Problem 1382: Balance a Binary Search Tree

Problem Information

Difficulty: Medium
Acceptance Rate: 84.69%
Paid Only: No
Tags: Divide and Conquer, Greedy, Tree, Depth-First Search, Binary Search Tree, Binary Tree

Problem Description

Given the `root` of a binary search tree, return _a**balanced** binary search tree with the same node values_. If there is more than one answer, return **any of them**.

A binary search tree is **balanced** if the depth of the two subtrees of every node never differs by more than `1`.

Example 1:

Input: root = [1,null,2,null,3,null,4,null,null] **Output:** [2,1,3,null,null,null,4]
Explanation: This is not the only correct answer, [3,1,4,null,2] is also correct.

Example 2:

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

Constraints:

* The number of nodes in the tree is in the range `[1, 104]`. * `1 <= Node.val <= 105`

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* balanceBST(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 balanceBST(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 balanceBST(self, root: Optional[TreeNode]) -> Optional[TreeNode]:
```