24th August 2021

1. Iterator in Generic Tree
2. Iterator in Binary Search Tree
3. Root to All leaf path in Binary Tree
4. All Single child in Binary Tree
5. Count Single Child in Binary Tree

26th August 2021

O(1) space Traversal in Binary Tree.

1. Inorder Morris Traversal
2. Pre Order Morris Traversal
3. Post order Morris Traversal
4. Iterator of Binary tree using Morris traversal

Application

28th August 2021 (Morning)

1. Path Sum in Binary Tree 2
2. Diameter of Binary Tree (All Methods)
3. Maximum path sum in B/w two leaf
4. All nodes distance K in B.T.
5. Burning Tree

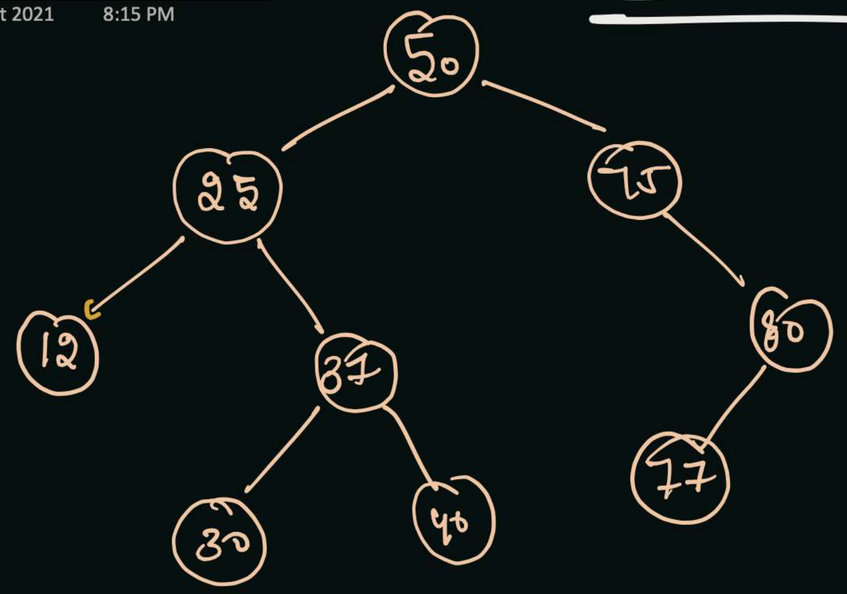
28th August 2021 (Evening)

1. Burning Tree 2
2. max-width of Binary Tree
3. Convert BST to Doubly LL
4. Convert Sorted DLL to BST
5. Path sum in Binary Tree

Morris Traversal

Thursday, 26 August 2021 8:15 PM

In Order.



In binary Tree we can move from parent to child

→ left pointer + right pointer

But using Recursion we can achieve traversal from child to parent.

