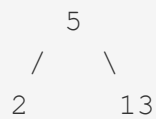


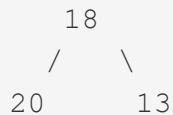
给定一个二叉搜索树 (Binary Search Tree) , 把它转换成为累加树 (Greater Tree), 使得每个节点的值是原来的节点值加上所有大于它的节点值之和。

例如:

输入: 二叉搜索树:



输出: 转换为累加树:



分析, 个人无法从一次递归找到全部解的方法, 只能先统计数字, 然后统计该数字之后的和, 并再进行一次递归遍历得到答案。

```
/**
 * 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:
    void dfs(vector<int> &nums, TreeNode* root) {
        nums.push_back(root->val);
        if(root->left!=NULL) {
            dfs(nums, root->left);
        }
        if(root->right!=NULL) {
            dfs(nums, root->right);
        }
    }

    void add(map<int, int> &sum, TreeNode* root, vector<int> &nums) {
```

```

        int n=nums.size();
        for(int i=0;i<n;i++){
            if(root->val<nums[i]){
                root->val=root->val+sum[n-1]-sum[i-1];
                break;
            }
        }
        if(root->left!=NULL){
            add(sum, root->left, nums);
        }
        if(root->right!=NULL){
            add(sum, root->right, nums);
        }
    }
    TreeNode* convertBST(TreeNode* root) {
        vector<int> nums;
        if(root==NULL){
            return root;
        }
        dfs(nums, root);
        sort(nums.begin(), nums.end());
        map<int, int> sum;
        int n=nums.size();
        for(int i=0;i<n;i++){
            sum[i]=sum[i-1]+nums[i];
        }
        add(sum, root, nums);
        return root;
    }
};

```