AlgoExpert

Quad Layout | Swift | 12px | Sublime | Monok

Prompt | Scratchpad | Our Solution(s) | Video Explanation | Run Code

Solution 1 | Solution 2 | Solution 3 | Solution 4

```swift
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

class Program {
  class BinaryTree: Hashable {
    var value: Int
    var left: BinaryTree?
    var right: BinaryTree?

    init(value: Int) {
      self.value = value
    }

    static func == (lhs: BinaryTree, rhs: BinaryTree) -> Bool {
      return ObjectIdentifier(lhs) == ObjectIdentifier(rhs)
    }

    var hashValue: Int {
      return ObjectIdentifier(self).hashValue
    }
  }

  // Average case: when the tree is balanced
  // O(n) time | O(n) space - where n is the number of nodes in the Binary Tree
  static func allKindsOfNodeDepths(_ root: BinaryTree?) -> Int {
    var nodeCounts = [BinaryTree: Int]()
    var nodeDepths = [BinaryTree: Int]()
    addNodeCounts(root, &nodeCounts)
    addNodeDepths(root, &nodeDepths, &nodeCounts)
    return sumAllNodeDepths(root, nodeDepths)
  }

  static func sumAllNodeDepths(_ node: BinaryTree?, _ nodeDepths: [BinaryTree: Int]) -> Int {
    if let n = node {
      return sumAllNodeDepths(n.left, nodeDepths) + sumAllNodeDepths(n.right, nodeDepths) + nodeDepths[n]!
    }
    return 0
  }

  static func addNodeDepths(_ tree: BinaryTree?, _ nodeDepths: inout [BinaryTree: Int], _ nodeCounts: inout [BinaryTree: Int]) {
    if let node = tree {
      nodeDepths[node] = 0

      if let left = node.left {
        addNodeDepths(left, &nodeDepths, &nodeCounts)
        nodeDepths[node] = nodeDepths[node]! + nodeDepths[left]! + nodeCounts[left]!
      }

      if let right = node.right {
        addNodeDepths(right, &nodeDepths, &nodeCounts)
        nodeDepths[node] = nodeDepths[node]! + nodeDepths[right]! + nodeCounts[right]!
      }
    }
  }

  static func addNodeCounts(_ tree: BinaryTree?, _ nodeCounts: inout [BinaryTree: Int]) {
    if let node = tree {
      nodeCounts[node] = 1

      if let left = node.left {
        addNodeCounts(left, &nodeCounts)
        nodeCounts[node] = nodeCounts[node]! + nodeCounts[left]!
      }

      if let right = node.right {
        addNodeCounts(right, &nodeCounts)
        nodeCounts[node] = nodeCounts[node]! + nodeCounts[right]!
      }
    }
  }
}
```