1457. Pseudo-Palindromic Paths in a Binary Tree

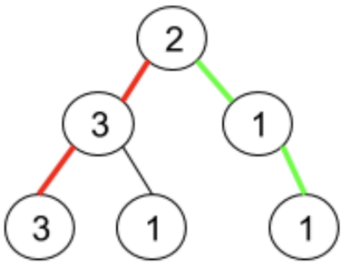
Solved

Medium · Topics · Companies · Hint

Given a binary tree where node values are digits from 1 to 9. A path in the binary tree is said to be **pseudo-palindromic** if at least one permutation of the node values in the path is a palindrome.

*Return the number of **pseudo-palindromic** paths going from the root node to leaf nodes.*

Example 1:



```
Input: root = [2,3,1,3,1,null,1]
Output: 2
Explanation: The figure above represents the given binary tree. There are three paths going from the
root node to leaf nodes: the red path [2,3,3], the green path [2,1,1], and the path [2,3,1]. Among
these paths only red path and green path are pseudo-palindromic paths since the red path [2,3,3] can
```

2.7K · 37

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     public var val: Int
 *     public var left: TreeNode?
 *     public var right: TreeNode?
 *     public init() { self.val = 0; self.left = nil; self.right = nil; }
 *     public init(_ val: Int) { self.val = val; self.left = nil; self.right = nil; }
 *     public init(_ val: Int, _ left: TreeNode?, _ right: TreeNode?) {
 *         self.val = val
 *         self.left = left
 *         self.right = right
 *     }
 * }
 */
class Solution {
```

```swift
    func pseudoPalindromicPaths (_ root: TreeNode?) -> Int {
        guard let root = root else {
            return 0
        }
        func dfs(_ root: TreeNode, _ path:[Int:Int]) -> Int {
            if(root.left == nil && root.right == nil) {
                var p = path
                p[root.val,default:0]+=1

                var countOdd = 0
                for (i,v) in p {
                    if(v % 2 != 0){
                        countOdd = countOdd + 1
                    }
                }
                return countOdd > 1 ? 0 : 1
            }

            var p = path
            p[root.val,default:0]+=1
            var l = 0, r = 0
            if let left = root.left {
                l = dfs(left,p)
            }

            if let right = root.right {
                r = dfs(right,p)
            }

            return l + r
        }

        return dfs(root,[:])
    }
}
```