95. Unique Binary Search Trees II

Oct. 30, 2018 | 38.8K views

Average Rating: 4.50 (30 votes)

Given an integer `n`, generate all structurally unique **BST's** (binary search trees) that store values 1 ... *n*.

Example:

```
Input: 3
Output:
[
  [1,null,3,2],
  [3,2,null,1],
  [3,1,null,null,2],
  [2,1,3],
  [1,null,2,null,3]
]
Explanation:
The above output corresponds to the 5 unique BST's shown below:

   1         3     3      2      1
    \       /     /      / \      \
     3     2     1      1   3      2
    /     /       \                 \
   2     1         2                 3
```

Constraints:

- `0 <= n <= 8`

Solution

Tree definition

First of all, here is the definition of the `TreeNode` which we would use.

Java Python Copy

```python
# Definition for a binary tree node.
class TreeNode:
    def __init__(self, x):
        self.val = x
        self.left = None
        self.right = None
```

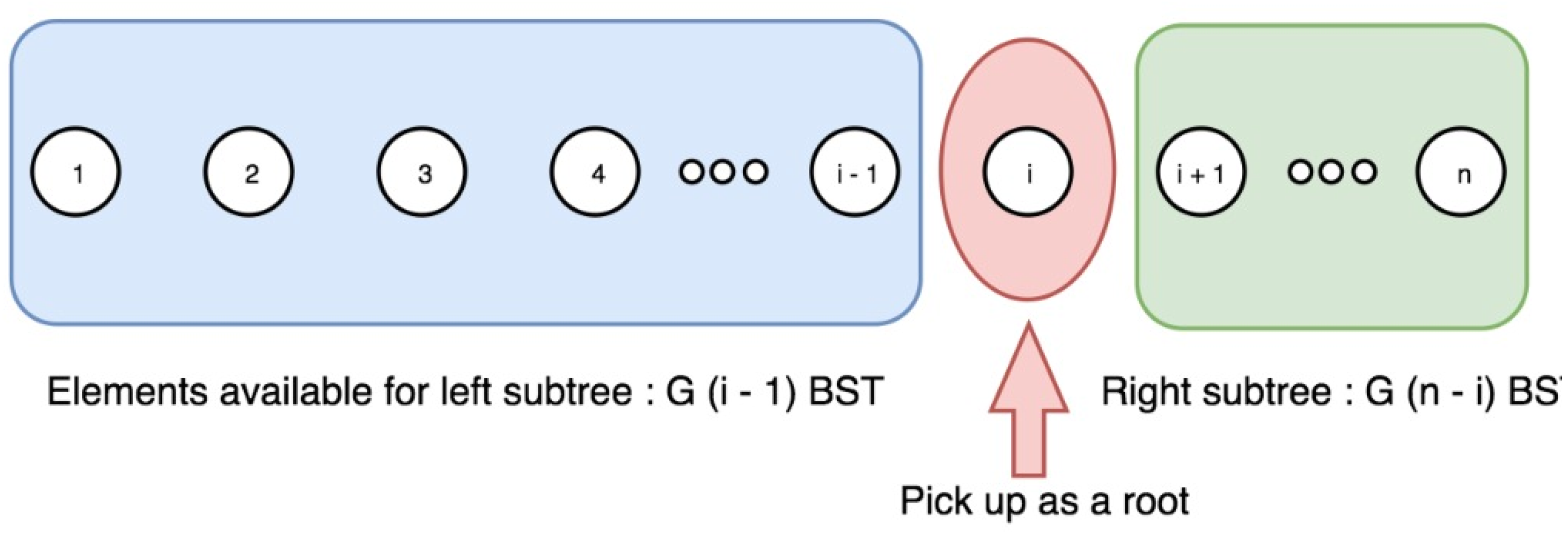
Approach 1: Recursion

First of all let's count how many trees do we have to construct. As you could check in this article, the number of possible BST is actually a Catalan number.

Let's follow the logic from the above article, this time not to count but to actually construct the trees.

