Documentation for "95. Unique Binary Search Trees II"

Problem Description

Given an integer n, the task is to return all the structurally unique Binary Search Trees (BSTs) that have exactly n nodes with unique values from 1 to n. The output can be in any order.

Example 1:

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 leq n leq 8)

Solution Explanation

The solution involves generating all unique BSTs using a recursive approach. The primary idea is to select each number between 1 and n as the root, and recursively generate all possible left and right subtrees.

Explanation of the Code

1. TreeNode Class:

- Defines the structure of a tree node with a value (val), and pointers to the left and right children (left and right).

2. Solution Class:

- *generateTrees(self, n: int) -> List[Optional[TreeNode]]:*
    - ✓ This is the main function that starts the tree generation process.
    - ✓ It returns an empty list if n is 0.
    - ✓ Otherwise, it calls the helper function generate_trees with the range from 1 to n and an empty memoization dictionary.

3. generate_trees(self, start, end, memo):

- This is a recursive helper function that generates all unique BSTs for the given range (start to end).
- *Base Case:* If start is greater than end, return a list containing None indicating no tree can be formed.
- *Memoization:* Uses a dictionary memo to store results of previously computed ranges to avoid redundant computations.
- *Recursive Case:*
    - ✓ Iterates over each value i from start to end.
    - ✓ For each i, recursively generates all possible left and right subtrees.

    - ✓ Combines each left subtree with each right subtree and attaches them to the current root i.
- *Memoization Update:* Stores the list of all trees formed for the current range in memo.
- Returns the list of all trees formed for the current range.

Key Points

- **Recursive Tree Generation:** The solution leverages recursion to explore all possible tree structures.
- **Memoization:** Efficiently stores intermediate results to avoid redundant calculations and improve performance.
- **Tree Construction:** By systematically selecting each number as a root and combining all possible left and right subtrees, the solution ensures that all unique BSTs are generated.

This documentation explains the problem, provides the solution code, and details the logic and implementation of the solution.