37. Morris Traversal | Preorder | Inorder

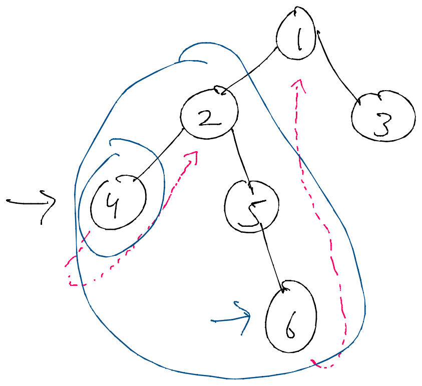
31 March 2022 06:29 PM

Morris Traversal | Inorder | Preorder

$\hookrightarrow$ Threaded Binary Tree

T.C $\Rightarrow O(N)$

S.C $\Rightarrow O(1)$

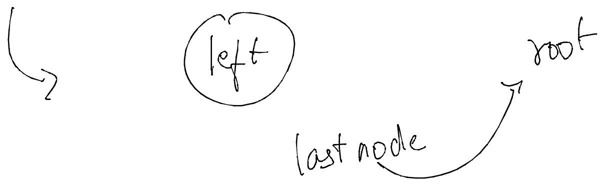


Inorder:

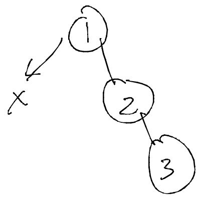
left Root Right

4 2 5 6 1 3

Pattern:



1st Case:



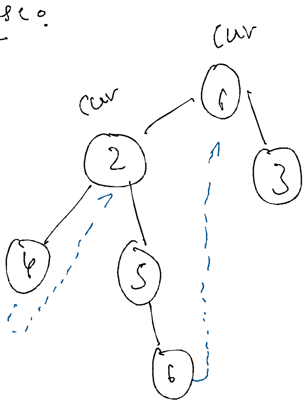
No left

If left → null

print ()

→ right

2nd case:

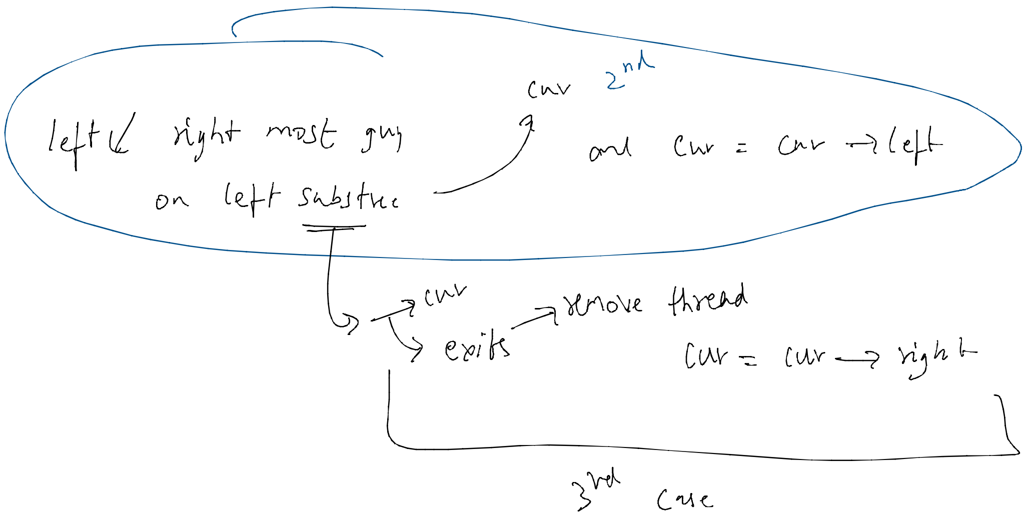


Before going left, whichever is right most guy on left subtree $\xrightarrow{}$ to the curr and then

