538 Convert BST to Greater Tree

2018年4月14日 14:14

Given a Binary Search Tree (BST), convert it to a Greater Tree such that every key of the original BST is changed to the original key plus sum of all keys greater than the original key in BST.

Example:

Input: The root of a Binary Search Tree like this:

```
      5
    /   \
   2     13
```

Output: The root of a Greater Tree like this:

```
     18
    /   \
   20    13
```

来自 <https://leetcode.com/problems/convert-bst-to-greater-tree/description/>

给定一个二叉搜索树（Binary Search Tree），把它转换成为累加树（Greater Tree)，使得每个节点的值是原来的节点值加上所有大于它的节点值之和。

Solution for Python3:

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None
class Solution:
    def __init__(self):
        self.total = 0
    def convertBST(self, root):
        """
        :type root: TreeNode
        :rtype: TreeNode
        """
```

```
15          if root:
16              self.convertBST(root.right)
17              self.total += root.val
18              root.val = self.total
19              self.convertBST(root.left)
20          return root
```

Solution for C++:

```
1 /**
2  * Definition for a binary tree node.
3  * struct TreeNode {
4  *     int val;
5  *     TreeNode *left;
6  *     TreeNode *right;
7  *     TreeNode(int x) : val(x),
8 left(NULL), right(NULL) {}
9  * };
10  */
11 class Solution1 {
12 private:
13     int sum = 0;
14 public:
15     TreeNode* convertBST(TreeNode* root) {
16         if (root != NULL) {
17             convertBST(root->right);
18             sum += root->val;
19             root->val = sum;
20             convertBST(root->left);
21         }
22         return root;
23     }
24 };
25
26 class Solution2 {
```

```
27 private:
28     int sum = 0;
29 public:
30     TreeNode* convertBST(TreeNode* root) {
31         stack<TreeNode*> s;
32         TreeNode* cur = root;
33         int sum = 0;
34         while (!s.empty() || cur) {
35             if (cur) {
36                 s.push(cur);
37                 cur = cur->right;
38             } else {
39                 cur = s.top()->left;
40                 int tmp = sum;
41                 sum += s.top()->val;
42                 s.top()->val += tmp;
43                 s.pop();
44             }
45         }
46         return root;
47     }
   };
```