Our Solution(s) Run Code

Solution 1 Solution 2 Solution 3

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

package main

// O(n^2) time | O(n) space
func NumberOfBinaryTreeTopologies(n int) int {
  return helper(n, map[int]int{0: 1})
}

func helper(n int, cache map[int]int) int {
  if val, found := cache[n]; found {
    return val
  }
  numberOfTrees := 0
  for leftTreeSize := 0; leftTreeSize < n; leftTreeSize++ {
    rightTreeSize := n - 1 - leftTreeSize
    numberOfLeftTrees := helper(leftTreeSize, cache)
    numberOfRightTrees := helper(rightTreeSize, cache)
    numberOfTrees += numberOfLeftTrees * numberOfRightTrees
  }
  cache[n] = numberOfTrees
  return numberOfTrees
}

```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```go
package main

func NumberOfBinaryTreeTopologies(n int) int {
  // Write your code here.
  return -1
}

```

Our Tests

Custom Output Submit Code

```
func (s *TestSuite) TestCase23(t *testing.T) {
    expected := 3
    output := NumberOfBinaryTreeTopologies(23)
    require.Equal(t, expected, output)
}

func (s *TestSuite) TestCase24(t *testing.T) {
    expected := 3
    output := NumberOfBinaryTreeTopologies(24)
    require.Equal(t, expected, output)
}

func (s *TestSuite) TestCase25(t *testing.T) {
    expected := 3
```

Run or submit code when you're ready.