Problem 1448: Count Good Nodes in Binary Tree

Problem Information

Difficulty: Medium
Acceptance Rate: 73.67%
Paid Only: No
Tags: Tree, Depth-First Search, Breadth-First Search, Binary Tree

Problem Description

Given a binary tree `root`, a node _X_ in the tree is named **good** if in the path from root to _X_ there are no nodes with a value _greater than_ X.

Return the number of **good** nodes in the binary tree.

Example 1:

Input: root = [3,1,4,3,null,1,5] **Output:** 4 **Explanation:** Nodes in blue are **good**. Root Node (3) is always a good node. Node 4 -> (3,4) is the maximum value in the path starting from the root. Node 5 -> (3,4,5) is the maximum value in the path Node 3 -> (3,1,3) is the maximum value in the path.

Example 2:

Input: root = [3,3,null,4,2] **Output:** 3 **Explanation:** Node 2 -> (3, 3, 2) is not good, because "3" is higher than it.

Example 3:

Input: root = [1] **Output:** 1 **Explanation:** Root is considered as **good**.

Constraints:

* The number of nodes in the binary tree is in the range `[1, 10^5]`. * Each node's value is between `[-10^4, 10^4]`.

Code Snippets

C++:

```
/**
* 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 {
public:
int goodNodes(TreeNode* root) {

}
};
```

Java:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* int val;
* TreeNode left;
* TreeNode right;
* TreeNode() {}
* TreeNode(int val) { this.val = val; }
* TreeNode(int val, TreeNode left, TreeNode right) {
* this.val = val;
* this.left = left;
```

```
* this.right = right;
* }
* }
*/
class Solution {
public int goodNodes(TreeNode root) {

}
}
```

Python3:

```
# Definition for a binary tree node.
# class TreeNode:
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution:
def goodNodes(self, root: TreeNode) -> int:
```