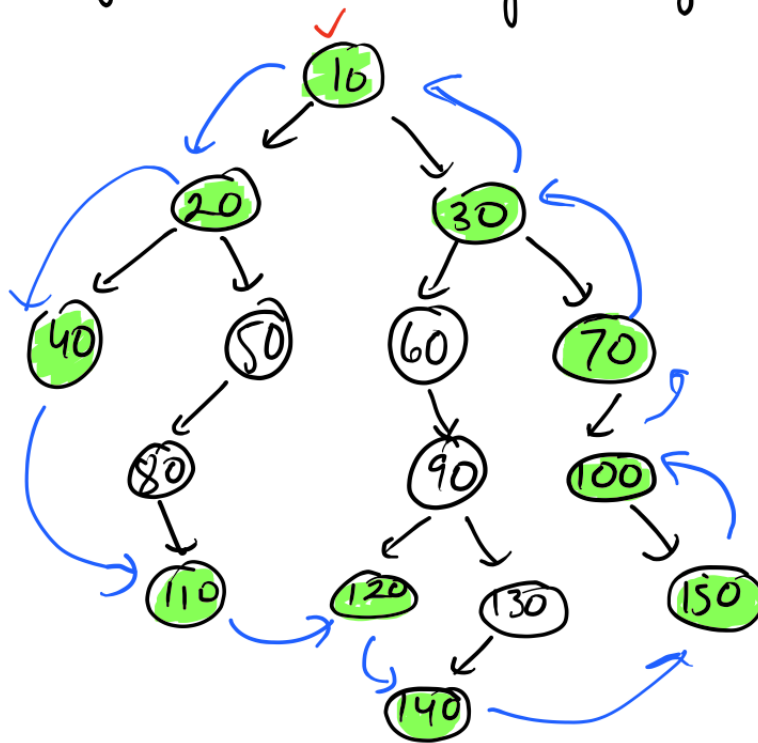


Boundary Traversal of Binary Tree



Boundary ??
root Left Leaf

O/P : 10, 20, 40, 110, 120, 140, 150

100, 70, 30

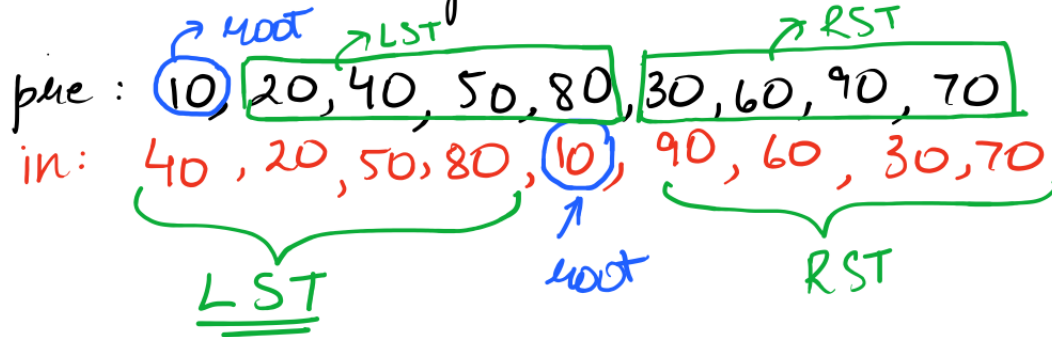
↳ Right Boundary

- print Root
- print Left Boundary
- print Leaf
