Problem 2792: Count Nodes That Are Great Enough

Problem Information

Difficulty: Hard
Acceptance Rate: 56.34%
Paid Only: Yes
Tags: Divide and Conquer, Tree, Depth-First Search, Binary Tree

Problem Description

You are given a `root` to a binary tree and an integer `k`. A node of this tree is called **great enough** if the followings hold:

* Its subtree has **at least** `k` nodes. * Its value is **greater** than the value of **at least** `k` nodes in its subtree.

Return _the number of nodes in this tree that are great enough._

The node `u` is in the **subtree** of the node `v`, if `u == v` or `v` is an ancestor of `u`.

Example 1:

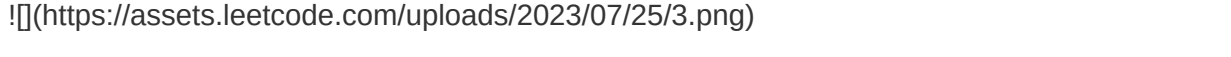
Input: root = [7,6,5,4,3,2,1], k = 2 **Output:** 3 **Explanation:** Number the nodes from 1 to 7. The values in the subtree of node 1: {1,2,3,4,5,6,7}. Since node.val == 7, there are 6 nodes having a smaller value than its value. So it's great enough. The values in the subtree of node 2: {3,4,6}. Since node.val == 6, there are 2 nodes having a smaller value than its value. So it's great enough. The values in the subtree of node 3: {1,2,5}. Since node.val == 5, there are 2 nodes having a smaller value than its value. So it's great enough. It can be shown that other nodes are not great enough. See the picture below for a better understanding.

Example 2:

Input: root = [1,2,3], k = 1 **Output:** 0 **Explanation:** Number the nodes from 1 to 3. The values in the subtree of node 1: {1,2,3}. Since node.val == 1, there are no nodes having a smaller value than its value. So it's not great enough. The values in the subtree of node 2: {2}. Since node.val == 2, there are no nodes having a smaller value than its value. So it's not great enough. The values in the subtree of node 3: {3}. Since node.val == 3, there are no nodes having a smaller value than its value. So it's not great enough. See the picture below for a better understanding.

Example 3:

Input: root = [3,2,2], k = 2 **Output:** 1 **Explanation:** Number the nodes from 1 to 3. The values in the subtree of node 1: {2,2,3}. Since node.val == 3, there are 2 nodes having a smaller value than its value. So it's great enough. The values in the subtree of node 2: {2}. Since node.val == 2, there are no nodes having a smaller value than its value. So it's not great enough. The values in the subtree of node 3: {2}. Since node.val == 2, there are no nodes having a smaller value than its value. So it's not great enough. See the picture below for a better understanding.



Constraints:

* The number of nodes in the tree is in the range `[1, 104]`. * `1 <= Node.val <= 104` * `1 <= k <= 10`

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 countGreatEnoughNodes(TreeNode* root, int k) {

}
};
```

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 countGreatEnoughNodes(TreeNode root, int k) {

}
}
```

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 countGreatEnoughNodes(self, root: Optional[TreeNode], k: int) -> int:
```