AlgoExpert

Quad Layout | C# | 12px | Sublime | Monok

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

Solution 1 | Solution 2 | Solution 3 | Solution 4

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

using System.Collections.Generic;

public class Program {
  // Average case: when the tree is balanced
  // O(n) time | O(n) space - where n is the number of nodes in the Binary Tree
  public static int AllKindsOfNodeDepths(BinaryTree root) {
    Dictionary<BinaryTree, int> nodeCounts = new Dictionary<BinaryTree, int>();
    Dictionary<BinaryTree, int> nodeDepths = new Dictionary<BinaryTree, int>();
    addNodeCounts(root, nodeCounts);
    addNodeDepths(root, nodeDepths, nodeCounts);
    return sumAllNodeDepths(root, nodeDepths);
  }

  public static int sumAllNodeDepths(BinaryTree node,
    Dictionary<BinaryTree, int> nodeDepths) {
    if (node == null) return 0;
    return sumAllNodeDepths(node.left, nodeDepths) + sumAllNodeDepths(node.right,
             nodeDepths) +
           nodeDepths[node];
  }

  public static void addNodeDepths(BinaryTree node, Dictionary<BinaryTree, int> nodeDepths,
    Dictionary<BinaryTree, int> nodeCounts) {
    nodeDepths[node] = 0;
    if (node.left != null) {
      addNodeDepths(node.left, nodeDepths, nodeCounts);
      nodeDepths[node] = nodeDepths[node] + nodeDepths[node.left] +
        nodeCounts[node.left];
    }
    if (node.right != null) {
      addNodeDepths(node.right, nodeDepths, nodeCounts);
      nodeDepths[node] = nodeDepths[node] + nodeDepths[node.right] +
        nodeCounts[node.right];
    }
  }

  public static void addNodeCounts(BinaryTree node, Dictionary<BinaryTree, int> nodeCounts) {
    nodeCounts[node] = 1;
    if (node.left != null) {
      addNodeCounts(node.left, nodeCounts);
      nodeCounts[node] = nodeCounts[node] + nodeCounts[node.left];
    }
    if (node.right != null) {
      addNodeCounts(node.right, nodeCounts);
      nodeCounts[node] = nodeCounts[node] + nodeCounts[node.right];
    }
  }

  public class BinaryTree {
    public int value;
    public BinaryTree left;
    public BinaryTree right;

    public BinaryTree(int value) {
      this.value = value;
      left = null;
      right = null;
    }
  }
}
```