Algorithm

Let's pick up number `i` out of the sequence `1 ..n` and use it as the root of the current tree. Then there are `i - 1` elements available for the construction of the left subtree and `n - i` elements available for the right subtree. As we already discussed that results in `G(i - 1)` different left subtrees and `G(n - i)` different right subtrees, where `G` is a Catalan number.



Now let's repeat the step above for the sequence `1 ... i - 1` to construct all left subtrees, and then for the sequence `i + 1 ... n` to construct all right subtrees.

This way we have a root `i` and two lists for the possible left and right subtrees. The final step is to loop over both lists to link left and right subtrees to the root.

Java Python Copy

```python
class Solution:
    def generateTrees(self, n):
        """
        :type n: int
        :rtype: List[TreeNode]
        """
        def generate_trees(start, end):
            if start > end:
                return [None,]

            all_trees = []
            for i in range(start, end + 1):  # pick up a root
                # all possible left subtrees if i is choosen to be a root
                left_trees = generate_trees(start, i - 1)

                # all possible right subtrees if i is choosen to be a root
                right_trees = generate_trees(i + 1, end)

                # connect left and right subtrees to the root i
                for l in left_trees:
                    for r in right_trees:
                        current_tree = TreeNode(i)
                        current_tree.left = l
                        current_tree.right = r
                        all_trees.append(current_tree)

            return all_trees
```

Complexity analysis

- Time complexity : The main computations are to construct all possible trees with a given root, that is actually Catalan number G_n as was discussed above. This is done `n` times, that results in time complexity nG_n. Catalan numbers grow as $\frac{4^n}{n^{3/2}}$ that gives the final complexity $\mathcal{O}(\frac{4^n}{n^{1/2}})$. Seems to be large but let's not forget that here we're asked to generate $G_n \sim \frac{4^n}{n^{3/2}}$ tree objects as output.
- Space complexity : nG_n as we keep G_n trees with `n` elements each, that results in $\mathcal{O}(\frac{4^n}{n^{1/2}})$ complexity.

Rate this article:

Previous Next

Comments: 11 Sort By

Type comment here... (Markdown is supported)

Preview Post

bjwu ★215 November 9, 2018 2:23 AM
One thing that can be improved is to introduce memorization. :)
I really enjoy Q 95 & 96
18 Share Reply
SHOW 6 REPLIES

LearningMind ★205 January 12, 2019 1:49 AM
So you mean if someone does not know Catalan number, can not compute time and space complexity?
Do you mind to explain the complexity without so called "Catalan number"?
14 Share Reply
SHOW 3 REPLIES

rostoe ★16 April 20, 2020 1:33 AM Report
How come this problem is medium, and merge k linked lists is hard?
7 Share Reply

kiljaeden ★3 July 4, 2020 6:50 PM
how come does not this require deepcopy?
2 Share Reply

finback ★12 June 5, 2020 8:14 PM
No need for a deep copy for these subTrees?
2 Share Reply
SHOW 1 REPLY

pygirl5 ★11 March 30, 2020 8:58 AM
Could you please explain the solution based on DP?
2 Share Reply

maxwellNorah ★92 February 17, 2020 8:43 AM
Why would there be n = 0 in the test case? Question says BST that stores value 1...n, seems a little unnecessary to test n = 0. If you want to be rigorous all the way, might as well test negative integers since n is only defined as an integer.
2 Share Reply

henrykira ★221 April 15, 2019 9:12 PM
Why the complexity is n*Gn but not n*G_n-1? I think once you pick the root, it is no longer Gn for building that certain tree?
1 Share Reply

smitamandal1 ★1 November 28, 2019 4:21 PM Report
Can someone help with the time complexity analysis of this approach, the 4^n/n^(3/2) part?
1 Share Reply

neildawg ★165 January 30, 2019 12:53 PM
what's does this if (start > end) { mean?
0 Share Reply
SHOW 2 REPLIES

‹ 1 2 ›