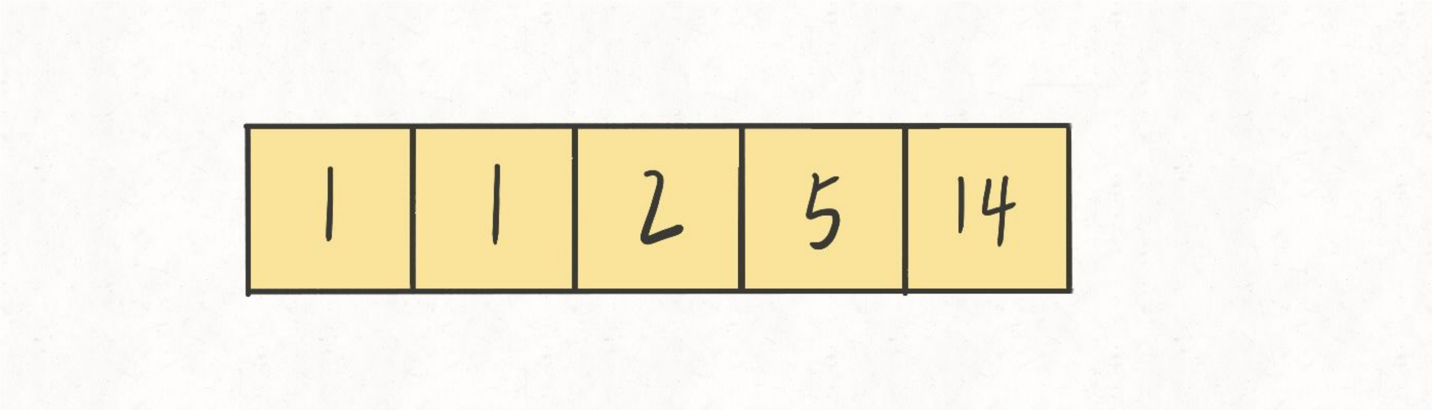
Catalan（卡特兰数）



> 明安图数，又称卡塔兰数，英文名Catalan number，是组合数学中一个常出现于各种计数问题中的数列。
> 以中国蒙古族数学家明安图 (1692-1763)和比利时的数学家欧仁·查理·卡塔兰 (1814–1894)的名字来命名，其前几项为（从第零项开始）：1, 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, ...
>
> 卡特兰数Cn满足以下递推关系 [2]：
>
> $$1.C_{n+1}=C_0C_n+C_1C_{n-1}+\cdots+C_nC_0$$
> $$2.(n-3)C_n=\frac{n}{2}(C_3C_{n-1}+C_4C_{n-2}+C_5C_{n-3}+\cdots+C_{n-2}C_4+C_{n-1}C_3)$$
>
> 摘抄自百度百科
> https://baike.baidu.com/item/卡特兰数/6125746?fr=aladdin

我们先从简单的入手 （以前一道周练水题）

P1044 [NOIP2003 普及组] 栈

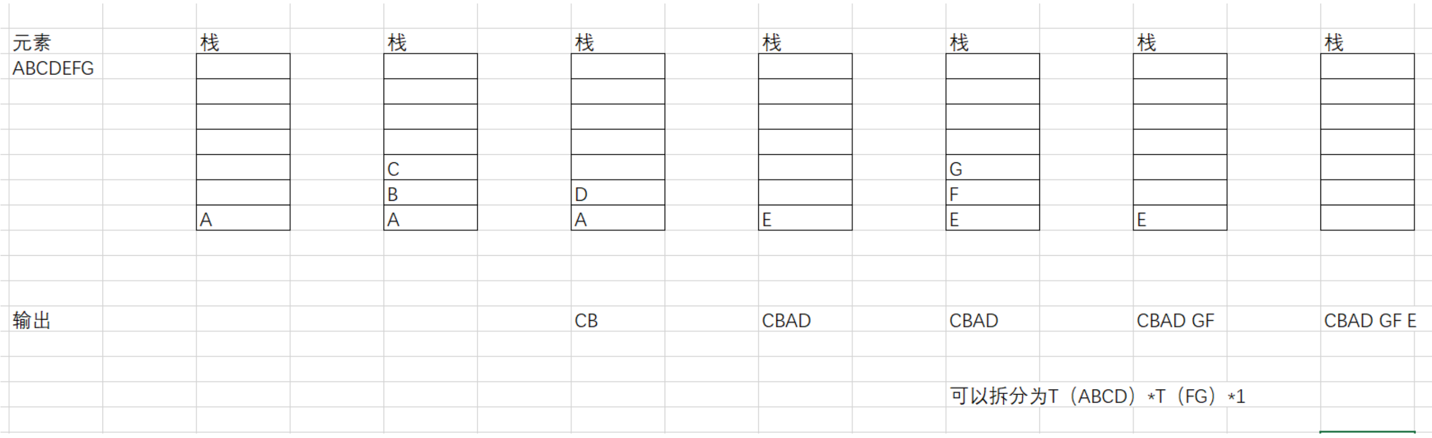
- 对于没给元素（输入为0）
  - **结果设 1 （稍后分析为什么设1）**