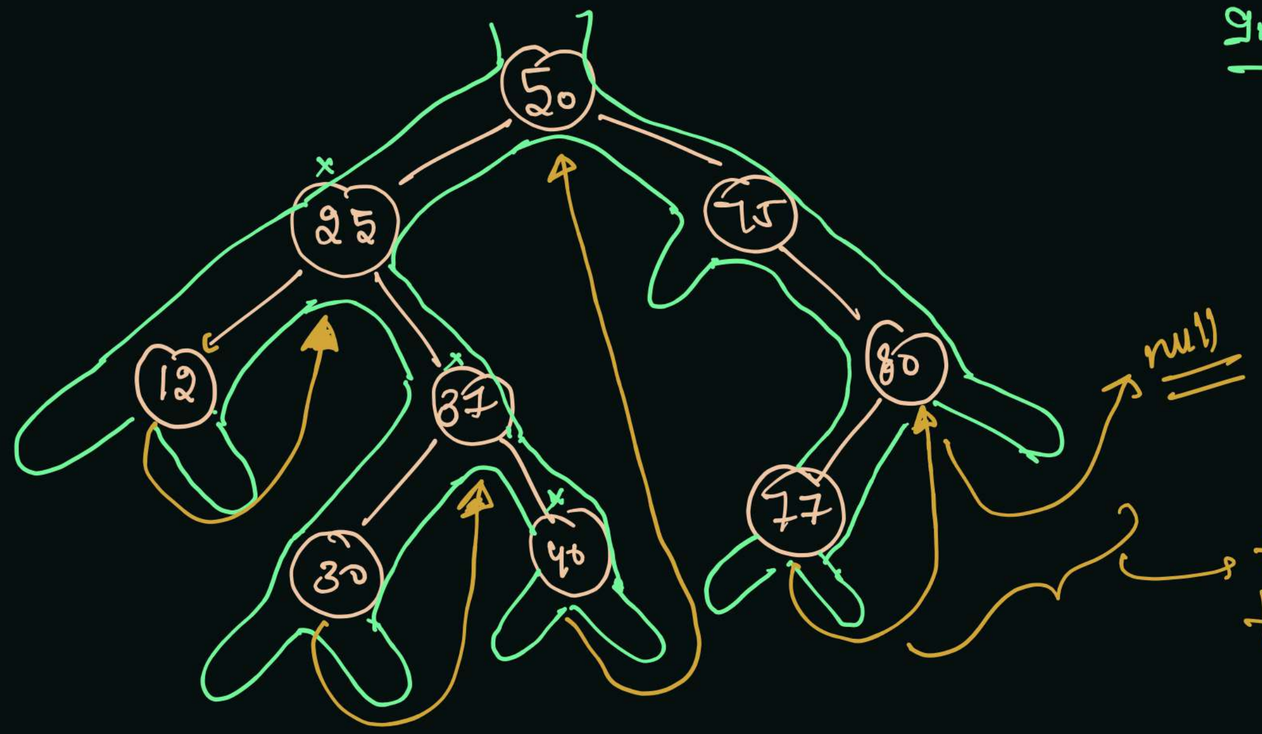
Ex → Top to Bottom behavior of recursion.

How is it possible to travel from child to parent ?? → a). Extra space (Recursive space) → Backtrack pointer

b). thread pointer ??

when we travel in Stack → In stack we store our parent as temporary variable. → Bottom-up

→ can we achieve backpath without it



In order

Base case.

call(left)

print

call (Right)

these are thread connection.

