递推算法

（课件来源：奥赛一本通）

讲师：王骏峣

递推法是一种重要的数学方法，在数学的各个领域中都有广泛的运用，也是计算机用于数值计算的一个重要算法。这种算法特点是：一个问题的求解需一系列的计算，在已知条件和所求问题之间总存在着某种相互联系的关系，在计算时，如果可以找到前后过程之间的数量关系（即递推式），那么，从问题出发逐步推到已知条件，此种方法叫逆推。无论顺推还是逆推，其关键是要找到递推式。这种处理问题的方法能使复杂运算化为若干步重复的简单运算，充分发挥出计算机擅长于重复处理的特点。

转移. 方程

初始化.

递归边界.

递推算法的首要问题是得到相邻的数据项间的关系（即递推关系）。递推算法避开了求通项公式的麻烦，把一个复杂的问题的求解，分解成了连续的若干步简单运算。一般说来，可以将递推算法看成是一种特殊的迭代算法。

斐波那契

$$fib_i = fib_{i-1} + fib_{i-2}$$

【例1】数字三角形。如下所示为一个数字三角形。请编一个程序计算从顶到底的某处的一条路径，使该路径所经过的数字总和最大。只要求输出总和。

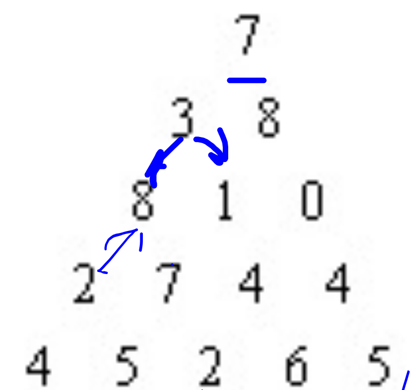
1、 一步可沿左斜线向下或右斜线向下走；

2、 三角形行数小于等于100；

3、 三角形中的数字为0，1，...，99；

测试数据通过键盘逐行输入，如上例数据应以如下所示格式输入：

5
7
3 8
8 1 0
2 7 4 4
4 5 2 6 5



【算法分析】

此题解法有多种，从递推的思想出发，设想，当从顶层沿某条路径走到第i层向第i+1层前进时，我们的选择一定是沿其下两条可行路径中最大数字的方向前进，为此，我们可以采用倒推的手法，设a[i][j]存放从i,j 出发到达n层的最大值，则a[i][j]=max{a[i][j]+a[i+1][j]，a[i][j]+a[i+1][j+1]}，a[1][1] 即为所求的数字总和的最大值。

a[i+1][j] += a[i][j];

a[i+1][j+1] += a[i][j];

【参考程序】

```
#include<iostream>
using namespace std;
int main()
{
  int n,i,j,a[101][101];
  cin>>n;
  for (i=1;i<=n;i++)
   for (j=1;j<=i;j++)
     cin>>a[i][j];                          //输入数字三角形的值
  for (i=n-1;i>=1;i--)
   for (j=1;j<=i;j++)
     {
       if (a[i+1][j]>=a[i+1][j+1]) a[i][j]+=a[i+1][j];//路径选择
       else  a[i][j]+=a[i+1][j+1];
     }
   cout<<a[1][1]<<endl;
}
```

P1258 P1341 P1345 P1211 P1762

【例2】满足F1=F2=1，Fn=Fn-1+Fn-2的数列称为斐波那契数列（Fibonacci），它的前若干项是1，1，2，3，5，8，13，21，34……求此数 列第n项（n>=3）。

即：f1=1　　　　　　　（n=1）
　　f2=1　　　　　　　（n=2）
　　fn=fn-1 + fn-2　　（n>=3）

$n=2$　　$0: n-2$
　　　　$1: n-1$

- **程序如下：**

```
#include<iostream>
#include<cstdio>
using namespace std;
int main()
{
      int f0=1, f1=1, f2;
      int n;
      cin>>n;
 for (int i=3; i<=n; ++i)
     {
           f2=f0+f1;
           f0=f1;
           f1=f2;
     }
printf("%d\n",f2);
return 0;
}
```

