```
1  // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3  using System.Collections.Generic;
4
5  public class Program {
6    // Average case: when the tree is balanced
7    // O(nlog(n)) time | O(h) space - where n is the number of nodes in
8    // the Binary Tree and h is the height of the Binary Tree
9    public static int AllKindsOfNodeDepths(BinaryTree root) {
10     int sumOfAllDepths = 0;
11     Stack<BinaryTree> stack = new Stack<BinaryTree>();
12     stack.Push(root);
13     while (stack.Count > 0) {
14       BinaryTree node = stack.Pop();
15       if (node == null) continue;
16
17       sumOfAllDepths += nodeDepths(node, 0);
18       stack.Push(node.left);
19       stack.Push(node.right);
20     }
21     return sumOfAllDepths;
22   }
23
24   public static int nodeDepths(BinaryTree node, int depth) {
25     if (node == null) return 0;
26     return depth + nodeDepths(node.left, depth + 1) + nodeDepths(node.right, depth + 1);
27   }
28
29   public class BinaryTree {
30     public int value;
31     public BinaryTree left;
32     public BinaryTree right;
33
34     public BinaryTree(int value) {
35       this.value = value;
36       left = null;
37       right = null;
38     }
39   }
40 }
41
```