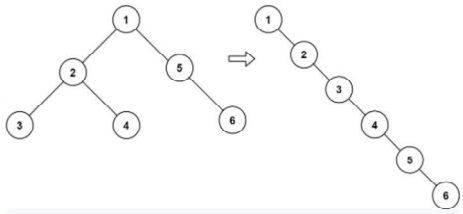
114. Flatten Binary Tree to Linked List

01 April 2022 08:37 AM

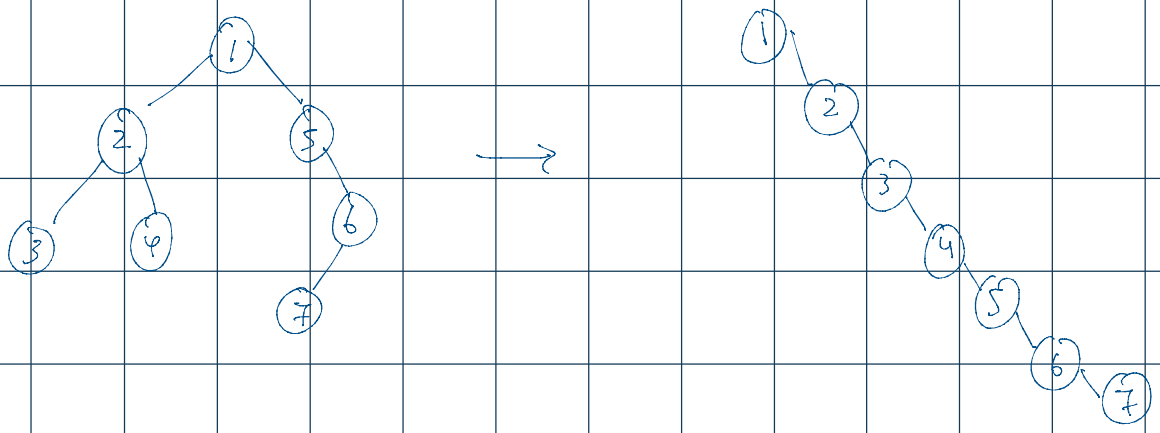
Given the `root` of a binary tree, flatten the tree into a "linked list":

- The "linked list" should use the same `TreeNode` class where the `right` child pointer points to the next node in the list and the `left` child pointer is always `null`.
- The "linked list" should be in the same order as a **pre-order traversal** of the binary tree.

Example 1:

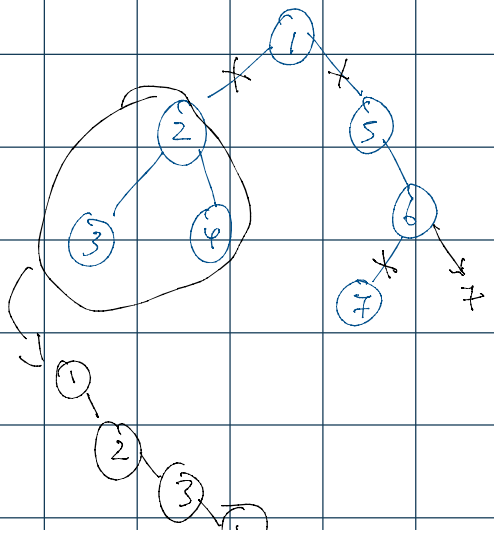


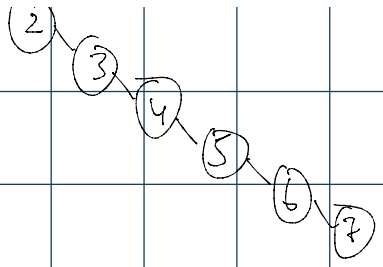
```
Input: root = [1,2,5,3,4,null,6]
Output:
[1,null,2,null,3,null,4,null,5,null,6]
```



PreOrder => 1 2 3 4 5 6 7

Recursive Solution:

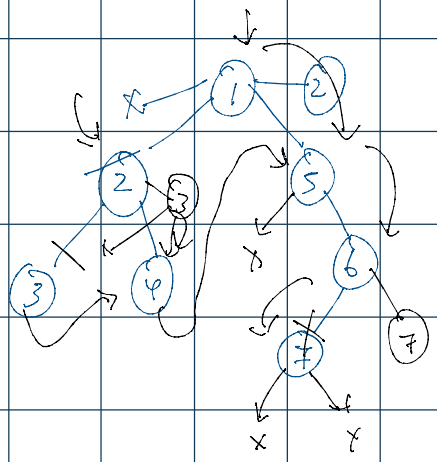




* Do traversal to reach (7) (which is right)