$f_n = f_{n-2} + f_{n-1}$

$f_{n-2} = f_{n-1}$

$f_{n-1} = f_n$

$n{+}{+}.$

有关Fibonacci数列，我们先来考虑一个简单的问题：楼梯有n个台阶，上楼可以一步上一阶，也可以一步上两阶。一共有多少种上楼的方法？

这是一道计数问题。在没有思路时，不妨试着找规律。n=5时，一共有8种方法：

5=1+1+1+1+1	5=2+1+1+1
5=1+2+1+1	5=1+1+2+1
5=1+1+1+2	5=2+2+1
5=2+1+2	5=1+2+2

其中有5种方法第1步走了1阶(背景灰色)，3种方法第1步走了2阶，没有其他可能。假设f(n)为n个台阶的走法总数，把n个台阶的走法分成两类。

第1类：第1步走1阶，剩下还有n-1阶要走，有f(n-1)种方法。

第2类：第1步走2阶，剩下还有n-2阶要走，有f(n-2)种方法。

这样，就得到了递推式：f(n)=f(n-1)+f(n-2),不要忘记边界情况：f(1)=1,f(2)=2。把f(n)的前几项列出：1，2，3，5，8，……。

再例如，把雌雄各一的一对新兔子放入养殖场中。每只雌兔在出生两个月以后，每月产雌雄各一的一对新兔子。试问第n个月后养殖场中共有多少对兔子。

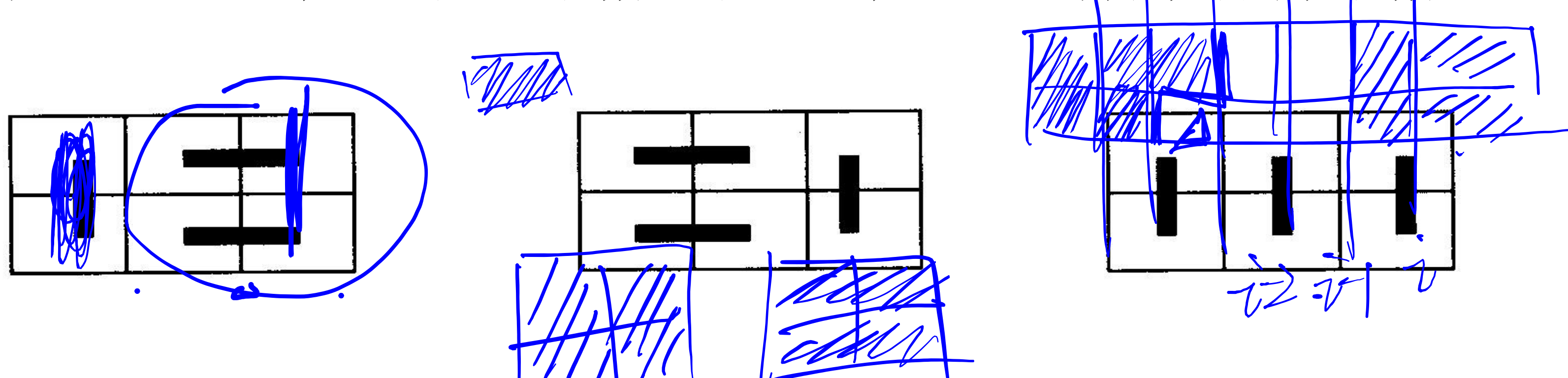
$f_i = f_{i-1} + f_{i-2}$

还是先找找规律。

第1个月：一对新兔子r1。用小写字母表示新兔子。
第2个月：还是一对新兔子，不过已经长大，具备生育能力了，用大写字母R1表示。
第3个月：R1生了一对新兔子r2，一共2对。
第4个月：R1又生一对新兔子r3，一共3对。另外，r2长大了，变成R2
第5个月：R1和R2各生一对，记为r4和r5，共5对。此外，r3长成R3。
第6个月：R1、R2和R3各生一对，记为r6～r8，共8对。此外，r4和r5长大。
……