path is completely travel or not

→ If left is null path is complete

80, ~~lr~~

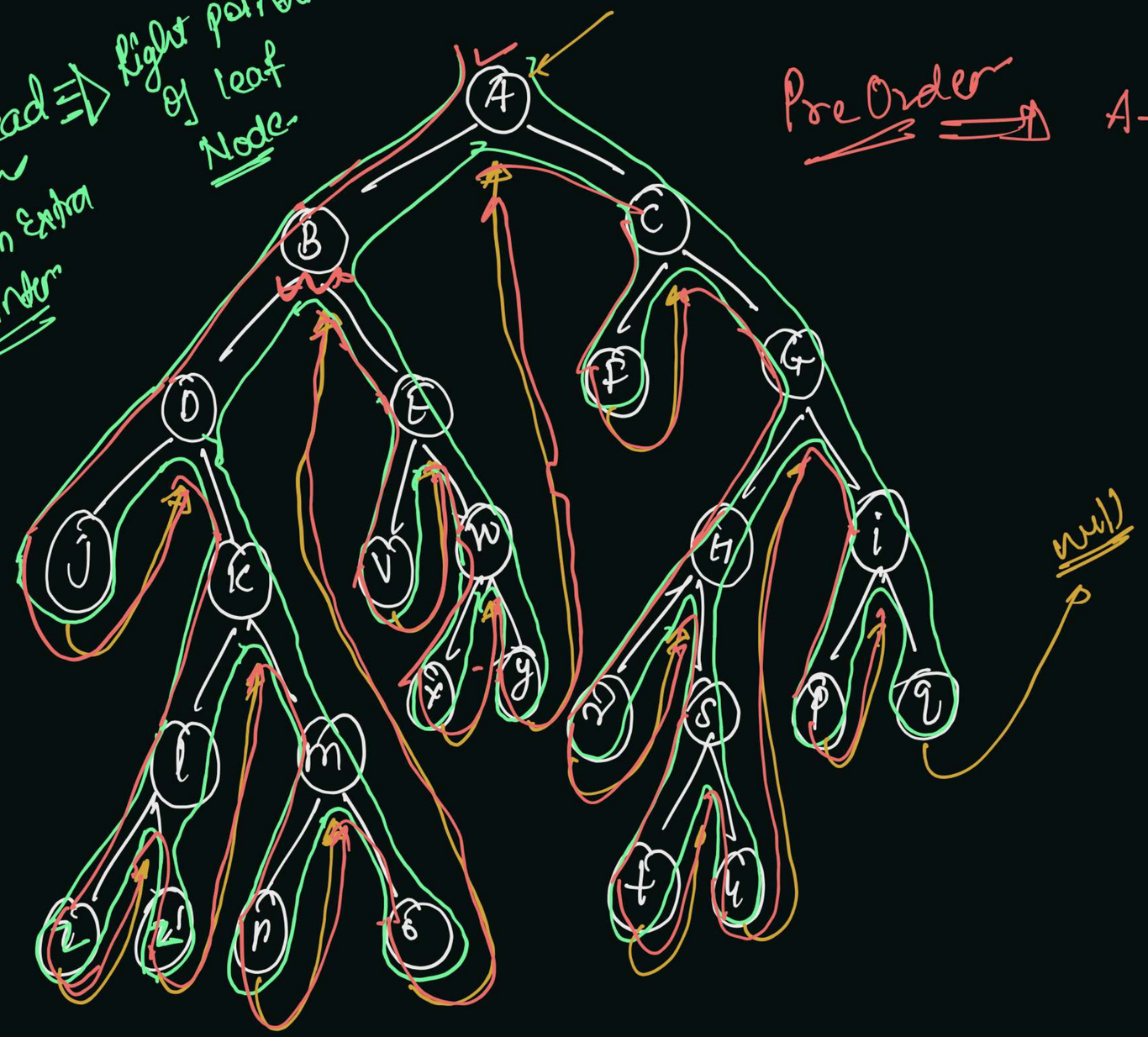
75, ~~lr~~

50, ~~lr~~

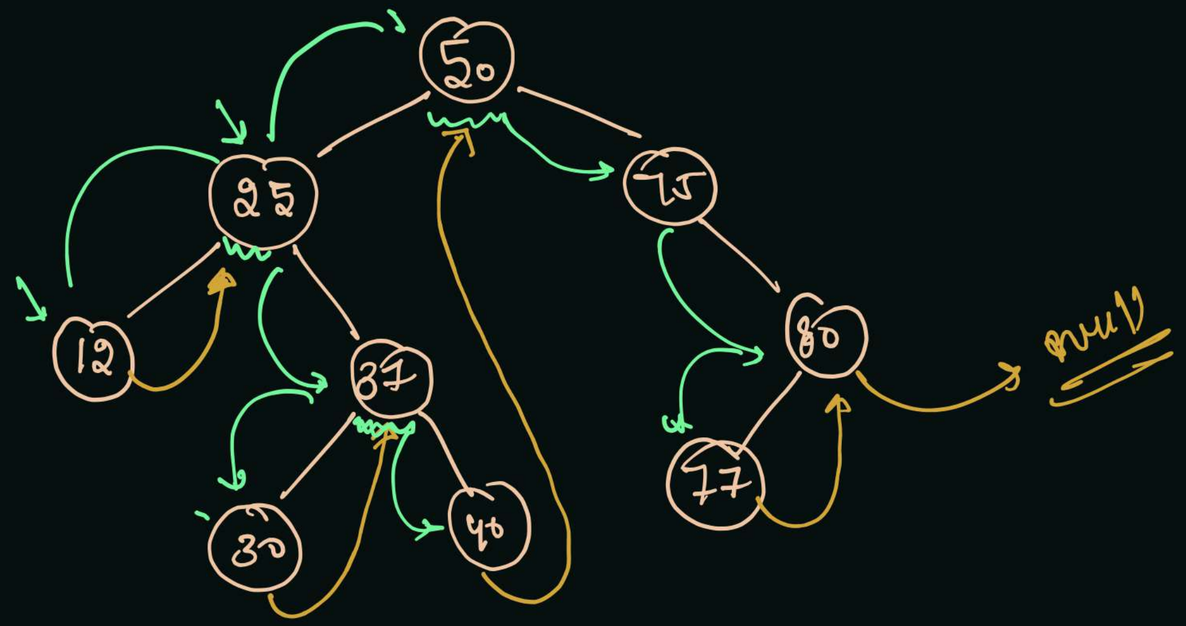
12, 25, 30, 37, 40, 50, 75, 77

Thread ⇒ Right pointer of leaf Node.

Not an Extra pointer



Pre Order ⇒ A → B → D → J → K → l → z → z' → m → n → o → E → V → w → x → y → C → F → G → H → r → S → t → u → i → p → q



