## 1373. Maximum Sum BST in Binary Tree <Hard>

***Binary Search Dynamic Programming Tree***

class Solution {

public:

int maxSumBST(TreeNode\* root) {

int ans = 0;

dfs(root, ans);

return ans;

}

private:

tuple<int, int, int, bool> dfs(TreeNode\* root, int& ans) {

if (!root) return {INT\_MAX, INT\_MIN, 0, true};

auto [leftMin, leftMax, leftSum, leftValid] = dfs(root->left, ans);

auto [rightMin, rightMax, rightSum, rightValid] = dfs(root->right, ans);

bool valid =

leftValid && rightValid && root->val > leftMax && root->val < rightMin;

int sum = valid ? root->val + leftSum + rightSum : -1;

ans = max(ans, sum);

return {min(root->val, leftMin), max(root->val, rightMax), sum, valid};

}

};

int res = 0;

int maxSumBST(TreeNode\* root) {

int x = recursive(root, root->val, root->val);

return max(res, x);

}

int recursive (TreeNode\* subroot, int& min\_val, int& max\_val) {

int max\_sum = subroot->val;

/\* left \*/

if (subroot->left){

int min\_l, max\_l;

int sum\_l = recursive(subroot->left, min\_l, max\_l);

res = max(res, sum\_l);

if (sum\_l == INT\_MIN || max\_sum == INT\_MIN || max\_l >= subroot->val)

max\_sum = INT\_MIN;

else {

max\_sum += sum\_l;

min\_val = min\_l;

}

}

else

min\_val = subroot->val;

/\* right \*/

if (subroot->right) {

int min\_r, max\_r;

int sum\_r = recursive(subroot->right, min\_r, max\_r);

res = max(res, sum\_r);

if (sum\_r == INT\_MIN || max\_sum == INT\_MIN || min\_r <= subroot->val)

max\_sum = INT\_MIN;

else{

max\_sum += sum\_r;

max\_val = max\_r;

}

}

else

max\_val = subroot->val;

return max\_sum;

}