把这些数排列起来：1，1，2，3，5，8，……，事实上，可以直接推导出来递推关系f(n)=f(n-1)+f(n-2):第n个月的兔子由两部分组成，一部分是上个月就有的老兔子f(n-1)，一部分是上个月出生的新兔子f(n-2)(第n-1个月时具有生育能力的兔子数就等于第n-2个月兔子总数)。根据加法原理，f(n)=f(n-1)+f(n-2)。

【例3】 有 2*n的一个长方形方格，用一个1*2的骨牌铺满方格。



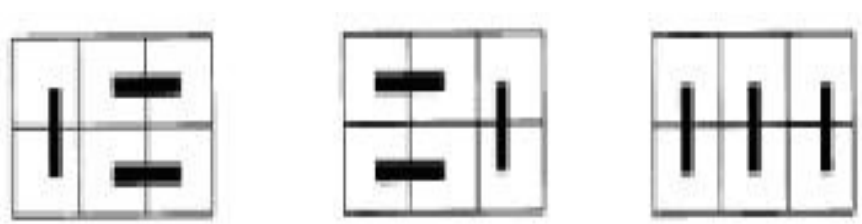
编写一个程序，试对给出的任意一个n(n>0), 输出铺法总数。

【算法分析】

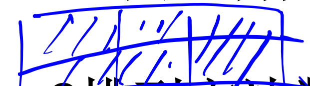
（1）面对上述问题，如果思考方法不恰当，要想获得问题的解答是相当困难的。可以用递推方法归纳出问题解的一般规律。

（2）当n=1时，只能是一种铺法，铺法总数有示为$x_1=1$。

（3）当n=2时：骨牌可以两个并列竖排，也可以并列横排，再无其他方法，如下左图所示，因此，铺法总数表示为$x_2=2$;



（4）当n=3时：骨牌可以全部竖排，也可以认为在方格中已经有一个竖排骨牌，则需要在方格中排列两个横排骨牌（无重复方法），若已经在方格中排列两个横排骨牌，则必须在方格中排列一个竖排骨牌。如上右图，再无其他排列方法，因此铺法总数表示为x3=3。



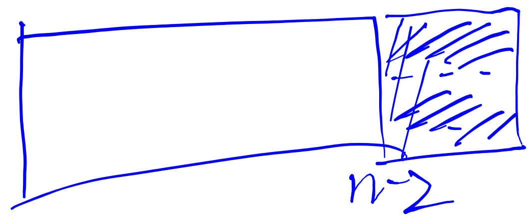
由此可以看出，当n=3时的排列骨牌的方法数是n=1和n=2排列方法数的和。

（5）推出一般规律：对一般的n，要求xn可以这样来考虑，若第一个骨牌是竖排列放置，剩下有n-1个骨牌需要排列，这时排列方法数为x_{n-1}；若第一个骨牌是横排列，整个方格至少有2个骨牌是横排列（1*2骨牌），因此剩下n-2个骨牌需要排列，这是骨牌排列方法数为x_{n-2}。从第一骨牌排列方法考虑，只有这两种可能，所以有：

