```python
# Copyright © 2020 AlgoExpert, LLC. All rights reserved.

# Average case: when the tree is balanced
# O(n) time | O(n) space - where n is the number of nodes in the Binary Tree
def allKindsOfNodeDepths(root):
    nodeCounts = {}
    addNodeCounts(root, nodeCounts)
    nodeDepths = {}
    addNodeDepths(root, nodeDepths, nodeCounts)
    return sumAllNodeDepths(root, nodeDepths)


def sumAllNodeDepths(node, nodeDepths):
    if node is None:
        return 0
    return sumAllNodeDepths(node.left, nodeDepths) + sumAllNodeDepths(node.right, nodeDepths) + nodeDepths[node]


def addNodeDepths(node, nodeDepths, nodeCounts):
    nodeDepths[node] = 0
    if node.left is not None:
        addNodeDepths(node.left, nodeDepths, nodeCounts)
        nodeDepths[node] += nodeDepths[node.left] + nodeCounts[node.left]
    if node.right is not None:
        addNodeDepths(node.right, nodeDepths, nodeCounts)
        nodeDepths[node] += nodeDepths[node.right] + nodeCounts[node.right]


def addNodeCounts(node, nodeCounts):
    nodeCounts[node] = 1
    if node.left is not None:
        addNodeCounts(node.left, nodeCounts)
        nodeCounts[node] += nodeCounts[node.left]
    if node.right is not None:
        addNodeCounts(node.right, nodeCounts)
        nodeCounts[node] += nodeCounts[node.right]


# This is the class of the input binary tree.
class BinaryTree:
    def __init__(self, value):
        self.value = value
        self.left = None
        self.right = None

```