

Solution 1Solution 2Solution 3Solution 4

```
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->value = value;
13         left = NULL;
14         right = NULL;
15     }
16 };
17
18 int sumAllNodeDepths(BinaryTree *node,
19                     unordered_map<BinaryTree *, int> &nodeDepths);
20 void addNodeDepths(BinaryTree *node,
21                   unordered_map<BinaryTree *, int> &nodeDepths,
22                   unordered_map<BinaryTree *, int> &nodeCounts);
23 void addNodeCounts(BinaryTree *node,
24                   unordered_map<BinaryTree *, int> &nodeCounts);
25
26 // Average case: when the tree is balanced
27 // O(n) time | O(n) space - where n is the number of nodes in the Binary Tree
28 ▼ int allKindsOfNodeDepths(BinaryTree *root) {
29     unordered_map<BinaryTree *, int> nodeCounts = {};
30     addNodeCounts(root, nodeCounts);
31     unordered_map<BinaryTree *, int> nodeDepths = {};
32     addNodeDepths(root, nodeDepths, nodeCounts);
33     return sumAllNodeDepths(root, nodeDepths);
34 }
35
36 int sumAllNodeDepths(BinaryTree *node,
37 ▼                     unordered_map<BinaryTree *, int> &nodeDepths) {
38     if (node == NULL)
39         return 0;
40     return sumAllNodeDepths(node->left, nodeDepths) +
41            sumAllNodeDepths(node->right, nodeDepths) + nodeDepths[node];
42 }
43
44 void addNodeDepths(BinaryTree *node,
45                   unordered_map<BinaryTree *, int> &nodeDepths,
46 ▼                   unordered_map<BinaryTree *, int> &nodeCounts) {
47     nodeDepths.insert({node, 0});
48 ▼     if (node->left != NULL) {
49         addNodeDepths(node->left, nodeDepths, nodeCounts);
50         nodeDepths[node] += nodeDepths[node->left] + nodeCounts[node->left];
51     }
52 ▼     if (node->right != NULL) {
53         addNodeDepths(node->right, nodeDepths, nodeCounts);
54         nodeDepths[node] += nodeDepths[node->right] + nodeCounts[node->right];
55     }
56 }
57
58 void addNodeCounts(BinaryTree *node,
59 ▼                   unordered_map<BinaryTree *, int> &nodeCounts) {
60     nodeCounts.insert({node, 1});
61 ▼     if (node->left != NULL) {
62         addNodeCounts(node->left, nodeCounts);
63         nodeCounts[node] += nodeCounts[node->left];
64     }
65 ▼     if (node->right != NULL) {
66         addNodeCounts(node->right, nodeCounts);
67         nodeCounts[node] += nodeCounts[node->right];
68     }
69 }
```