$$x_n=x_{n-1}+x_{n-2} \quad （n>2）$$

$$x_1=1$$

$$x_2=2$$

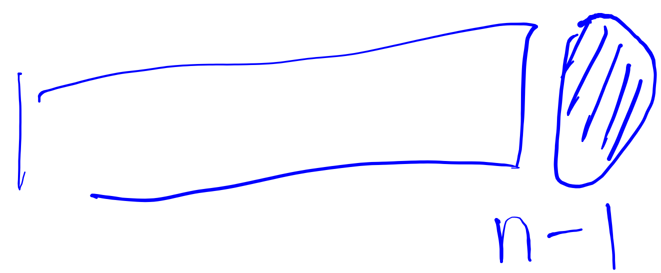


$x_n=x_{n-1}+x_{n-2}$就是问题求解的递推公式。任给n都可以从中获得解答。例如n=5，

$$x_3=x_2+x_1=3$$

$$x_4=x_3+x_2=5$$

$$x_5=x_4+x_3=8$$



$x_{n-1}+x_{n-2}.$

下面是输入n，输出x_1~x_n的c++程序：

```
#include<iostream>
using namespace std;
int main()
{
  int n,i,j,a[101];
  cout<<"input n:";
//输入骨牌数
  cin>>n;
  a[1]=1;a[2]=2;
  cout<<"x[1]="<<a[1]<<endl;
  cout<<"x[2]="<<a[2]<<endl;
  for (i=3;i<=n;i++)
//递推过程
   {
     a[i]=a[i-1]+a[i-2];

cout<<"x["<<i<<"]="<<a[i]<<endl;
     }
}
```

下面是运行程序输入 n=30，输出的结果：

```
input n: 30
   x[1]=1
   x[2]=2
   x[3]=3
 ........
   x[29]=832040
   x[30]=1346269
```

问题的结果就是有名的斐波那契数。

【例4】昆虫繁殖

$f_i = f_{i-1} + f_{i-2}$

兔子 1个月大的时候 可以生.

【问题描述】

科学家在热带森林中发现了一种特殊的昆虫，这种昆虫的繁殖能力很强。每对成虫过x个月产y对卵，每对卵要过两个月长成成虫。假设每个成虫不死，第一个月只有一对成虫，且卵长成成虫后的第一个月不产卵(过X个月产卵)，问过Z个月以后，共有成虫多少对？0=<X<=20,1<=Y<=20,X=<Z<=50

【输入格式】

x,y,z的数值

$f_i += f_{i-x} * y$

【输出格式】

过Z个月以后，共有成虫对数

模拟.

成虫:
1月卵:
0月卵.
$i-x$:
: 3+2+2

【输入样例】

1 2 8

1 2 8

第1个月: 1
第2个月: 1
3 : 1+2
4 : 1+2+2

【输出样例】

37

【参考程序】

```
#include<iostream>
using namespace std;
int main()
{
  long long a[101]={0},b[101]={0},i,j,x,y,z;
  cin>>x>>y>>z;
  for(i=1;i<=x;i++){a[i]=1;b[i]=0;}
  for(i=x+1;i<=z+1;i++)//因为要统计到第z个月后，所以要for到z+1
  {
    b[i]=y*a[i-x];
    a[i]=a[i-1]+b[i-2];
  }
  cout<<a[z+1]<<endl;
  return 0;
}
```

下课休息，到 10:10

【例5】位数问题

【问题描述】

在所有的N位数中，有多少个数中有偶数个数字3？由于结果可能很大，你只需要输出这个答案对12345取余的值。

【输入格式】

读入一个数N

【输出格式】

输出有多少个数中有偶数个数字3。

【输入样例】

2

【输出样例】

73

【数据规模】

1<=N<=1000

【样例说明】

在所有的2位数字，包含0个3的数有72个，包含2个3的数有1个，共73个

f[i][0] 表示前i位偶

f[i][1] 表奇数

f[i][0/1]

1	3
1	2
3	3

f[i][0] = f[i-1][1] + f[i-1][0] × 9

f[i][1] = f[i-1][0] + f[i-1][1] × 9

3 不含3

【算法分析】

方法一：排列组合(但需要运用动态规划)。

可以列出公式,在n个格子中放x个3(其中x为偶数,包括0).。

c(n,x)*9^(n-x)-c(n-1,x)*9^(n-x-1) 含义为在n个格子中取x个3,且不考虑第一位的特殊情况为c(n,x)*9^(n-x)。

而第一位为0的情况,为c(n-1,x)*9^(n-x-1),两者减下,就为答案。

方法二：递推

考虑这种题目,一般来说都是从第i-1位推导第i位,且当前位是取偶数还是取奇数的。

恍然大悟.可以用f[i][0]表示前i位取偶数个3有几种情况,f[i][1]表示前i位取奇数个3有几种情况。

则状态转移方程可以表示为:

f[i][0]=f[i-1][0]*9+f[i-1][1];f[i][1]=f[i-1][0]+f[i-1][1]*9;

