

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

Solution 1	Solution 2	Solution 3	Solution 4
<pre>1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved. 2 3 using namespace std; 4 5 ▼ class BinaryTree { 6     public: 7         int value; 8         BinaryTree *left; 9         BinaryTree *right; 10 11 ▼ BinaryTree(int value) { 12     this-&gt;value = value; 13     left = NULL; 14     right = NULL; 15 } 16 }; 17 18 int nodeDepths(BinaryTree *node, int depth = 0); 19 20 // Average case: when the tree is balanced 21 // O(nlog(n)) time   O(h) space - where n is the number of nodes in 22 // the Binary Tree and h is the height of the Binary Tree 23 ▼ int allKindsOfNodeDepths(BinaryTree *root) { 24     int sumOfAllDepths = 0; 25     vector&lt;BinaryTree *&gt; stack = {root}; 26 ▼ while (stack.size() &gt; 0) { 27     BinaryTree *node = stack.back(); 28     stack.pop_back(); 29     if (node == NULL) 30         continue; 31     sumOfAllDepths += nodeDepths(node); 32     stack.push_back(node-&gt;left); 33     stack.push_back(node-&gt;right); 34 } 35 return sumOfAllDepths; 36 } 37 38 ▼ int nodeDepths(BinaryTree *node, int depth) { 39     if (node == NULL) 40         return 0; 41     return depth + nodeDepths(node-&gt;left, depth + 1) + 42           nodeDepths(node-&gt;right, depth + 1); 43 }</pre>			

