Problem 156: Binary Tree Upside Down

Problem Information

Difficulty: Medium
Acceptance Rate: 65.04%
Paid Only: Yes
Tags: Tree, Depth-First Search, Binary Tree

Problem Description

Given the `root` of a binary tree, turn the tree upside down and return _the new root_.

You can turn a binary tree upside down with the following steps:

1. The original left child becomes the new root. 2. The original root becomes the new right child. 3. The original right child becomes the new left child.

![](https://assets.leetcode.com/uploads/2020/08/29/main.jpg)

The mentioned steps are done level by level. It is **guaranteed** that every right node has a sibling (a left node with the same parent) and has no children.

Example 1:

![](https://assets.leetcode.com/uploads/2020/08/29/updown.jpg)

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

Example 2:

Input: root = [] **Output:** []

Example 3:

Input: root = [1] **Output:** [1]

Constraints:

* The number of nodes in the tree will be in the range `[0, 10]`. * `1 <= Node.val <= 10` * Every right node in the tree has a sibling (a left node that shares the same parent). * Every right node in the tree has no children.

Code Snippets

C++:

```
/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* TreeNode *left;
* TreeNode *right;
* TreeNode() : val(0), left(nullptr), right(nullptr) {}
* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
* TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left),
right(right) {}
* };
*/
class Solution {
public:
TreeNode* upsideDownBinaryTree(TreeNode* root) {

}
};
```

Java:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* int val;
* TreeNode left;
* TreeNode right;
* TreeNode() {}
* TreeNode(int val) { this.val = val; }
* TreeNode(int val, TreeNode left, TreeNode right) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
class Solution {
public TreeNode upsideDownBinaryTree(TreeNode root) {

}
}
```

Python3:

```
# Definition for a binary tree node.
# class TreeNode:
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution:
def upsideDownBinaryTree(self, root: Optional[TreeNode]) ->
Optional[TreeNode]:
```