

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1Solution 2Solution 3

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 using namespace std;
4
5 // O(n^2) time | O(n) space
6 int numberOfBinaryTreeTopologies(int n) {
7     vector<int> cache{1};
8     for (int m = 1; m < n + 1; m++) {
9         int numberOfTrees = 0;
10        for (int leftTreeSize = 0; leftTreeSize < m; leftTreeSize++) {
11            int rightTreeSize = m - 1 - leftTreeSize;
12            int numberOfLeftTrees = cache[leftTreeSize];
13            int numberOfRightTrees = cache[rightTreeSize];
14            numberOfTrees += numberOfLeftTrees * numberOfRightTrees;
15        }
16        cache.push_back(numberOfTrees);
17    }
18    return cache[n];
19 }
20
```

Solution 1Solution 2Solution 3

```
1 using namespace std;
2
3 int numberOfBinaryTreeTopologies(int n) {
4     // Write your code here.
5     return -1;
6 }
7
```

Our Tests

```
1 // Test 1: n = 1, expected result = 1
2 // Test 2: n = 2, expected result = 2
3 // Test 3: n = 3, expected result = 5
4 // Test 4: n = 4, expected result = 14
5 // Test 5: n = 5, expected result = 42
6 // Test 6: n = 6, expected result = 112
7 // Test 7: n = 7, expected result = 322
8 // Test 8: n = 8, expected result = 843
9 // Test 9: n = 9, expected result = 2147
10 // Test 10: n = 10, expected result = 5528
```

Custom Output

Submit Code

```
1 // Custom Output
2
```

