree



Preorder → Root Node (front)

Postorder → Root Node (last Node)

Postorder → A, I, B, D, H, G, f, E, C

9, 3, 15, 20, 7 → Inorder

9, 15, 7, 20, 3 → Postorder

Preorder → 3, 9, 20, 15, 7

Unique Binary Tree