Problem 1373: Maximum Sum BST in Binary Tree

Problem Information

Difficulty: Hard
Acceptance Rate: 45.73%
Paid Only: No
Tags: Dynamic Programming, Tree, Depth-First Search, Binary Search Tree, Binary Tree

Problem Description

Given a **binary tree** `root`, return _the maximum sum of all keys of**any** sub-tree which is also a Binary Search Tree (BST)_.

Assume a BST is defined as follows:

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

Example 1:

Input: root = [1,4,3,2,4,2,5,null,null,null,null,null,null,4,6] **Output:** 20 **Explanation:** Maximum sum in a valid Binary search tree is obtained in root node with key equal to 3.

Example 2:

Input: root = [4,3,null,1,2] **Output:** 2 **Explanation:** Maximum sum in a valid Binary search tree is obtained in a single root node with key equal to 2.

Example 3:

Input: root = [-4,-2,-5] **Output:** 0 **Explanation:** All values are negatives. Return an empty BST.

Constraints:

* The number of nodes in the tree is in the range `[1, 4 * 104]`. * `-4 * 104 <= Node.val <= 4 * 104`

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:
int maxSumBST(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 maxSumBST(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 maxSumBST(self, root: Optional[TreeNode]) -> int:
```