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

class Program {
    // O(n^2) time | O(1) space
    func numberOfBinaryTreeTopologies(_ n: Int) -> Int {
        var cache = [0: 1]
        return numberOfBinaryTreeTopologiesHelper(n, &cache)
    }

    func numberOfBinaryTreeTopologiesHelper(_ n: Int, _ cache: inout [Int: Int]) -> Int {
        if let cachedValue = cache[n] {
            return cachedValue
        }

        var numberOfTopologies = 0

        for leftTreeSize in 0 ..< n {
            let rightTreeSize = n - 1 - leftTreeSize

            let leftNumberOfTopologies = numberOfBinaryTreeTopologiesHelper(leftTreeSize, &cache)
            let rightNumberOfTopologies = numberOfBinaryTreeTopologiesHelper(rightTreeSize, &cache)
            numberOfTopologies += leftNumberOfTopologies * rightNumberOfTopologies
        }

        cache[n] = numberOfTopologies
        return numberOfTopologies
    }
}

```