

Solution 1	Solution 2	Solution 3	Solution 4
<pre>1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved. 2 3 ▼ import java.util.*; 4 5 ▼ class Program { 6     // Average case: when the tree is balanced 7     // O(n) time   O(n) space - where n is the number of nodes in the Binary Tree 8 ▼ public static int allKindsOfNodeDepths(BinaryTree root) { 9     Map&lt;BinaryTree, Integer&gt; nodeCounts = new HashMap&lt;BinaryTree, Integer&gt;(); 10    Map&lt;BinaryTree, Integer&gt; nodeDepths = new HashMap&lt;BinaryTree, Integer&gt;(); 11    addNodeCounts(root, nodeCounts); 12    addNodeDepths(root, nodeDepths, nodeCounts); 13    return sumAllNodeDepths(root, nodeDepths); 14 } 15 16 ▼ public static int sumAllNodeDepths(BinaryTree node, Map&lt;BinaryTree, Integer&gt; nodeDepths) { 17     if (node == null) return 0; 18     return sumAllNodeDepths(node.left, nodeDepths) + sumAllNodeDepths(node.right, nodeDepths) + nodeDepths.get(node); 19 } 20 21 ▼ public static void addNodeDepths(BinaryTree node, Map&lt;BinaryTree, Integer&gt; nodeDepths, Map&lt;BinaryTree, Integer&gt; nodeCounts) { 22     nodeDepths.put(node, 0); 23 ▼     if (node.left != null) { 24         addNodeDepths(node.left, nodeDepths, nodeCounts); 25         nodeDepths.put(node, nodeDepths.get(node) + nodeDepths.get(node.left) + nodeCounts.get(node.left)); 26     } 27 ▼     if (node.right != null) { 28         addNodeDepths(node.right, nodeDepths, nodeCounts); 29         nodeDepths.put(node, nodeDepths.get(node) + nodeDepths.get(node.right) + nodeCounts.get(node.right)); 30     } 31 } 32 33 ▼ public static void addNodeCounts(BinaryTree node, Map&lt;BinaryTree, Integer&gt; nodeCounts) { 34     nodeCounts.put(node, 1); 35 ▼     if (node.left != null) { 36         addNodeCounts(node.left, nodeCounts); 37         nodeCounts.put(node, nodeCounts.get(node) + nodeCounts.get(node.left)); 38     } 39 ▼     if (node.right != null) { 40         addNodeCounts(node.right, nodeCounts); 41         nodeCounts.put(node, nodeCounts.get(node) + nodeCounts.get(node.right)); 42     } 43 } 44 45 ▼ static class BinaryTree { 46     int value; 47     BinaryTree left; 48     BinaryTree right; 49 50 ▼     public BinaryTree(int value) { 51         this.value = value; 52         left = null; 53         right = null; 54     } 55 } 56 } 57</pre>			

