Our Solution(s)   Run Code

Solution 1   Solution 2   Solution 3

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

using System.Collections.Generic;

public class Program {
  // O(n^2) time | O(n) space
  public static int NumberOfBinaryTreeTopologies(int n) {
    Dictionary<int, int> cache = new Dictionary<int, int>();
    cache.Add(0, 1);
    return NumberOfBinaryTreeTopologies(n, cache);
  }

  public static int NumberOfBinaryTreeTopologies(int n, Dictionary<in
    if (cache.ContainsKey(n)) {
      return cache[n];
    }
    int numberOfTrees = 0;
    for (int leftTreeSize = 0; leftTreeSize < n; leftTreeSize++) {
      int rightTreeSize = n - 1 - leftTreeSize;
      int numberOfLeftTrees = NumberOfBinaryTreeTopologies(leftTreeSiz
      int numberOfRightTrees = NumberOfBinaryTreeTopologies(rightTreeS
      numberOfTrees += numberOfLeftTrees * numberOfRightTrees;
    }
    cache.Add(n, numberOfTrees);
    return numberOfTrees;
  }
}
```

Your Solutions   Run Code

Solution 1   Solution 2   Solution 3

```csharp
public class Program {
  public static int NumberOfBinaryTreeTopologies(int n) {
    // Write your code here.
    return -1;
  }
}
```

Our Tests

Custom Output   Submit Code

```
    public void TestCase2() {
        Utils.AssertTrue(Program.NumberOfHistoryTreeTopologies(3) == 3);
    }

    [Test]
    public void TestCase3() {
        Utils.AssertTrue(Program.NumberOfHistoryTreeTopologies(4) == 5);
    }

    [Test]
    public void TestCase4() {
        Utils.AssertTrue(Program.NumberOfHistoryTreeTopologies(5) == 8);
    }

    [Test]
```

Run or submit code when you're ready.