挑战性题目DSCT302：求不同形态的平衡二叉树数目

问题描述：*n*个节点的不同形态的平衡二叉树数目是确定的。任意输入一个正整数*n*，请问有多少种不同的平衡二叉树形状？例如，输入3，输出1；输入4，输出4。

思路分析[[1]](#footnote-1)与算法描述[[2]](#footnote-2)：

本人实现了基于动态规划的AVL树形态数目求解，并通过斐波那契数列函数对给定节点数目的AVL树的最大深度进行了优化。下面将分别从这两个方面进行叙述。

平衡二叉树的基本特点是，树的左右两侧的子树高度差不超过1，且两棵子树均为平衡二叉树。给定节点数目N，求解可能的AVL树形态数目，考虑利用动态规划进行求解。

因为树的形态与节点数N与树深度H均有关，故动态规划函数的自变量为节点数目N与树深度H，函数值为形态数目。由于当求解了小规模AVL树的形态数目后，大规模AVL树可以在该子问题的求解基础上直接利用已有的结果，故本问题满足最优子结构，且无后效性，故可以利用动态规划进行求解。下面讨论问题的状态转移方程与初始条件。

记动态规划数组为AVL[N][H],则容易知道AVL[0][0]=1，AVL[0][1]=AVL[1][0]=0，AVL[1][1]=1。因为不存在深度为0的AVL树，也不存在节点数为0的AVL树，但需注意为了状态转移的可行，我们定义深度与节点数同时为0的AVL树由1棵。考虑到AVL树的左右两侧高度差最多为1，故在考虑状态转移方程时应分为如下三种情况进行考虑：

* 左右子树高度相同；
* 左子树比右子树深；
* 右子树比左子树深。

当确定了上述三种情况之一后，又由于除去根节点后的余下的N-1个节点均可能位于左右子树中的任意一棵，因此为了考虑全面，需要再对左右子树的所有可能节点数目进行考虑。对此，应用乘法与加法原理后，我们有转移方程如下所示：



其中左子树较深与右子树较深的情况完全对称，故直接考虑2倍的数量即可。

下面探讨AVL树深度的优化。设一棵深度为H的AVL树至少需要的节点数目为。为了考虑一棵平衡二叉树所能达到的最大深度，我们考虑AVL树的左右两侧不平衡，即高度差为1，这样可以在同等节点数目的情况下达到更大的深度。因此，上述高度为H的AVL树的最少节点数目的递推公式如下：



容易发现，这便是斐波那契数列的形式，初始值与边界条件在上式中已经给出。由于表达的是：若一棵AVL树想达到深度H，则至少需要个节点。因此，若给定节点数N，则可以通过如下公式判断其可能的最大深度：



利用此公式，不仅可以减小AVL数组的初始化大小以减少空间复杂度，还可以在AVL数组的递推求解中减小循环的次数，从而减小时间复杂度。

下面分析算法的时间复杂度。由于动态规划数组AVL为二维数组，且在求解时需要对左右子树的所有可能节点数情况进行遍历，故时间复杂度约为，空间复杂度约为。其中的来自上述深度优化，由斐波那契数列通项公式求解而来。

1. 算法思路不但要给出解题的算法内涵，还应该分析该算法的时间复杂度。此页背面也可以答题，但不应续页。 [↑](#footnote-ref-1)
2. 代码的写作和测试建议使用GCC、G++等通用C/C++编译器进行编译以利于跨平台的性能测试。作业纸质版本和测试的可执行文件请提交给主管助教。测试样例命令格式为：DSCT302\_2018270103012.exe 4，输出结果样例为：4。 [↑](#footnote-ref-2)