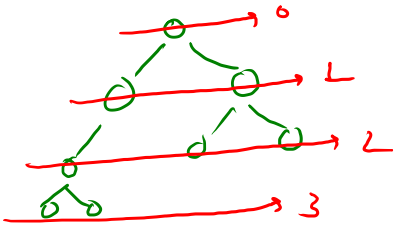
Traversals in B.T.

↳ (1) Preorder Traversal → Node left right.

↳ (2) Postorder Traversal → left right node.

↳ (3) Inorder Traversal → left node right.

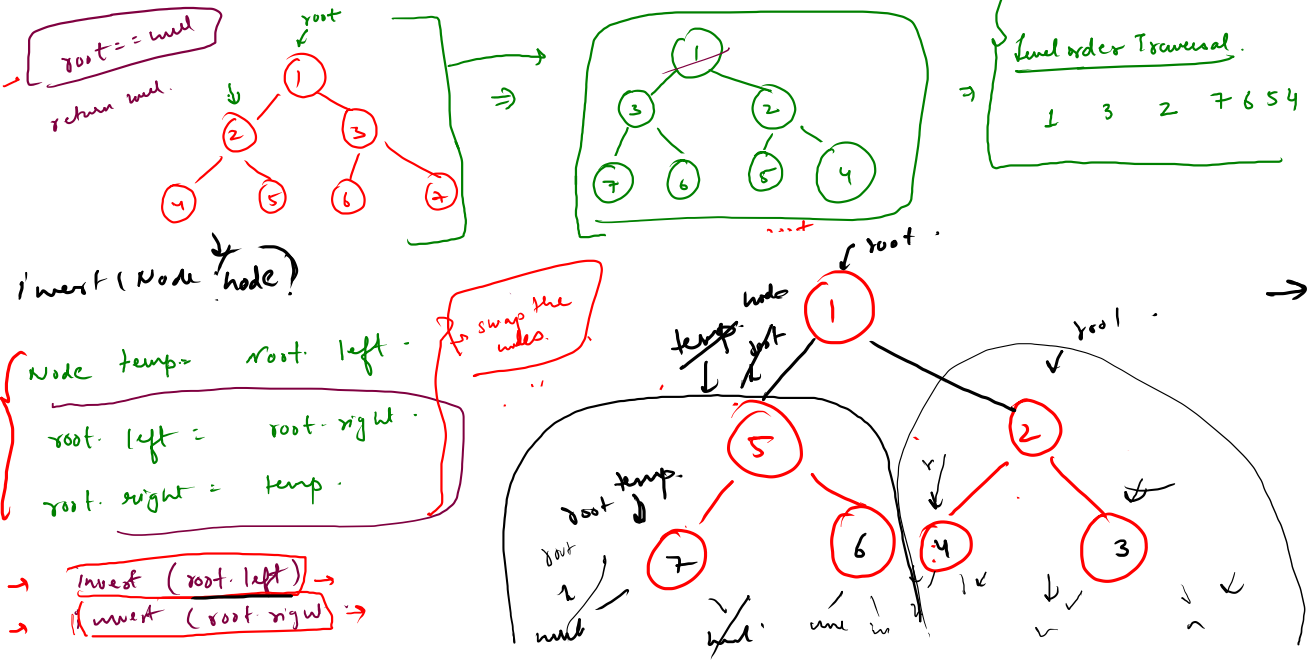
↳ (4) Level order Traversal →



Invert Binary Tree (Day 43)

Problem | Submissions | Leaderboard | Discussions

Given preorder of binary tree first you need to construct binary tree now you have root of a constructed binary tree, invert these binary tree, and return level order of these tree. Example

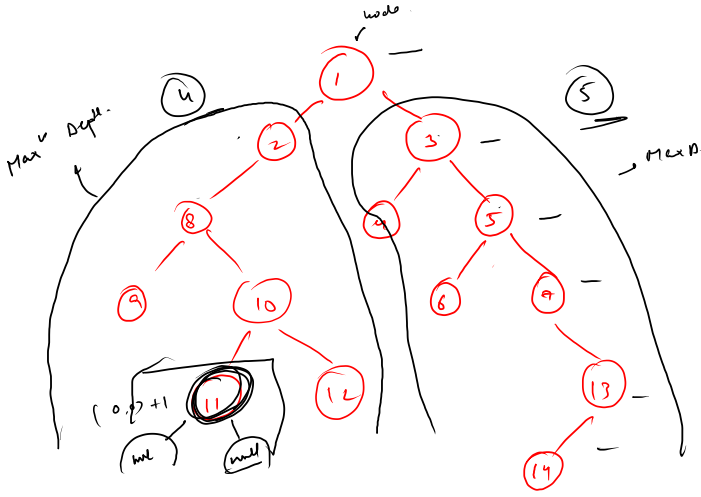


```java
public static Node invertTree(Node root){
    if(root == null){
        return null;
    }

    Node temp = root.left;
    root.left = root.right;
    root.right = temp;

    invertTree(root.left);
    invertTree(root.right);

    return root;

}
```

→ 2-3 coding

→ 2-3 dry run

B1.



if root == null
return 0;

→ MaxDepth.