边界条件:f[1][1]=1;f[1][0]=9;

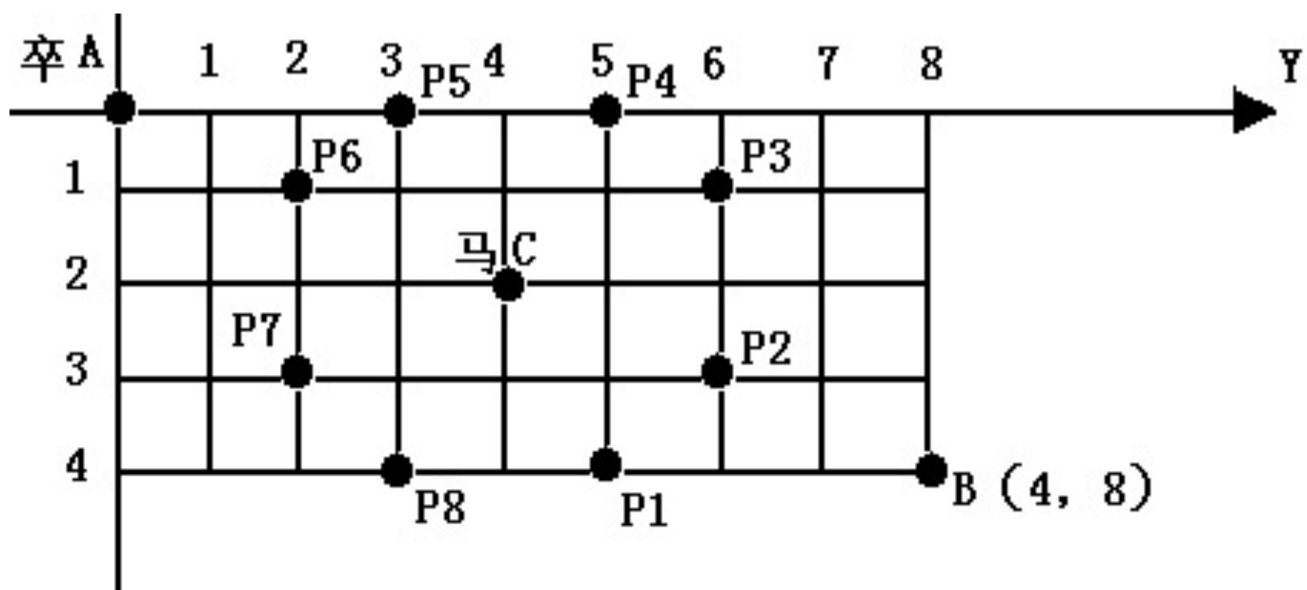
【参考程序】

```
#include<iostream>
using namespace std;
int main()
{
  int f[1001][2],n,i,x;
  cin>>n;
  f[1][1]=1;f[1][0]=9;
  for(i=2;i<=n;i++)
   {
      x=f[1][0];
      if(i==n)x--;
      f[i][0]=(f[i-1][0]*x+f[i-1][1])%12345;
      f[i][1]=(f[i-1][1]*x+f[i-1][0])%12345;
   }
   cout<<f[n][0];
   return 0;
}
```

【例6】过河卒（Noip2002）

【问题描述】

棋盘上A点有一个过河卒，需要走到目标B点。卒行走的规则：可以向下、或者向右。同时在棋盘上的任一点有一个对方的马（如C点），该马所在的点和所有跳跃一步可达的点称为对方马的控制点，如**图3-1**中的C点和P1，……，P8，卒不能通过对方马的控制点。棋盘用坐标表示，A点(0,0)、B点(n, m) (n,m为不超过20的整数),同样马的位置坐标是需要给出的，C≠A且C≠B。现在要求你计算出卒从A点能够到达B点的路径的条数。