- print Right Boundary

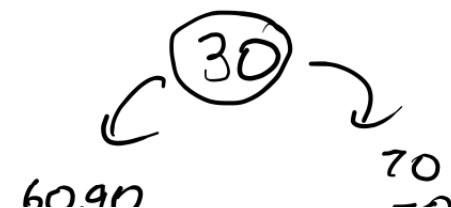
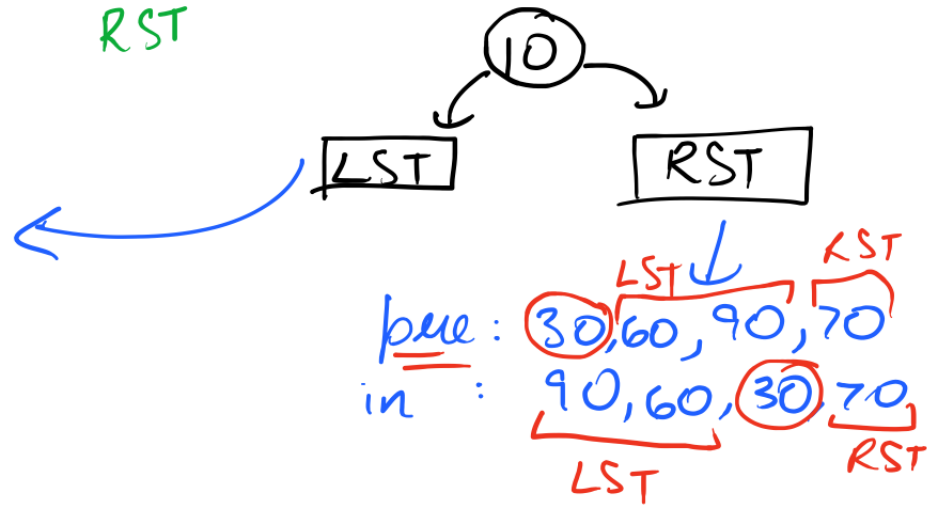
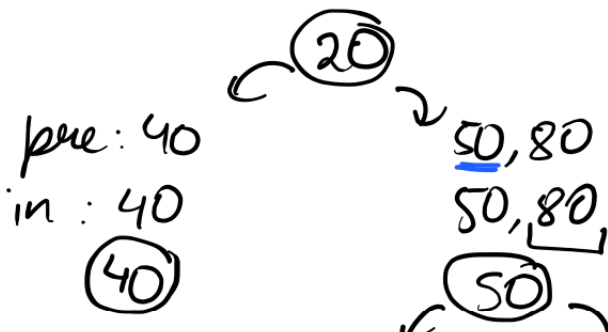
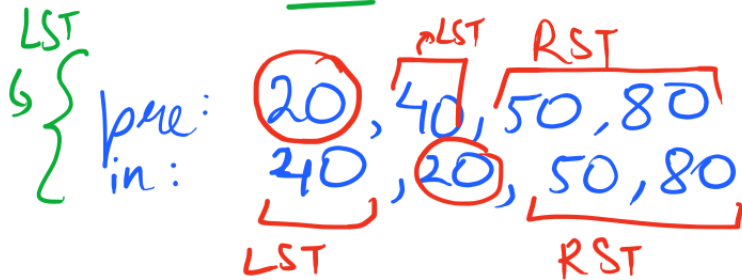


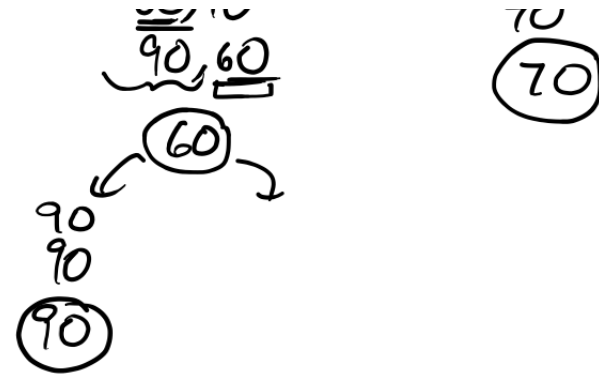
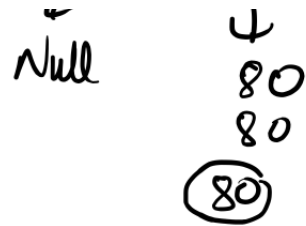


