Our Solution(s) Run Code

Solution 1

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

package main

type BinaryTree struct {
  Value       int
  Left, Right *BinaryTree
}

// O(n) time | O(log(n)) space
func MaxPathSum(tree *BinaryTree) int {
  _, maxSum := findMaxSum(tree)
  return maxSum
}

func findMaxSum(tree *BinaryTree) (int, int) {
  if tree == nil {
    return 0, 0
  }
  leftMaxSumAsBranch, leftMaxPathSum := findMaxSum(tree.Left)
  rightMaxSumAsBranch, rightMaxPathSum := findMaxSum(tree.Right)
  maxChildSumAsBranch := max(leftMaxSumAsBranch, rightMaxSumAsBranch)

  value := tree.Value
  maxSumAsBranch := max(maxChildSumAsBranch+value, value)
  maxSumAsRootNode := max(leftMaxSumAsBranch+value+rightMaxSumAsBranch
  maxPathSum := max(leftMaxPathSum, rightMaxPathSum, maxSumAsRootNode)

  return maxSumAsBranch, maxPathSum
}

func max(first int, vals ...int) int {
  for _, val := range vals {
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```go
package main

type BinaryTree struct {
  Value       int
  Left, Right *BinaryTree
}

func MaxPathSum(tree *BinaryTree) int {
  // Write your code here.
  return -1
}

```

Our Tests

Custom Output Submit Code

Run or submit code when you're ready.