(4,5) +1

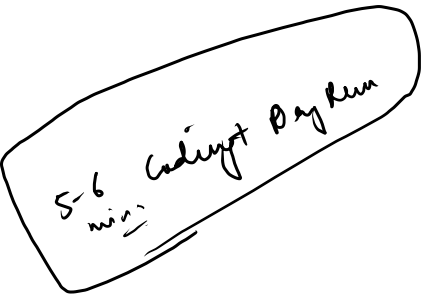
6

→ if height → consider edge.

→ base case
return -1;

→ if depth → Consider nodes.

→ base case
return 0;




```java
public static int  maximumDepth(Node root){
     if(root == null) {
         return 0;
     }

    int leftMaxDepth = maximumDepth(root.left);
    int rightMaxDepth = maximumDepth(root.right);

    int ans = Math.max(leftMaxDepth,rightMaxDepth)+1;
    return ans;
}
```

Sum of Root To Leaf Binary Numbers (Day 43)

Problem | Submissions | Leaderboard | Discussions

Given preorder of binary tree first you need to construct binary tree, Now You have root of a binary tree where each node has a value 0 or 1. Each root-to-leaf path represents a binary number starting with the most significant bit.

For example, if the path is 0 -> 1 -> 1 -> 0 -> 1, then this could represent 01101 in binary, which is 13. For all leaves in the tree, consider the numbers represented by the path from the root to that leaf. Return the sum of these numbers.

Input Format

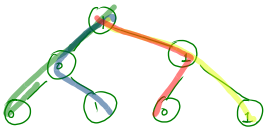
15

1 0 0 n n 1 n n 1 0 n n 1 n n

Output: 22

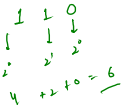
Explanation: (100) + (101) + (110) + (111) = 4 + 5 + 6 + 7 = 22

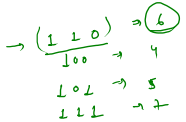
Constraints

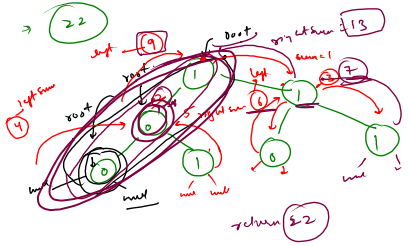


0 , 1

root - leaf.







sum = 0

if(root == null)

return 0

sum = 0 * 2 + 1

sum = 1 * 2 + 0

sum = 2 * 2 + 0

sum = 4 * 2 + 1 = 5

sum = 2*1 + 1 = 3

sum = 2 * 3 + 0 = 6

sum = 2 * 3 + 1 = 7

left sum & right sum denotes sum from root to leaf.

sum → current sum to a particular node

sum = (2 * sum) + root.data;

if (root.left == null && root.right == null) return sum;

int leftsum = fn(root.left, sum)

int rightsum = fn(root.right, sum);

return leftsum + rightsum;

9 + 13 = 22

4 + 5 = 9

6 + 7 = 13

main fn.

sum = 2*0 + 1 = 1

sum = 2 * 1 + 0 = 2

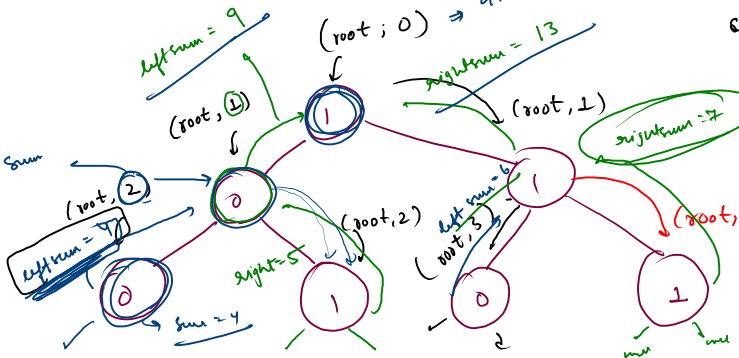
sum = 2*2 + 0 = 4

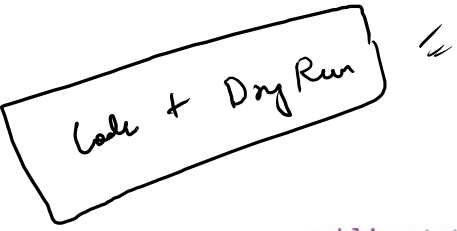
sum = 2*2 + 1 = 5

sum = 2*1 + 1 = 3

sum = 2*3 + 0 = 6

sum = 2*3 + 1 = 7.






```java
public static int  binarySum(Node root, int sum){
     if(root == null) {
         return 0;
     }

    sum = (2*sum)+root.data;

    if(root.left == null && root.right == null){
        return sum;
    }

    int leftSum = binarySum(root.left,sum);
    int rightSum = binarySum(root.right,sum);

    return leftSum+rightSum;
}
```