Solution 1

```swift
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

class Program {
    class BinaryTree {
        var value: Int?
        var left: BinaryTree?
        var right: BinaryTree?

        init(value: Int) {
            self.value = value
            left = nil
            right = nil
        }
    }

    // O(n) time | O(log(n)) space
    func maxPathSum(tree: BinaryTree?) -> Int {
        let rootMaxSumTuple = findMaxSum(tree: tree)
        return rootMaxSumTuple.1
    }

    func findMaxSum(tree: BinaryTree?) -> (Int, Int) {
        if tree === nil {
            return (0, 0)
        }

        let leftMaxSumTuple = findMaxSum(tree: tree?.left)
        let rightMaxSumTuple = findMaxSum(tree: tree?.right)
        let childStraightMaxSum = max(leftMaxSumTuple.0, rightMaxSumTuple.0)

        let value = tree!.value!

        let currentStraightMaxSum = max(value + childStraightMaxSum, value)
        let currentTriangleMaxSum = max(leftMaxSumTuple.0 + value + rightMaxSumTuple.0, currentStraightMaxSum)
        let currentMaxSum = max(max(leftMaxSumTuple.1, rightMaxSumTuple.1), currentTriangleMaxSum)

        return (currentStraightMaxSum, currentMaxSum)
    }
}

```