1. 设计一个递归算法，求在给定二叉树的结点总数N的情况下，二叉树可能拥有的形状数M。请看如下说明和要求：
2. 满足要求的任何一棵二叉树都是高度为N的满二叉树从根结点开始的子树。将这棵满二叉树的结点按照从上至下、从左至右的顺序进行编号，根结点的编号为1，则可以按层次输出任何结点总数为N的二叉树的所有结点编号。例如当N=3时，输出结果为：

1: 1, 2, 3

2: 1, 2, 4

3: 1, 2, 5

4: 1, 3, 6

5: 1, 3, 7

tree\_count is 5 when N is 3

1. 递归函数的原型是：void arrange(int arr[],int idx,int N,int &tree\_count)；其中arr是存放编号序列的数组，idx是当前需要计算的数组元素的下标，N是结点总数也是数组长度，tree\_count记录二叉树的数目。
2. 并不需要实际构造出一棵二叉树，只需要把各种可能的合法编号序列计算出来就能统计二叉树的数目，因此重点是编写算法arrange()，要求输出每一个合法的序列；
3. 请通过程序验证M和N之间满足卡塔兰数的关系，即。
4. 每一棵二叉树的高度是由其最后（最大）的结点编号决定的，设此编号为n，则二叉树的高度是。将arrange()函数的原型改造一下，变成void arrange(int arr[],int idx,int N,int &tree\_count,int &height)；用参数height记录所有二叉树的总高度，并利用height/M来计算平均高度；
5. 猜测并验证二叉树的平均高度与log2N之间的关系；
6. 当N逐渐增大，例如N>19，上述代码就会出错，原因是什么？如何修改？

二、用非递归（迭代）算法解决问题一。函数原型为：void buildtree(int N,int &tree\_count);即送入结点总数N，得到二叉树总数目tree\_count。整个算法的总体结构如：

…

while(idx>0){

…

if(…)

idx++;

else

idx--;

…

}

同问题一，需要设定一个长度为N的数组int arr[],idx为数组下标，当可以顺利确定当前下标上的结点编号之后，idx会自增以完成下一个位置上的计算。当idx到达N时，说明找到问题的一个解（即发现一棵新的二叉树），打印输出数组的内容。

语句idx--;是在当前位置上的所有合法取值都已经试验完毕，或者是刚刚输出了一个可行解之后执行的。

请仔细设计算法使得它具有尽可能低的算法复杂度。