- 对于只给一个元素（输入为1）
  - **结果为 1**

- 对于两个及以上元素（输入为n）
  1. 把每个元素最后输出都视为一种情况，全部累加就是最终结果
  2. 对于每一种情况，都是由最后输出的那个元素把栈分为了两个独立的情况

例如 有 `A B C D E F G` 七个元素要进栈（输入n=7）
假设 `E` 为最后出栈元素(即最后输出) 那么对于 `E` 前面的 `ABCD` 的出栈次序，和 `E` 后面的 `FG` 的出栈次序将互不干扰

如果计 `f（ABCD）` 为ABCD所有的情况， `f(EG)` 为FG的所有情况，那么可以用 `f（ABCD）*f（FG）` 来表示E最后输出的情况有的结果个数



E作为最后的输出元素

同理，对于其他元素依次计算最后输出时，所产生的结果数，最后相加就行了

所以对于上述7元素ABCDEFG的情况：

计算过程

> 当A最后输出 f(0)*f(6)
> 当B最后输出 f(1)*f(5)
> 当C最后输出 f(2)*f(4)
> 当D最后输出 f(3)*f(3)
> 当E最后输出 f(4)*f(2)
> 当F最后输出 f(5)*f(1)
> 当G最后输出 f(6)*f(0)
>
> 总数f(7)=上述结果之和

不难发现，想要知道f(7),就需要知道f(6)、f(5)、f(4)...
最终像递归一样的转化到子问题f(1)、f(0)

f(1)是只有一个元素时的情况，结果为1

不能因为没有元素而认为f(0)=0,要注意f(0)代表 没有元素 这一种情况，与别的数（如f(6)）乘积时要保留情况，所以f(0)应该为1，而且这也符合catalan数的定义

所以要模拟计算过程，需要从小的开始依次推到大的。
附上我的代码：

```
#include <iostream>
using namespace std;
int main()
{
    long long int a[20] = {0};
    a[0] = 1;
    int n = 0;
    scanf("%d", &n);
    int i = 1;
    while (i <= n)
    {
        int j = 1;  //j就代表着最后一个出栈的元素
        while (j >= 1 && j <= i)    //每种情况都要算一次并相加
        {
            a[i] += a[j - 1] * a[i - j];
            j++;
        }
        i++;
    }
    printf("%lld", a[n]);
    return 0;
}
```

另一个和刚刚类似的例题
96. 不同的二叉搜索树
https://leetcode-cn.com/problems/unique-binary-search-trees/
一样的原理来思考，和刚刚答案十分一致，无非是让根节点把左右分开，左右两棵子树独立，来和上题一样

```
class Solution {
public:
    int numTrees(int n) {
    long long int a[20] = {0};
    a[0] = 1;
    int i = 1;
    while (i <= n)
    {
        int j = 1;
        while (j >= 1 && j <= i)
        {
            a[i] += a[j - 1] * a[i - j];
            j++;
        }
        i++;
    }
    return a[n];
    }
};
```

上面两道题目都是因为题目数据小，直接递归就行，但是当n大时，不仅要高精度，还要提高效率 对于卡特兰数，f(N)是根据前面各数的迭代求和
通过演算，可以得出f(N)的各种表达式

