1. **列出所有的情况之类的问题，一般都是递归（DFS）;**
2. **递归回溯理解：**

所谓Backtracking都是这样的思路：**在当前局面下，你有若干种选择。那么尝试每一种选择。如果已经发现某种选择肯定不行（因为违反了某些限定条件），就返回；如果某种选择试到最后发现是正确解，就将其加入解集**  
  
所以你思考递归题时，只要明确三点就行：**选择 (Options)，限制 (Restraints)，结束条件 (Termination)**。

1. **卡特兰数:**

一、关于卡特兰数

卡特兰数是一种经典的组合数，经常出现在各种计算中，其前几项为 : 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452, ...

二、卡特兰数的一般公式

卡特兰数满足以下性质：

令h(0)=1,h(1)=1，catalan数满足递推式。h(n)= h(0)\*h(n-1)+h(1)\*h(n-2) + ... + h(n-1)h(0) (n>=2)。也就是说，如果能把公式化成上面这种形式的数，就是卡特兰数。

当然，上面这样的递推公式太繁琐了，于是数学家们又求出了可以快速计算的通项公式。h(n)=c(2n,n)-c(2n,n+1)(n=0,1,2,...)。这个公式还可以更简单得化为h(n)=C(2n,n)/(n+1)。后一个公式都可以通过前一个公式经过几步简单的演算得来，大家可以拿起笔试试，一两分钟就可以搞定。

三、卡特兰数的应用

卡特兰数经常出现在OI以及ACM中，在生活中也有广泛的应用。下面举几个例子。

1、出栈次序：一个栈（无穷大）的进栈次序为1、2、3……n。不同的出栈次序有几种。

我们可以这样想，假设k是最后一个出栈的数。比k早进栈且早出栈的有k-1个数，一共有h(k-1)种方案。比k晚进栈且早出栈的有n-k个数，一共有h(n-k)种方案。所以一共有h(k-1)\*h(n-k)种方案。显而易见，k取不同值时，产生的出栈序列是相互独立的，所以结果可以累加。k的取值范围为1至n，所以结果就为h(n)= h(0)\*h(n-1)+h(1)\*h(n-2) + ... + h(n-1)h(0)。

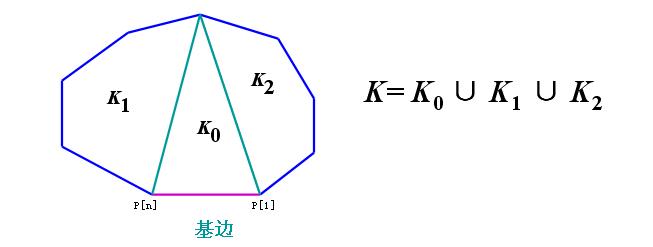
出栈入栈问题有许多的变种，比如n个人拿5元、n个人拿10元买物品，物品5元，老板没零钱。问有几种排队方式。熟悉栈的同学很容易就能把这个问题转换为栈。值得注意的是，由于每个拿5元的人排队的次序不是固定的，所以最后求得的答案要\*n!。拿10元的人同理，所以还要\*n!。所以这种变种的最后答案为h(n)\*n!\*n!。

2、二叉树构成问题。有n个结点，问总共能构成几种不同的二叉树。

我们可以假设，如果采用中序遍历的话，根结点第k个被访问到，则根结点的左子树有k-1个点、根结点的右指数有n-k个点。k的取值范围为1到n。讲到这里就很明显看得出是卡特兰数了。这道题出现在2015年腾讯实习生的在线笔试题中。有参加过的同学想必都有印象。

3、凸多边形的三角形划分。一个凸的n边形，用直线连接他的两个顶点使之分成多个三角形，每条直线不能相交，问一共有多少种划分方案。

这也是非常经典的一道题。我们可以这样来看，选择一个基边，显然这是多边形划分完之后某个三角形的一条边。图中我们假设基边是p1pn，我们就可以用p1、pn和另外一个点假设为pi做一个三角形，并将多边形分成三部分，除了中间的三角形之外，一边是i边形，另一边是n-i+1边形。i的取值范围是2到n-1。所以本题的解c(n)=c(2)\*c(n-1)+c(3)\*c(n-2)+...c(n-1)\*c(2)。令t(i)=c(i+2)。则t(i)=t(0)\*t(i-1)+t(1)\*t(i-2)...+t(i-1)\*t(0)。很明显，这就是一个卡特兰数了。



4、其他。诸如括号匹配问题、01序列问题、n边形格子从左下角走到右上角不跨过对角线问题。这些都是卡特兰数，其他问题也基本上是上面问题的变种。证明过程就不再赘述了。

四、卡特兰数通项公式的证明。

下面这篇文章图文并茂、写得很好，我没有把握写得比他更好。大家想看可以点链接。<http://blog.sina.com.cn/s/blog_6917f47301010cno.html>

1. **动态规划练习：（后一个状态是由前一个状态所决定的）**

**动态规划是一种算法思路。**

**它是利用空间存储历史信息使得未来需要历史信息时不需要重新计算，从而达到降低时间复杂度，是一种用空间换取时间的方法。**

**动态规划题目的基本思路：**

**首先我们要决定要存储什么历史信息以及用什么数据结构来存储信息。然后是最重要的递推式，就是如何从存储的历史信息中得到当前步骤的结果。最后我们需要考虑的就是起始条件的值。**

1. 例题(Best Time to Buy and Sell Stock), 用局部最优和全局最优的解法。

构造递推式，当前状态等于f(前一个状态)，f代表一种映射。

例题，爬梯子问题。

1. 求数组中最长递增序列的长度

由于这个最长上升序列不一定是连续的，对于每一个新加入的数，都有可能跟前面的序列构成一个较长的上升序列，或者跟后面的序列构成一个较长的上升序列。比如1,3,5,2,8,4,6，对于6来说，可以构成1,3,5,6，也可以构成2,4,6。因为前面那个序列长为4，后面的长为3，所以我们更愿意6组成那个长为4的序列，所以对于6来说，它组成序列的长度，实际上是之前最长一个升序序列长度加1，注意这个最长的序列的末尾是要小于6的，不然我们就把1,3,5,8,6这样的序列给算进来了。这样，我们的递推关系就隐约出来了，假设dp[i]代表加入第i个数能构成的最长升序序列长度，我们就是要在dp[0]到dp[i-1]中找到一个最长的升序序列长度，又保证序列尾值nums[j]小于nums[i]，然后把这个长度加上1就行了。同时，我们还要及时更新最大长度。

1. 求最长回文子串，求最长回文子序列 (用一个二维数组来记录)
2. 硬币找零问题（背包问题）