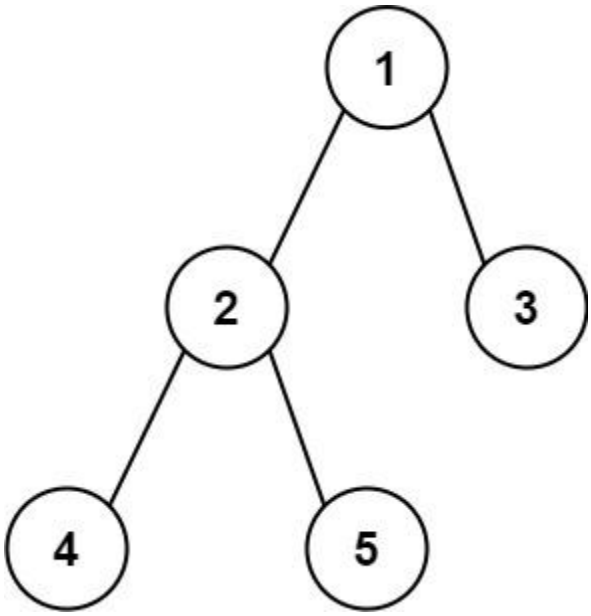
Given the root of a binary tree, return *the length of the diameter of the tree*.

The diameter of a binary tree is the length of the longest path between any two nodes in a tree. This path may or may not pass through the root.

The length of a path between two nodes is represented by the number of edges between them.

Example 1:



Input: root = [1,2,3,4,5]
Output: 3
Explanation: 3 is the length of the path [4,2,1,3] or [5,2,1,3].

Example 2:

Input: root = [1,2]
Output: 1

Constraints:

- **The number of nodes in the tree is in the range [1, 10^4].**
- **-100 <= Node.val <= 100**

Solution:

```
class Solution {
    public int diameterOfBinaryTree(TreeNode root) {

        // Create an array to hold the diameter of the tree
        int diameter[] = new int[1];

        // Recursively calculate the height of the tree and update the diameter array
        height(root,diameter);

        // Return the diameter of the tree
      return diameter[0];
    }

    public int height(TreeNode root, int diameter[]){

        // Base case: if the root is null, the height is 0
        if(root == null){
            return 0;
        }

        // Recursively calculate the height of the left and right subtrees
        int left = height(root.left,diameter);
        int right = height(root.right,diameter);

        // Update the diameter array by taking the maximum diameter that passes through the
current node
        diameter[0] = Math.max(diameter[0],left + right);

        // Return the maximum depth of the current node by adding 1 to the maximum depth of
its deepest subtree
        return Math.max(left,right)+1;
    }
}
```