Construct Tree from Preorder and Inorder Traversal

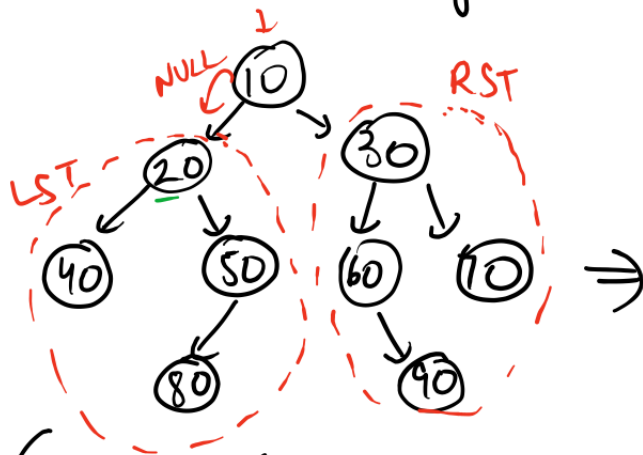


NLR
LNR





Flatten a Binary Tree

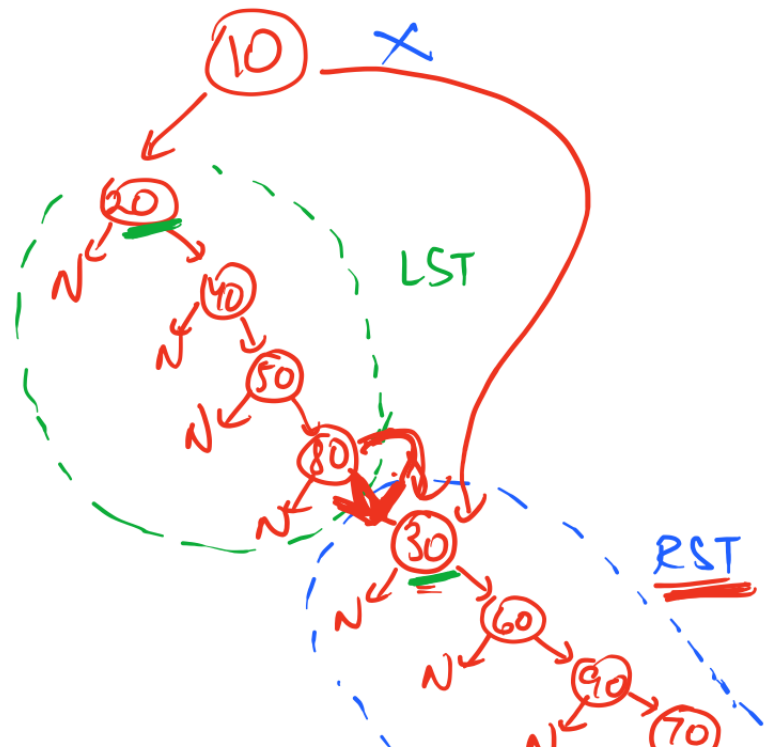


Flatten the BT

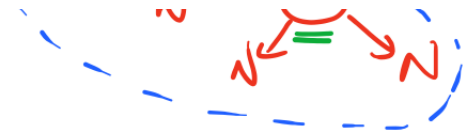
void flattenTree(Node root) {

// Right

// Left



3 $1 \rightarrow \text{root}$



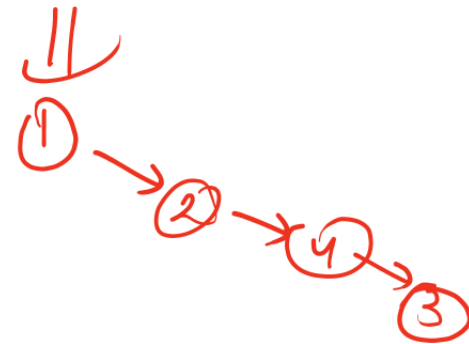
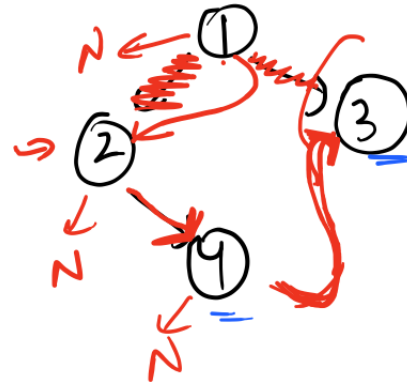
```
static Node lastVisited;

public static void flatten(Node root) {
    lastVisited = null;
    flattenTree(root);
}

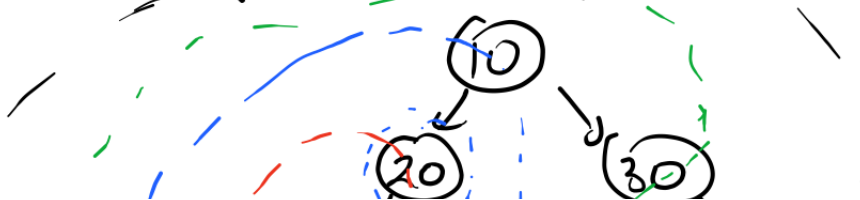
private static void flattenTree(Node root) {
    if (root == null) {
        return;
    }
    flattenTree(root.right);
    flattenTree(root.left);

    // flatten the current node
    root.left = null;
    root.right = lastVisited;
    lastVisited = root;
}
```