Conditions when we move toward right

If left is not exist then move toward right] left child is null

OR

If left subtree is completely travelled] → Manually

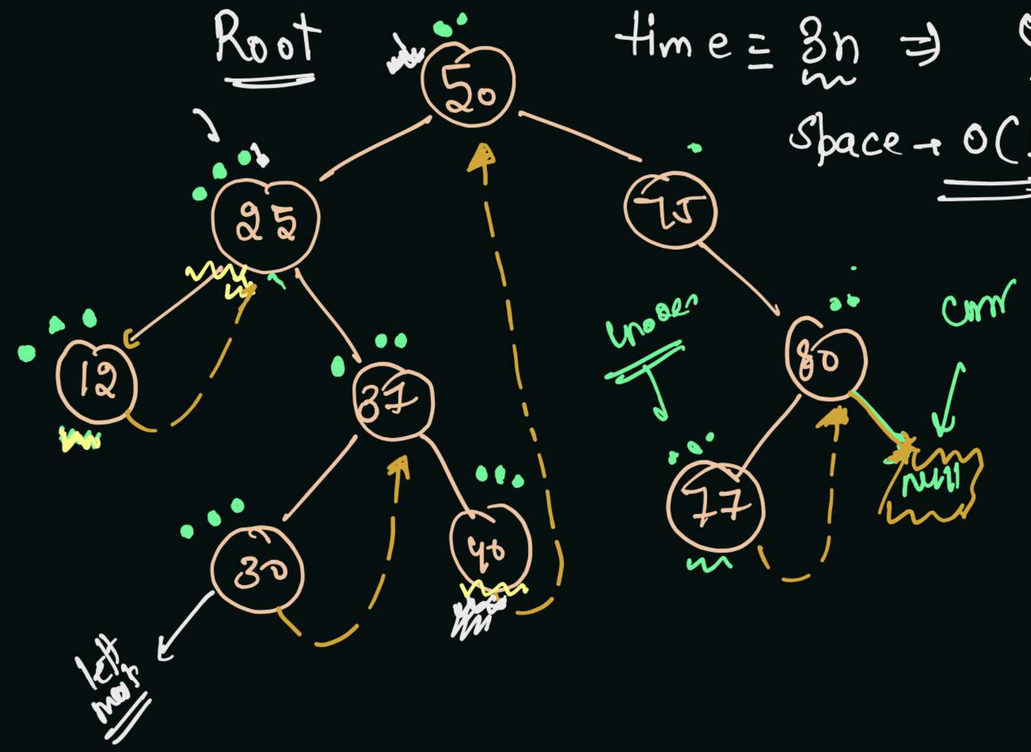
Something is required to know that left subtree is completely processed or not. → marker

without space ??