【算法分析】

跳马是一道老得不能再老的题目，可能是在学回溯或搜索等算法的时候，很多书上也有类似的题目，一些比赛中也出现过这一问题的变形（如NOIP1997初中组的第三题）。有些同学一看到这条题目就去搜索，即使你编程调试全通过了，运行时你也会发现：当n,m=15就会超时。

其实，本题稍加分析就能发现，要到达棋盘上的一个点，只能从左边过来（我们称之为左点）或是从上面过来（我们称之为上点），所以根据加法原理，到达某一点的路径数目，就等于到达其相邻的上点和左点的路径数目之和，因此我们可以使用逐列（或逐行）递推的方法来求出从起点到终点的路径数目。障碍点（马的控制点）也完全适用，只要将到达该点的路径数目设置为0即可。

用F[i][j]表示到达点(i,j)的路径数目，g[i][j]表示点(i, j)有无障碍，g[i][j]=0表示无障碍，g[i][j]=1表示有障碍。

则，递推关系式如下：

F[i][j] = F[i-1][j] + F[i][j-1]　　　　//i>0且j>0且g[i][j]= 0

递推边界有4个：

F[i][j] = 0　　　　//g[i][j] = 1

F[i][0] = F[i-1][0]　　　　//i > 0且g[i][0] = 0

F[0][j] = F[0][j-1]　　　　//j > 0且g[0][j] = 0

F[0][0] = 1

考虑到最大情况下：n=20,m=20，路径条数可能会超过2^{31}-1,所以要用高精度。

五种典型的递推关系

Ⅰ.Fibonacci数列

在所有的递推关系中，Fibonacci数列应该是最为大家所熟悉的。在最基础的程序设计语言Logo语言中，就有很多这类的题目。而在较为复杂的Basic、Pascal、C语言中，Fibonacci数列类的题目因为解法相对容易一些，逐渐退出了竞赛的舞台。可是这不等于说Fibonacci数列没有研究价值，恰恰相反，一些此类的题目还是能给我们一定的启发的。

Fibonacci数列的代表问题是由意大利著名数学家Fibonacci于1202年提出的“兔子繁殖问题”(又称“Fibonacci问题”)。

问题的提出：有雌雄一对兔子，假定过两个月便可繁殖雌雄各一的一对小兔子。问过n个月后共有多少对兔子？

解：设满x个月共有兔子Fx对，其中当月新生的兔子数目为Nx对。第x-1个月留下的兔子数目设为Fx-1对。则：

$F_x=N_x+ F_{x-1}$

$N_x=F_{x-2}$ (即第x-2个月的所有兔子到第x个月都有繁殖能力了)

∴ $F_x=F_{x-1}+F_{x-2}$ 边界条件：F0=0，F1=1

由上面的递推关系可依次得到

$F_2=F_1+F_0=1$，$F_3=F_2+F_1=2$，$F_4=F_3+F_2=3$，$F_5=F_4+F_3=5$，……。

Fabonacci数列常出现在比较简单的组合计数问题中，例如以前的竞赛中出现的“骨牌覆盖”问题。在优选法中，Fibonacci数列的用处也得到了较好的体现。

Ⅱ.Hanoi塔问题

问题的提出：Hanoi塔由n个大小不同的圆盘和三根木柱a,b,c组成。开始时，这n个圆盘由大到小依次套在a柱上，如图3-11所示。

要求把a柱上n个圆盘按下述规则移到c柱上：

图3-11

n-1个圆盘

n个圆盘

(1)一次只能移一个圆盘；

(2)圆盘只能在三个柱上存放；

(3)在移动过程中，不允许大盘压小盘。

问将这n个盘子从a柱移动到c柱上，总计需要移动多少个盘次？

$$f[n]=f[n-1]*2+1$$

解：设hn为n个盘子从a柱移到c柱所需移动的盘次。显然，当n=1时，只需把a 柱上的盘子直接移动到c柱就可以了，故h1=1。当n=2时，先将a柱上面的小盘子移动到b柱上去；然后将大盘子从a柱移到c 柱；最后，将b柱上的小盘子移到c柱上，共记3个盘次，故h2=3。以此类推，当a柱上有n(n2)个盘子时，总是先借助c柱把上面的n-1个盘子移动到b柱上，然后把a柱最下面的盘子移动到c柱上；再借助a柱把b柱上的n-1个盘子移动到c柱上；总共移动$h_{n-1}+1+h_{n-1}$个盘次。

