# ****Leetcode 95 – Unique Binary Search Trees II****

## Problem Understanding

You're given an integer n, and you must return **all structurally unique BSTs (binary search trees)** that store values 1 to n.

### Conditions:

* The BST must store values from 1 to n.
* Each node must follow the BST rule: left < root < right.
* You must generate **all unique tree structures**.

## Optimized Java Solution (Recursive Divide and Conquer)

class Solution {

public List<TreeNode> generateTrees(int n) {

if (n == 0) return new ArrayList<>();

return build(1, n);

}

private List<TreeNode> build(int start, int end) {

List<TreeNode> result = new ArrayList<>();

if (start > end) {

result.add(null); // Empty tree

return result;

}

for (int i = start; i <= end; i++) {

List<TreeNode> leftTrees = build(start, i - 1);

List<TreeNode> rightTrees = build(i + 1, end);

for (TreeNode left : leftTrees) {

for (TreeNode right : rightTrees) {

TreeNode root = new TreeNode(i);

root.left = left;

root.right = right;

result.add(root);

}

}

}

return result;

}

}

## Dry Run Using Table (n = 3)

|  |  |  |  |
| --- | --- | --- | --- |
| Root | Left Subtrees (range) | Right Subtrees (range) | Unique Combinations |
| 1 | [] | [2,3] | 2 trees |
| 2 | [1] | [3] | 1 × 1 = 1 tree |
| 3 | [1,2] | [] | 2 trees |

Total trees = 5 for n = 3.

Example trees:

* Tree 1: 1 → null ← 2 ← 3
* Tree 2: 1 ← null ← 3 ← 2
* ...

## Time / Space Complexity

|  |  |
| --- | --- |
| Metric | Value |
| Time | Catalan Number C(n) growth, exponential |
| Space | O(C(n)) + recursion stack |

Number of trees is exponential (Catalan sequence):  
C(3) = 5, C(4) = 14, C(5) = 42, etc.

## Alternate Approaches

|  |  |  |
| --- | --- | --- |
| Approach | Time Complexity | Notes |
| ✅ Recursive (Divide) | Exponential | Clean and intuitive |
| DP + Memoization | Better for large n | Avoid recomputation |