定义

递归定义

$f_n = f_0 * f_{n-1} + f_1 * fn - 2 + ... + f_{n-1} f_0$，其中$n \geq 2$

递推关系

$f_n = \frac{4n-2}{n+1} f_{n-1}$

通项公式

$f_n = \frac{1}{n+1} C_{2n}^{n}$

经化简后可得

$f_n = C_{2n}^{n} - C_{2n}^{n-1}$

只要我们在解决问题时得到了上面的一个关系，那么你就已经解决了这个问题，因为他们都是卡特兰数列

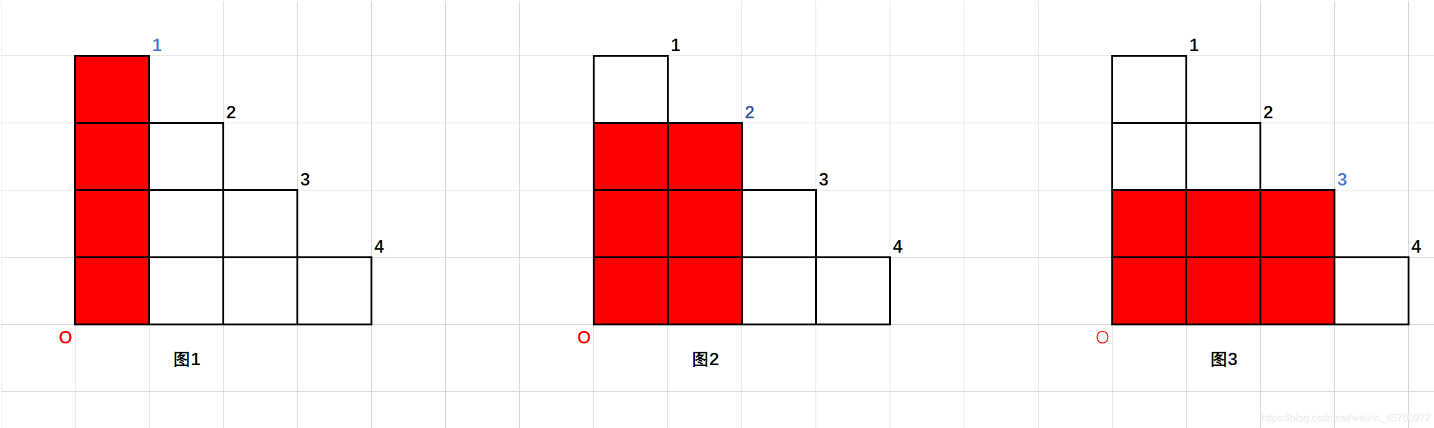
懂了Catalan数求解原理以及表达式后，就可以来思考这题
P2532 [AHOI2012]树屋阶梯

https://www.luogu.com.cn/problem/P2532

因为题目要求，用n个方块，叠层n层阶梯，就意味着每一个角都对于一个方块，方块没办法分割

比如下图对于n=4时，有四个角（1，2，3，4标的地方）

用红色的区域可以把阶梯分开，分为左上角，右下角两部分
如图2，左上角为1层的阶梯，右下角为2层的阶梯，这可以用刚刚的方法来处理，也就是 f（1）*f（2）
由此来想，这和Catalan数的递推是一样的，所以结果是卡特兰数，f(1)=1,f(2)=2,f(3)=5...



对于这题，由于n较大，用递归或递推表达式来算会运算好多步，所以直接考虑通项公式，注意高精度

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e4 + 5;
struct num
{
    int a[N], l;
} ans;
int n;
void cheng(num &b, int x) //chengfa
{
    int p = 0; //p是记录进位用的
    for (int i = 1; i <= b.l; i++)
    {
        b.a[i] = b.a[i] * x + p;
        p = b.a[i] / 10;
        b.a[i] %= 10;
    }
    while (p)
    {
        b.a[++b.l] = p;
        p /= 10;
        b.a[b.l] %= 10;
    }
}
void chu(num &b, int x) //除法
{
    int p = 0;
    for (int i = b.l; i >= 1; i--)
    {
        p = p * 10 + b.a[i];
        b.a[i] = p / x;
        p %= x;
    }
    while (!b.a[b.l])
        b.l--;
}
int main()
{
    scanf("%d", &n);
    ans.a[1] = ans.l = 1;
    for (int i = 2; i <= n; i++)
        cheng(ans, i + n);
    for (int i = 2; i <= n; i++)
        chu(ans, i);
    for (int i = ans.l; i >= 1; i--)
        printf("%d", ans.a[i]);
    return 0;
}
```

Catalan数列主要的作用是求有多少种情况，而不是列举每一种情况具体是什么样子。

此外还有 阶梯形路径走法个数，凸多边形的三角形划分，01序列，括号配对（本质也是01序列）等计数问题都可以用分析转化为Catalan数列，来解决问题