80, l r

75, l r

50, l r



time = $3n \Rightarrow O(n)$

Space → $O(1)$

Printing -

a)- If left is null → print curr.val

b)- If left subtree is completely processed. } → thread Breaking -

thread is already created in inorder traversal

Movement and marking -

a)- left is null, then move toward right, curr = curr.right

b)- left subtree is completely processed

Return → rightmost Node of leftNode of curr

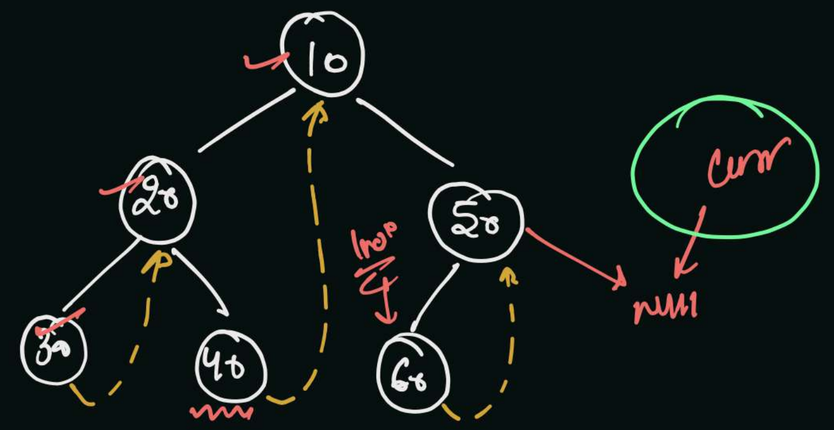
How to find rightmost of a node ??

while(temp.right != null && temp.right != curr)
temp = temp.right } → temp

rightmost Node -

if (rightmost Node.right = null) {
// create thread
} else { → left subtree is completely process
}

12, 25, 30, 37, 40, 50, 75, 77, 80




```java
public static ArrayList<Integer> morrisInTraversal(TreeNode node) {
    ArrayList<Integer> ans = new ArrayList<>();
    TreeNode curr = node;
    while(curr != null) {
        TreeNode leftNode = curr.left;
        if(leftNode != null) {
            TreeNode rightMostNode = getRightMostNode(leftNode, curr);
            if(rightMostNode.right != curr) {
                // create a thread and move toward left tree
                rightMostNode.right = curr; // thread creation
                curr = curr.left;
            } else {
                // if rightmostnode.right == curr that means left subt
                // 1. print the value
                ans.add(curr.val);
                // 2. break thread
                rightMostNode.right = null;
                // 3. move toward right
                curr = curr.right;
            }
        } else {
            // 1. print value
            ans.add(curr.val);
            // 2. move toward right
            curr = curr.right;
        }
    }
    return ans;
}
```

