Problem 2471: Minimum Number of Operations to Sort a Binary Tree by Level

Problem Information

Difficulty: Medium
Acceptance Rate: 74.27%
Paid Only: No
Tags: Tree, Breadth-First Search, Binary Tree

Problem Description

You are given the `root` of a binary tree with **unique values**.

In one operation, you can choose any two nodes **at the same level** and swap their values.

Return _the minimum number of operations needed to make the values at each level sorted in a**strictly increasing order**_.

The **level** of a node is the number of edges along the path between it and the root node _._

Example 1:

![](https://assets.leetcode.com/uploads/2022/09/18/image-20220918174006-2.png)

Input: root = [1,4,3,7,6,8,5,null,null,null,null,9,null,10] **Output:** 3 **Explanation:** - Swap 4 and 3. The 2nd level becomes [3,4]. - Swap 7 and 5. The 3rd level becomes [5,6,8,7]. - Swap 8 and 7. The 3rd level becomes [5,6,7,8]. We used 3 operations so return 3. It can be proven that 3 is the minimum number of operations needed.

Example 2:

![](https://assets.leetcode.com/uploads/2022/09/18/image-20220918174026-3.png)

Input: root = [1,3,2,7,6,5,4] **Output:** 3 **Explanation:** - Swap 3 and 2. The 2nd level becomes [2,3]. - Swap 7 and 4. The 3rd level becomes [4,6,5,7]. - Swap 6 and 5. The 3rd level becomes [4,5,6,7]. We used 3 operations so return 3. It can be proven that 3 is the minimum number of operations needed.

Example 3:

![](https://assets.leetcode.com/uploads/2022/09/18/image-20220918174052-4.png)

Input: root = [1,2,3,4,5,6] **Output:** 0 **Explanation:** Each level is already sorted in increasing order so return 0.

Constraints:

* The number of nodes in the tree is in the range `[1, 105]`. * `1 <= Node.val <= 105` * All the values of the tree are **unique**.

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:
int minimumOperations(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 int minimumOperations(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 minimumOperations(self, root: Optional[TreeNode]) -> int:
```