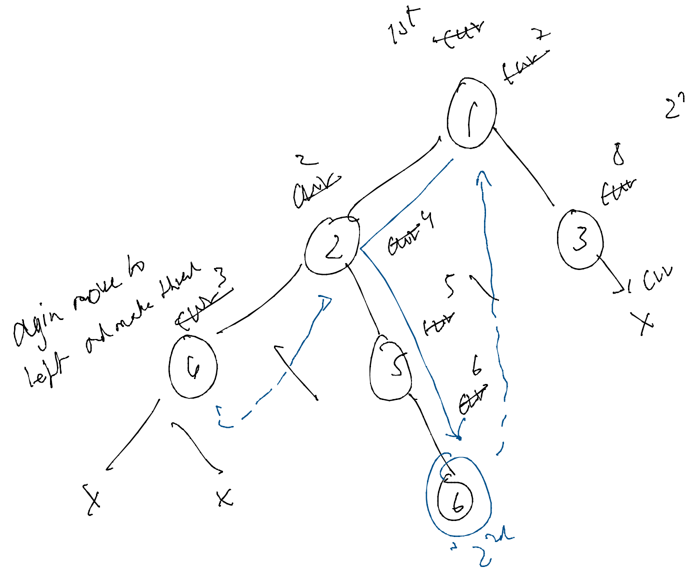
curr = curr → left

3rd Case: If the right most guy already pointing to the Cur, then you remove the thread. Then cur = cur → right.

2nd / 3rd Case



In-order Morris Traversal:
1st case: if left is null, print current node and go right.
2nd case: before going left, make right most node on left subtree connected to current node, then go left.
3rd case: if thread is already pointed to current node, then remove the thread.



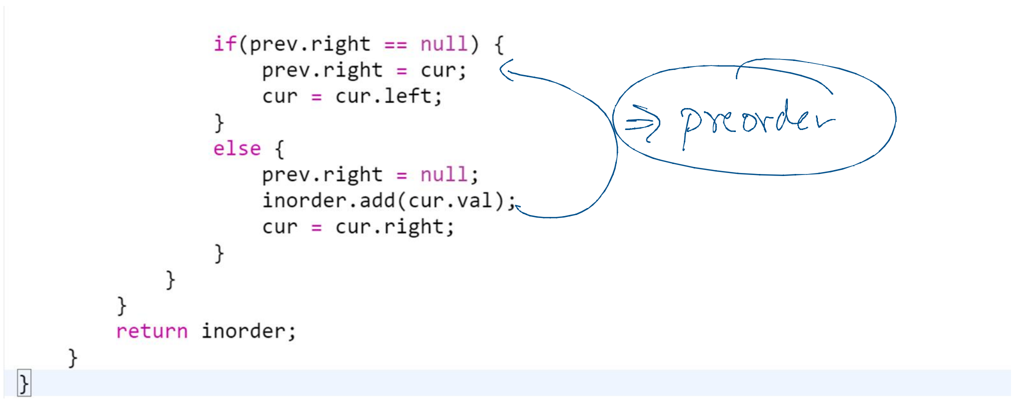
2nd Case ⇒ move to the left subtree right most guy and make thread
then move the cur = cur → left

4 2 5 6 1 3 output

```java
// Morris Traversal Inorder
class Solution {
    public List<Integer> inorderTraversal(TreeNode root) {
        List<Integer> inorder = new ArrayList<Integer>();

        TreeNode cur = root;
        while(cur != null) {
            if(cur.left == null) {
                inorder.add(cur.val);
                cur = cur.right;
            }
            else {
                TreeNode prev = cur.left;
                while(prev.right != null && prev.right != cur) {
                    prev = prev.right;
                }
```

Dry Run



```
                if(prev.right == null) {
                    prev.right = cur;
                    cur = cur.left;
                }
                else {
                    prev.right = null;
                    inorder.add(cur.val);
                    cur = cur.right;
                }
            }
        }
        return inorder;
    }
}
```

⇒ preorder

For preorder: only one change (root left right)