# 描述

给定二叉搜索树的根结点root，返回 L 和 R（含）之间的所有结点的值的和。

二叉搜索树保证具有唯一的值。

**示例 1：**

输入：root = [10,5,15,3,7,null,18], L = 7, R = 15

输出：32

**示例 2：**

输入：root = [10,5,15,3,7,13,18,1,null,6], L = 6, R = 10

输出：23

**提示：**

树中的结点数量最多为 10000个。

最终的答案保证小于 2^31。

# 分析

## 方法一：递归/深度优先搜索

我们对树进行深度优先搜索，对于当前节点 node，如果 node.val 小于等于 L，那么只需要继续搜索它的右子树；如果 node.val 大于等于 R，那么只需要继续搜索它的左子树；如果 node.val 在区间 (L, R) 中，则需要搜索它的所有子树。

**代码：**

public:

int rangeSumBST(TreeNode\* root, int L, int R) {

if(!root) return 0;//若树为空返回0

if(root->val>R)

//若根节点比R还大那只到根节点的左子树中找

return rangeSumBST(root->left,L,R);

else if(root->val<L)

//若根节点比 L还小那只到根节点的右子树中找

return rangeSumBST(root->right,L,R);

else

//若根节点在L与R之间加上根节点的值再到左右子树中找

return root->val+rangeSumBST(root->left,L,R)+

rangeSumBST(root->right,L,R);

}

};

或：

class Solution {

public:

int rangeSumBST(TreeNode\* root, int L, int R) {

if (root == nullptr) {

return 0;

}

if (root->val < L) {

return rangeSumBST(root->right,L,R);

}

if (root->val > R) {

return rangeSumBST(root->left,L,R);

}

return root->val + rangeSumBST(root->left,L,R) + rangeSumBST(root->right,L,R);

}

};