Problem 95: Unique Binary Search Trees II

Problem Information

Difficulty: Medium
Acceptance Rate: 61.45%
Paid Only: No
Tags: Dynamic Programming, Backtracking, Tree, Binary Search Tree, Binary Tree

Problem Description

Given an integer `n`, return _all the structurally unique**BST '**s (binary search trees), which has exactly _`n` _nodes of unique values from_ `1` _to_ `n`. Return the answer in **any order**.

Example 1:

![](https://assets.leetcode.com/uploads/2021/01/18/uniquebstn3.jpg)

Input: n = 3 **Output:** [[1,null,2,null,3],[1,null,3,2],[2,1,3],[3,1,null,null,2],[3,2,null,1]]

Example 2:

Input: n = 1 **Output:** [[1]]

Constraints:

* `1 <= n <= 8`

Code Snippets

C++:

```
/**
* Definition for a binary tree node.
* struct TreeNode {
```

```
* int val;
* TreeNode *left;
* TreeNode *right;
* TreeNode() : val(0), left(nullptr), right(nullptr) {}
* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
* TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left),
right(right) {}
* };
*/
class Solution {
public:
vector<TreeNode*> generateTrees(int n) {

}
};
```

Java:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* int val;
* TreeNode left;
* TreeNode right;
* TreeNode() {}
* TreeNode(int val) { this.val = val; }
* TreeNode(int val, TreeNode left, TreeNode right) {
* this.val = val;
* this.left = left;
* this.right = right;
* }
* }
*/
class Solution {
public List<TreeNode> generateTrees(int n) {

}
}
```

Python3:

```
# Definition for a binary tree node.
# class TreeNode:
# def __init__(self, val=0, left=None, right=None):
# self.val = val
# self.left = left
# self.right = right
class Solution:
def generateTrees(self, n: int) -> List[Optional[TreeNode]]:
```