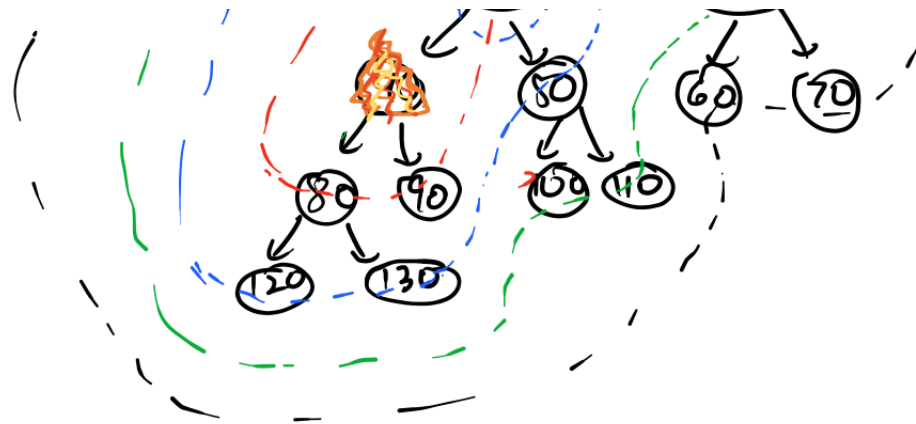
①
LV = null



Burning the Binary Tree



Adjacents
↳ Left
↳ Right

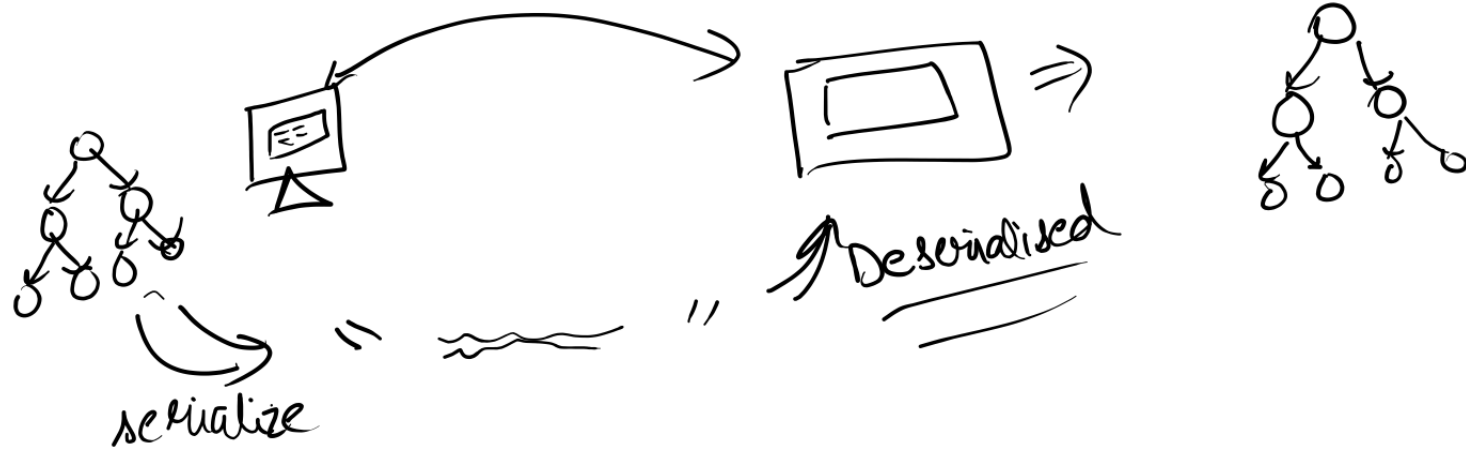


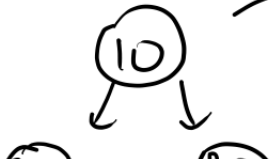
→ Right
 ↳ Parent H.M. → {child → Parent}

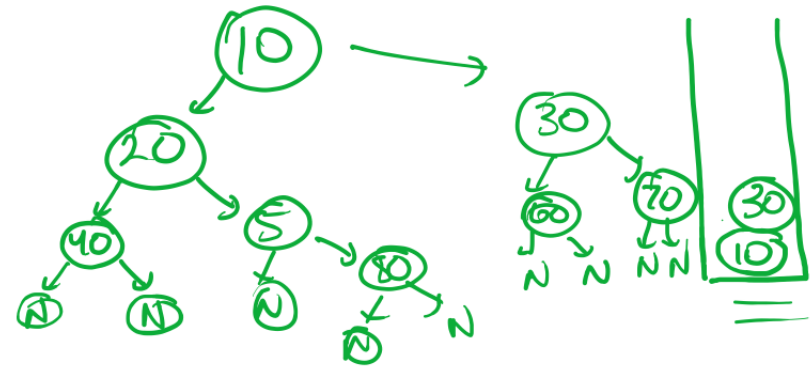
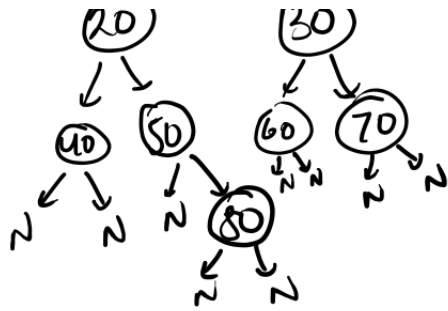
↳ BFS from Source Node

↳ visited set

↳ To help visit only unvisited




 → { 10, 20, 40, N, N, 5, N, 80, N, N, 30, 60, N, N, 70, N, N }



✓
 ↳ CW ✓
 ↳ HW
 } Tips
 +
LeetCode ✓