Root

array list
$O(n) + O(\log n)$

Stack

recursion

Inorder

ans → 30, 20, 40, 10, 60, 50

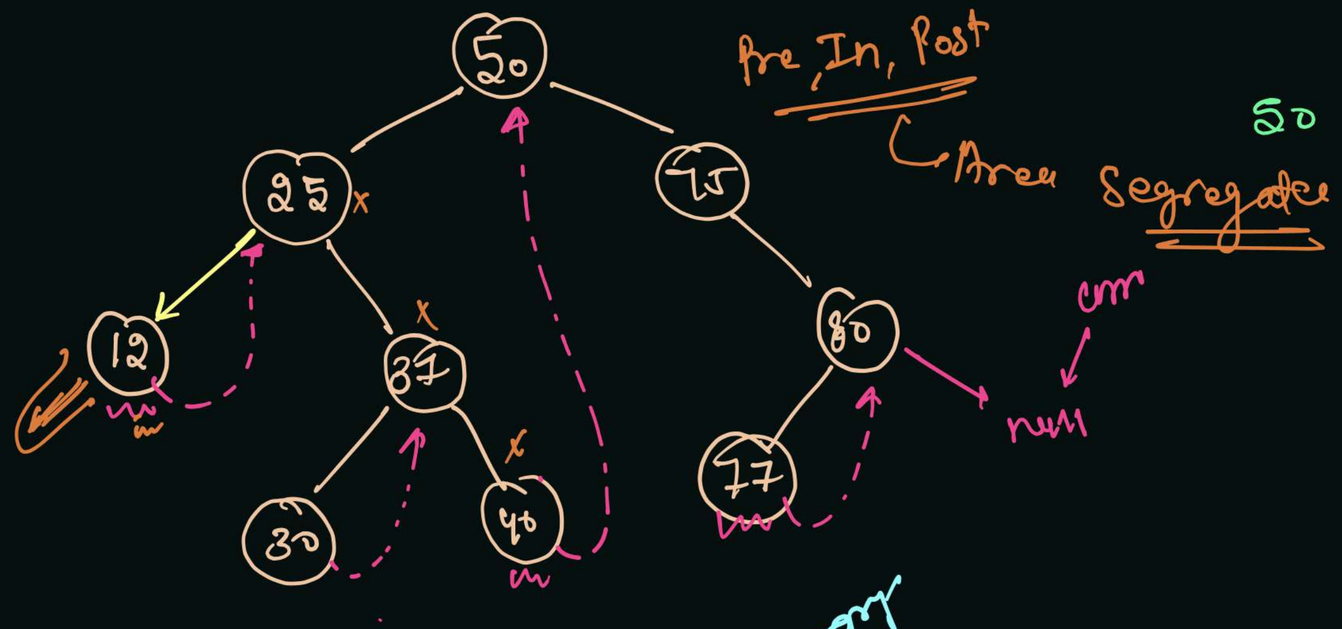
space of answer

PreOrder using Morris Traversal →

50, 25, 12, 37, 30, 40, 75, 80, 77

→ preorder ||

50 25 12 37 30 40 75 80 77



Pre, In, Post

↳ Tree Segregate

① condition for printing temporary

a)- as soon as visit node, create the thread.

b)- if leftnode is null

② Is left subtree process or not

a)- left node = null

b)- hightmost of leftnode . right == curr

③ when to move right subtree

a)- if left is complete Process

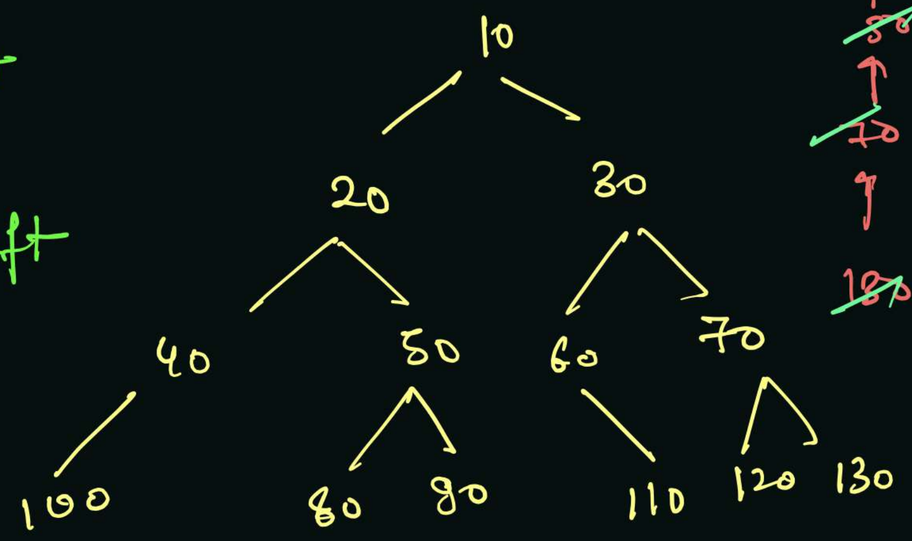
b)- left is not available

Post Order Traversal using Morris Traversal →

Post order → ~~100~~ → ~~40~~ → ~~80~~ → ~~90~~ → ~~50~~ → ~~20~~ → ~~110~~ → ~~60~~ → ~~120~~ → ~~130~~ → ~~70~~ → ~~30~~ → ~~10~~

Normal Preorder → Node left Right

Rev Euler Preorder → Node Right left



= Reverse (Reverse Eulere Preorder)

= Reverse (Node Right left)

= left Right Node

⇓

Post Order

Reverse Euler] Preorder

A ⇒ 10 → 30 → 70 → 130 → 120 → 60 → 110 → 20 → 50 → 90 → 80 → 40 → 100

A' → ~~100~~ → ~~40~~ → ~~80~~ → ~~90~~ → ~~50~~ → ~~20~~ → ~~110~~ → ~~60~~ → ~~120~~ → ~~130~~ → ~~70~~ → ~~30~~ → ~~10~~