Right left Root (Reverse post order)

→ this direction



prev = ~~null~~ 7 6 5 4 3 2 1

flatten (node)

{ if (node == null) return

flatten (node → right) // first I go to right

flatten (node → left)

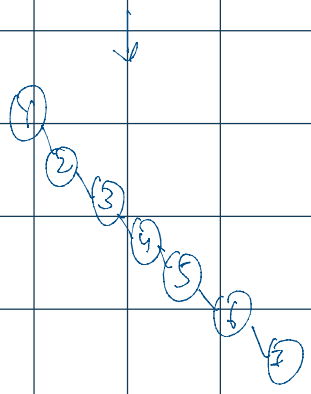
[node → right = prev (when prev = null initialize)

node → left = null]

prev = node

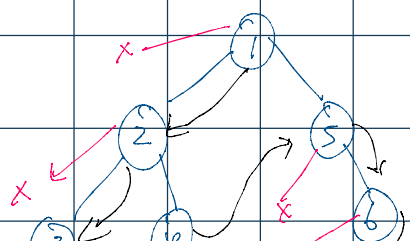
}

T.C ⇒ O(n) S.C ⇒ O(n)



```
// 1st approach -> Recursive Approach
// T.C -> O(N)
// S.C -> O(N)
class Solution {
    TreeNode prev = null;
    public void flatten(TreeNode root) {
        if(root == null) return;

        flatten(root.right);
        flatten(root.left);

        root.right = prev;
        root.left = null;

        prev = root;
    }
}
```

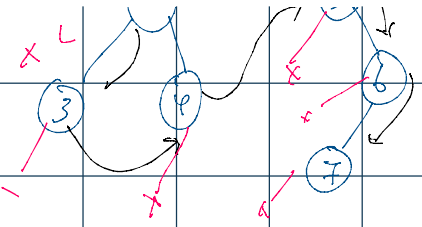
2) Iterative Solution using Stack



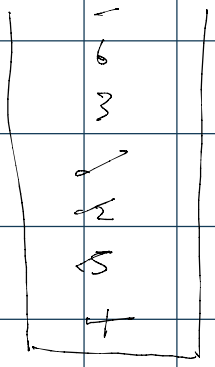
white ⇒ right

red ⇒ left

CW → 1 2 3 ~~4~~ ~~5~~ 6 7



cur → 1 2 3 4 5 6 -



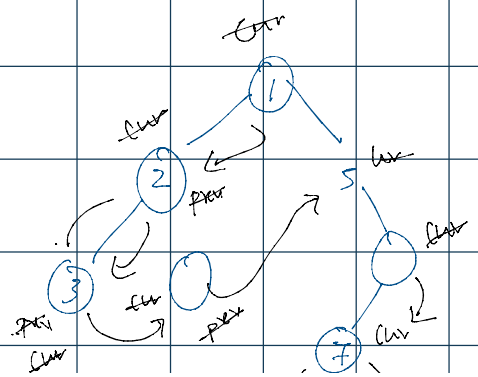

```
St.push(root)
while(!st.empty())
{
    cur = st.top(); st.pop();
    if(cur.right) St.push(cur.right)
    if(cur.left)  st.push(cur.left)
    if(!st.empty()) cur.right = st.top()
    cur.left = null
}
```

```java
// TC - O(N)
// SC - O(N)
// Iterative
class Solution {
    public void flatten(TreeNode root) {
        if(root == null) return;

        Deque<TreeNode> st = new ArrayDeque<>();
        st.push(root);
        while(!st.isEmpty()) {
            TreeNode cur = st.peek();
            st.pop();

            if(cur.right != null) {
                st.push(cur.right);
            }
            if(cur.left != null) {
                st.push(cur.left);
            }
```

T.C → O(N)

S.C → O(N)

3rd Approach : Morris



Left Subtree (find out last guy of preorder which is 4)

and then connect it to right subtree

curr = 1 2 4 5 6 7 null

prev = 2 4 3 7

```
Curr = root
while (curr != null)
{ if (cur. left != null)
  { prev = cur. left
    while (prev. right)
      prev = prev. right
    prev. right = cur. right   (4 → 5)   (3 → 4
    cur. right = cur. left     (1 → 2)   (2 → 3)
    cur. left  = null
  }
  cur = cur. right
}
```

T.C ⇒ O(n)

S.C ⇒ O(1)