

Solution Review: Problem Challenge 3

We'll cover the following

- Count of Structurally Unique Binary Search Trees (hard)
- Solution
- Code
 - Time complexity
 - Space complexity
- Memoized version

Count of Structurally Unique Binary Search Trees (hard)

Given a number 'n', write a function to return the count of structurally unique Binary Search Trees (BST) that can store values 1 to 'n'.

Example 1:

```
Input: 2
Output: 2
Explanation: As we saw in the previous problem, there are 2 unique BSTs storing nu
mbers from 1-2.
```

Example 2:

```
Input: 3
Output: 5
Explanation: There will be 5 unique BSTs that can store numbers from 1 to 5.
```

Solution

This problem is similar to Structurally Unique Binary Search Trees (https://www.educative.io/collection/page/5668639101419520/5671464854355968/567997479518 2080/). Following a similar approach, we can iterate from 1 to 'n' and consider each number as the root of a tree and make two recursive calls to count the number of left and right sub-trees.

Code

Here is what our algorithm will look like:

```
Java

Python3

C++

Js JS

1

2 class TreeNode:

3 def __init__(self, val):

4 self.val = val

C--

educative.left = None

6 self.right = None
```

```
7
8
9 def count_trees(n):
10
    if n <= 1:
11
        return 1
12
      count = 0
      for i in range(1, n+1):
13
        # making 'i' root of the tree
14
        countOfLeftSubtrees = count_trees(i - 1)
15
16
        countOfRightSubtrees = count_trees(n - i)
17
        count += (countOfLeftSubtrees * countOfRightSubtrees)
18
19
      return count
20
21
22 def main():
      print("Total trees: " + str(count trees(2)))
23
      print("Total trees: " + str(count_trees(3)))
24
25
26
27
   main()
28
                                                                                        \leftarrow
D
```

Time complexity

The time complexity of this algorithm will be exponential and will be similar to Balanced Parentheses

(https://www.educative.io/collection/page/5668639101419520/5671464854355968/575326411712 1024/). Estimated time complexity will be $O(n*2^n)$ but the actual time complexity ($O(4^n/\sqrt{n})$) is bounded by the Catalan number

(https://en.wikipedia.org/wiki/Catalan_number) and is beyond the scope of a coding interview. See more details here (https://en.wikipedia.org/wiki/Central_binomial_coefficient).

Space complexity

The space complexity of this algorithm will be exponential too, estimated $O(2^n)$ but the actual will be ($O(4^n/\sqrt{n})$.

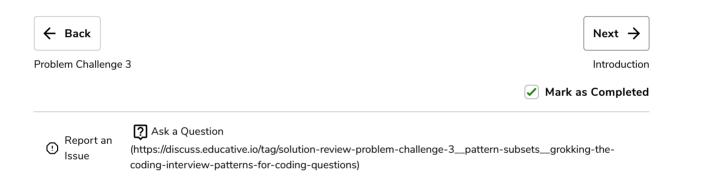
Memoized version

Our algorithm has overlapping subproblems as our recursive call will be evaluating the same sub-expression multiple times. To resolve this, we can use memoization and store the intermediate results in a **HashMap**. In each function call, we can check our map to see if we have already evaluated this sub-expression before. Here is the memoized version of our algorithm, please see highlighted changes:

```
Python3
                             ⊘ C++
                                          Js JS
  🕌 Java
    8
       def count_trees(n):
    9
         return count_trees_rec({}, n)
   10
   11
   12 def count_trees_rec(map, n):
   13
         if n in map:
   14
           return map[n]
   15
jagducative<sub>rn 1</sub>
          count - 0
```

```
±υ
19
      for i in range(1, n+1):
20
        # making 'i' the root of the tree
21
        countOfLeftSubtrees = count_trees_rec(map, i - 1)
22
        countOfRightSubtrees = count_trees_rec(map, n - i)
        count += (countOfLeftSubtrees * countOfRightSubtrees)
23
24
25
      map[n] = count
26
      return count
27
28
29
    def main():
30
      print("Total trees: " + str(count_trees(2)))
      print("Total trees: " + str(count_trees(3)))
31
32
33
34
    main()
35
\triangleright
                                                                                   []
```

The time complexity of the memoized algorithm will be $O(n^2)$, since we are iterating from '1' to 'n' and ensuring that each sub-problem is evaluated only once. The space complexity will be O(n) for the memoization map.



Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.