$\therefore h_n=2h_{n-1}+1$　边界条件：$h_1=1$

Ⅲ.平面分割问题

问题的提出：设有n条封闭曲线画在平面上，而任何两条封闭曲线恰好相交于两点，且任何三条封闭曲线不相交于同一点，问这些封闭曲线把平面分割成的区域个数。

解：设a_n为n条封闭曲线把平面分割成的区域个数。由图3-13可以看出：$a_2-a_1=2$；$a_3-a_2=4$；$a_4-a_3=6$。

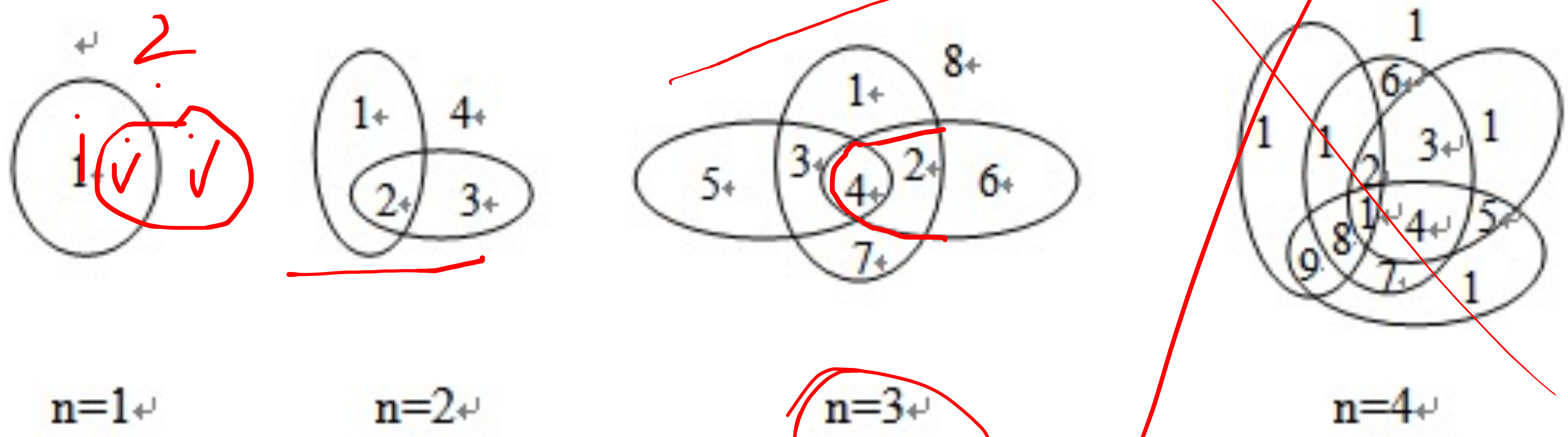


图 13

从这些式子中可以看出$a_n-a_{n-1}=2(n-1)$。当然，上面的式子只是我们通过观察4幅图后得出的结论，它的正确性尚不能保证。下面不妨让我们来试着证明一下。当平面上已有n-1条曲线将平面分割成a_{n-1}个区域后，第n-1条曲线每与曲线相交一次，就会增加一个区域，因为平面上已有了n-1条封闭曲线，且第n条曲线与已有的每一条闭曲线恰好相交于两点，且不会与任两条曲线交于同一点，故平面上一共增加2(n-1)个区域，加上已有的an-1个区域，一共有$a_{n-1}+2(n-1)$个区域。所以本题的递推关系是$a_n=a_{n-1}+2(n-1)$，边界条件是$a_1=1$。

