1382. Balance a Binary Search Tree <Medium>

/\*\*

\* 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 {

private:

void inorderTraversal(TreeNode\* root, vector<TreeNode\*>& nodes) {

if (root == nullptr)

return;

inorderTraversal(root->left, nodes);

nodes.push\_back(root);

inorderTraversal(root->right, nodes);

}

TreeNode\* buildBST(vector<TreeNode\*>& nodes, int begin, int end) {

if (begin == end) return nullptr;

int mid = begin + (end - begin) / 2;

TreeNode\* root = nodes[mid];

root->left = buildBST(nodes, begin, mid);

root->right = buildBST(nodes, mid + 1, end);

return root;

}

public:

TreeNode\* balanceBST(TreeNode\* root) {

vector<TreeNode\*> nodes;

inorderTraversal(root, nodes);

return buildBST(nodes, 0, nodes.size());

}

};