

Our Solution(s)

Run Code

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 class Program {
4     class BinaryTree {
5         var value: Int?
6         var left: BinaryTree?
7         var right: BinaryTree?
8
9         init(value: Int) {
10             self.value = value
11             left = nil
12             right = nil
13         }
14     }
15
16     // O(n) time | O(log(n)) space
17     func maxPathSum(tree: BinaryTree?) -> Int {
18         let rootMaxSumTuple = findMaxSum(tree: tree)
19         return rootMaxSumTuple.1
20     }
21
22     func findMaxSum(tree: BinaryTree?) -> (Int, Int) {
23         if tree === nil {
24             return (0, 0)
25         }
26
27         let leftMaxSumTuple = findMaxSum(tree: tree?.left)
28         let rightMaxSumTuple = findMaxSum(tree: tree?.right)
29         let childStraightMaxSum = max(leftMaxSumTuple.0, rightMaxSumTu
30
31         let value = tree!.value!
32
33         let currentStraightMaxSum = max(value + childStraightMaxSum, v
```

Your Solutions

Run Code

Solution 1 Solution 2 Solution 3

```
1 class Program {
2     // This is an input class. Do not edit.
3     class BinaryTree {
4         var value: Int?
5         var left: BinaryTree?
6         var right: BinaryTree?
7
8         init(value: Int) {
9             self.value = value
10            left = nil
11            right = nil
12        }
13    }
14
15    func maxPathSum(tree: BinaryTree?) -> Int {
16        // Write your code here.
17        return -1
18    }
19 }
20
```

Our Tests

```
1 class ProgramTests {
2     Test Suite 1
3     test maxPathSum() {
4         let tree = BinaryTree(value: 1)
5         tree?.left = BinaryTree(value: 2)
6         tree?.right = BinaryTree(value: 3)
7         tree?.left?.left = BinaryTree(value: 4)
8         tree?.left?.right = BinaryTree(value: 5)
9         tree?.right?.left = BinaryTree(value: 6)
10        tree?.right?.right = BinaryTree(value: 7)
11    }
12 }
```

Custom Output

Submit Code

```

10 def test_max_depth_0():
11     root = Node(1)
12     assert max_depth(root) == 0
13
14 def test_max_depth_1():
15     root = Node(1)
16     root.left = Node(2)
17     assert max_depth(root) == 1
18
19 def test_max_depth_2():
20     root = Node(1)
21     root.left = Node(2)
22     root.left.right = Node(3)
23     assert max_depth(root) == 2
24
25 def test_max_depth_3():
26     root = Node(1)
27     root.left = Node(2)
28     root.left.right = Node(3)
29     root.left.right.right = Node(4)
30     assert max_depth(root) == 3

```

Run or submit code when you're ready.