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

package main

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

type Level struct {
  Root  *BinaryTree
  Depth int
}

// Average case: when the tree is balanced
// O(n) time | O(h) space - where n is the number of nodes in
// the Binary Tree and h is the height of the Binary Tree
func NodeDepths(root *BinaryTree) int {
  sumOfDepths := 0
  stack := []Level{{Root: root, Depth: 0}}
  var top Level
  for len(stack) > 0 {
    top, stack = stack[len(stack)-1], stack[:len(stack)-1]
    node, depth := top.Root, top.Depth
    if node == nil {
      continue
    }
    sumOfDepths += depth
    stack = append(stack, Level{Root: node.Left, Depth: depth + 1})
    stack = append(stack, Level{Root: node.Right, Depth: depth + 1})
  }
  return sumOfDepths
}

```