Problem 99: Recover Binary Search Tree

Problem Information

Difficulty: Medium
Acceptance Rate: 58.02%
Paid Only: No
Tags: Tree, Depth-First Search, Binary Search Tree, Binary Tree

Problem Description

You are given the `root` of a binary search tree (BST), where the values of **exactly** two nodes of the tree were swapped by mistake. _Recover the tree without changing its structure_.

Example 1:

Input: root = [1,3,null,null,2] **Output:** [3,1,null,null,2] **Explanation:** 3 cannot be a left child of 1 because 3 > 1. Swapping 1 and 3 makes the BST valid.

Example 2:

Input: root = [3,1,4,null,null,2] **Output:** [2,1,4,null,null,3] **Explanation:** 2 cannot be in the right subtree of 3 because 2 < 3. Swapping 2 and 3 makes the BST valid.

Constraints:

* The number of nodes in the tree is in the range `[2, 1000]`. * `-231 <= Node.val <= 231 - 1`

Follow up: A solution using `O(n)` space is pretty straight-forward. Could you devise a constant `O(1)` space solution?

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:
void recoverTree(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 void recoverTree(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 recoverTree(self, root: Optional[TreeNode]) -> None:
"""
Do not return anything, modify root in-place instead.
"""
```