501 Find Mode in Binary Search Tree

2018年4月13日 14:18

Given a binary search tree (BST) with duplicates, find all the mode(s) (the most frequently occurred element) in the given BST.
Assume a BST is defined as follows:

- The left subtree of a node contains only nodes with keys **less than or equal to** the node's key.
- The right subtree of a node contains only nodes with keys **greater than or equal to** the node's key.
- Both the left and right subtrees must also be binary search trees.

For example:
Given BST [1,null,2,2],

```
   1
    \
     2
    /
   2
```

return [2].
Note: If a tree has more than one mode, you can return them in any order.
Follow up: Could you do that without using any extra space? (Assume that the implicit stack space incurred due to recursion does not count).

来自 <https://leetcode.com/problems/find-mode-in-binary-search-tree/description/>

Solution for Python3:

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None

class Solution:
    def findMode(self, root):
        """
        :type root: TreeNode
        :rtype: List[int]
        """
        self.currVal = None
        self.maxCount = 0
        self.currCount = 0
        self.modeCount = 0
        self.modes = None
```

```python

        self.inOrder(root)

        self.modes = [0]*self.modeCount
        self.modeCount = 0
        self.currCount = 0
        self.inOrder(root)
        return self.modes

    def handleValue(self, val):
        if val != self.currVal:
            self.currVal = val
            self.currCount = 0
        self.currCount += 1
        if self.currCount > self.maxCount:
            self.maxCount = self.currCount
            self.modeCount = 1
        elif self.currCount == self.maxCount:
            if self.modes:
                self.modes[self.modeCount] = self.currVal
            self.modeCount += 1

    def inOrder(self, root):
        if not root:
            return;
        self.inOrder(root.left)
        self.handleValue(root.val)
        self.inOrder(root.right)
```

Solution for C++:

```cpp
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    vector<int> findMode(TreeNode* root) {
        inorder(root);
```

```
            modes.resize(modeCount);
            modeCount = 0;
            currCount = 0;
            inorder(root);
            return modes;
        }
        int currVal;
        int currCount = 0;
        int maxCount = 0;
        int modeCount = 0;

        vector<int> modes;

        void handleValue(int val) {
            if (val != currVal) {
                currVal = val;
                currCount = 0;
            }
            currCount++;
            if (currCount > maxCount) {
                maxCount = currCount;
                modeCount = 1;
            } else if (currCount == maxCount) {
                if (modes.size())
                    modes[modeCount] = currVal;
                modeCount++;
            }
        }

        void inorder(TreeNode* root) {
            if (root == NULL)
                return;
            inorder(root->left);
            handleValue(root->val);
            inorder(root->right);
        }
    };
```