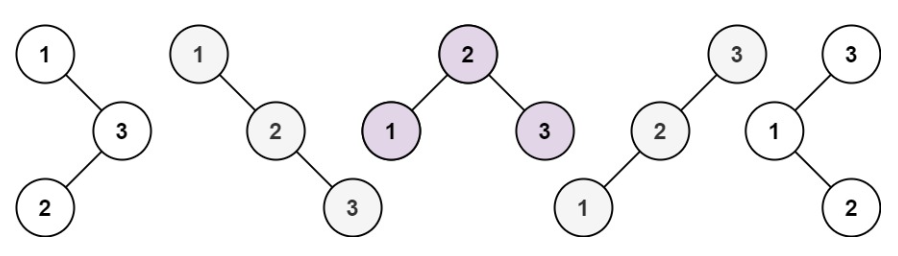
# 题目

给你一个整数n，请你生成并返回所有由n个节点组成且节点值从1到n互不相同的不同二叉搜索树。可以按任意顺序返回答案。

示例 1：



输入：n = 3

输出：[[1,null,2,null,3],[1,null,3,2],[2,1,3],[3,1,null,null,2],[3,2,null,1]]

示例 2：

输入：n = 1

输出：[[1]]

提示：

1 <= n <= 8

# 分析

## 方法一：递归/模拟

这是一个经典的动态规划问题，可以使用递归和记忆化搜索来解决。具体步骤如下：

1、定义一个函数generateTrees(start, end)，用于生成从start到end之间的所有不同二叉搜索树的根节点。

- 如果start > end，说明当前区间为空，返回一个空指针。

- 遍历区间[start, end]，以每个数字i作为根节点，递归生成左右子树，并将其连接到根节点上。

- 将生成的二叉搜索树的根节点加入到结果集中。

2、在主函数中，调用generateTrees(1, n)来生成所有的不同二叉搜索树。

以下是实现这个算法的代码：

class Solution {

public:

vector<TreeNode\*> generateTrees(int n) {

if (n == 0) return {};

return generateTrees(1, n);

}

vector<TreeNode\*> generateTrees(int start, int end) {

vector<TreeNode\*> trees;

if (start > end) {

trees.push\_back(nullptr);

return trees;

}

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

vector<TreeNode\*> leftTrees = generateTrees(start, i - 1);

vector<TreeNode\*> rightTrees = generateTrees(i + 1, end);

for (auto& left : leftTrees) {

for (auto& right : rightTrees) {

TreeNode\* root = new TreeNode(i);

root->left = left;

root->right = right;

trees.push\_back(root);

}

}

}

return trees;

}

};

这个算法的时间复杂度是 O(4^n / sqrt(n))，其中 n 是节点数。