

Binary Tree Tilt

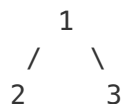
Given a binary tree, return the tilt of the **whole tree**.

The tilt of a **tree node** is defined as the **absolute difference** between the sum of all left subtree node values and the sum of all right subtree node values. Null node has tilt 0.

The tilt of the **whole tree** is defined as the sum of all nodes' tilt.

Example:

Input:



Output: 1

Explanation:

Tilt of node 2 : 0

Tilt of node 3 : 0

Tilt of node 1 : $|2-3| = 1$

Tilt of binary tree : $0 + 0 + 1 = 1$

Note:

1. The sum of node values in any subtree won't exceed the range of 32-bit integer.
2. All the tilt values won't exceed the range of 32-bit integer.

Solution 1

```
public class Solution {
    int result = 0;

    public int findTilt(TreeNode root) {
        postOrder(root);
        return result;
    }

    private int postOrder(TreeNode root) {
        if (root == null) return 0;

        int left = postOrder(root.left);
        int right = postOrder(root.right);

        result += Math.abs(left - right);

        return left + right + root.val;
    }
}
```

written by [shawngao](#) original link [here](#)

Solution 2

```
public class Solution {  
  
    int tilt = 0;  
  
    public int findTilt(TreeNode root) {  
        postorder(root);  
        return tilt;  
    }  
  
    public int postorder(TreeNode root) {  
        if (root == null) return 0;  
        int leftSum = postorder(root.left);  
        int rightSum = postorder(root.right);  
        tilt += Math.abs(leftSum - rightSum);  
        return leftSum + rightSum + root.val;  
    }  
  
}
```

written by [compton_scatter](#) original link [here](#)

Solution 3

```
class Solution {
public:
    int findTilt(TreeNode* root) {
        if(root == NULL) return 0;

        int res = 0;

        postorder(root, res);

        return res;
    }
private:
    int postorder(TreeNode* root, int& res){
        if(root == NULL) return 0;

        int leftsum= postorder(root->left, res);

        int rightsum = postorder(root->right, res);

        res += abs(leftsum - rightsum);

        return leftsum + rightsum + root->val;
    }
};
```

written by [Sublele](#) original link [here](#)

From [Leetcoder](#).