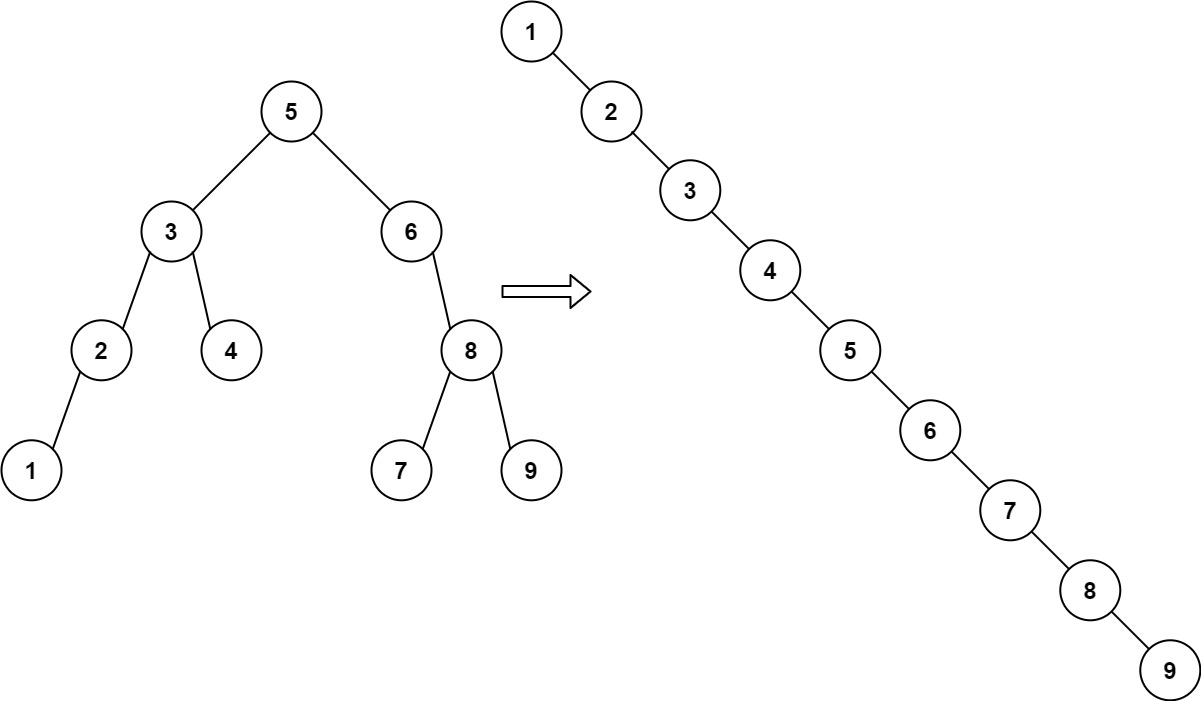
# 题目

给你一棵二叉搜索树，请按中序遍历将其重新排列为一棵递增顺序搜索树，使树中最左边的节点成为树的根节点，并且每个节点没有左子节点，只有一个右子节点。

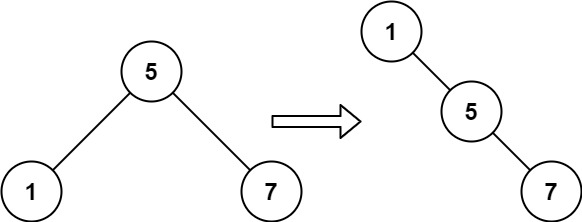
示例 1：



输入：root = [5,3,6,2,4,null,8,1,null,null,null,7,9]

输出：[1,null,2,null,3,null,4,null,5,null,6,null,7,null,8,null,9]

示例 2：



输入：root = [5,1,7]

输出：[1,null,5,null,7]

提示：

树中节点数的取值范围是 [1, 100]

0 <= Node.val <= 1000

# 分析

class Solution {

public:

TreeNode\* increasingBST(TreeNode\* root) {

stack<TreeNode\*> stk;

TreeNode\* cur = root;

TreeNode\* prev = nullptr;

TreeNode\* first = nullptr;

while (cur != nullptr || !stk.empty()) {

while (cur != nullptr) {

stk.push(cur);

cur = cur->left; //先深度遍历到最左侧的叶子节点

}

cur = stk.top(); //该叶子节点作为头结点

stk.pop();

if (prev == nullptr) {

first = cur;

}

else {

prev->right = cur;

}

cur->left = nullptr;

prev = cur;

cur = cur->right;

}

return first; //返回头结点

}

};