Problem 501: Find Mode in Binary Search Tree

Problem Information

Difficulty: Easy
Acceptance Rate: 58.15%
Paid Only: No
Tags: Tree, Depth-First Search, Binary Search Tree, Binary Tree

Problem Description

Given the `root` of a binary search tree (BST) with duplicates, return _all the[mode(s)](https://en.wikipedia.org/wiki/Mode_\(statistics\)) (i.e., the most frequently occurred element) in it_.

If the tree has more than one mode, return them in **any order**.

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.

Example 1:

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

Example 2:

Input: root = [0] **Output:** [0]

Constraints:

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

Follow up: Could you do that without using any extra space? (Assume that the implicit stack space incurred due to recursion does not count).

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:
vector<int> findMode(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[] findMode(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 findMode(self, root: Optional[TreeNode]) -> List[int]:
```