平面分割问题是竞赛中经常触及到的一类问题，由于其灵活多变，常常感到棘手，下面的【例7】是另一种平面分割问题，有兴趣的读者不妨自己先试着求一下其中的递推关系。

IV.Catalan数

Catalan数首先是由Euler在精确计算对凸n边形的不同的对角三角形剖分的个数问题时得到的，它经常出现在组合计数问题中。

问题的提出：在一个凸n边形中，通过不相交于n边形内部的对角线，把n边形拆分成若干三角形，不同的拆分数目用h_n表示，h_n即为Catalan数。例如五边形有如下五种拆分方案(图3-14)，故h_5=5。求对于一个任意的凸n边形相应的h_n。

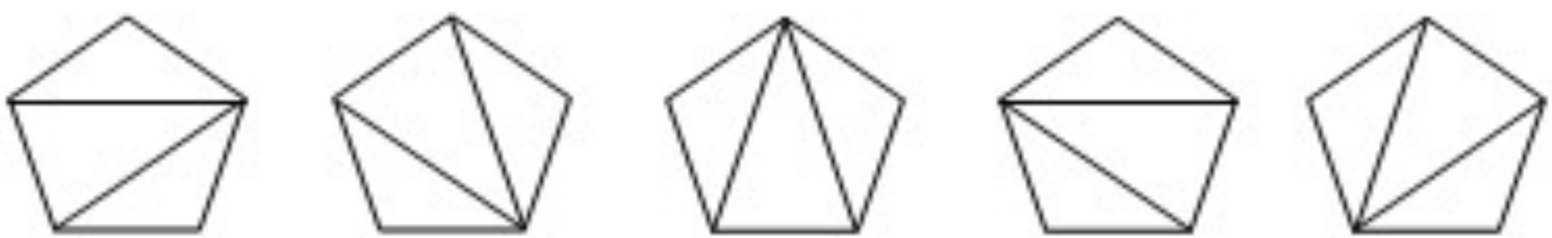


图 3-14

解：设 C_n 表示凸 n 边形的拆分方案总数。由题目中的要求可知一个凸 n 边形的任意一条边都必然是一个三角形的一条边，边 P_1 P_n 也不例外，再根据“不在同一直线上的三点可以确定一个三角形”，只要在 P_2，P_3，……，P_{n-1} 点中找一个点 $P_k(1<k<n)$，与 P_1、P_n 共同构成一个三角形的三个顶点，就将 n 边形分成了三个不相交的部分(如图 3-15 所示)，我们分别称之为区域①、区域②、区域③，其中区域③必定是一个三角形，区域①是一个凸 k 边形，区域②是一个凸 n-k+1 边形，区域①的拆分方案总数是 C_k，区域②的拆分方案数为 C_{n-k+1}，故包含△$P_1P_kP_n$ 的 n 边形的拆分方案数为 C_kC_{n-k+1} 种，而 P_k 可以是 P_2，P_3，……，P_{n-1} 种任一点，根据加法原理，凸 n 边形的三角拆分方案总数为 $\sum_{i=2}^{n-1} C_iC_{n-i+1}$，同时考虑到计算的方便，约定边界条件 $C_2=1$。

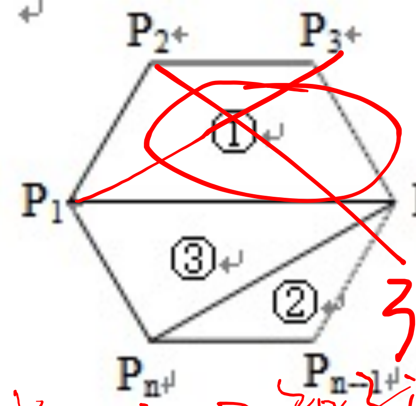


图 3-15

递推：已知子问题答案

递归：未知子问题

拆开子问题并解决

Catalan数是比较复杂的递推关系，尤其在竞赛的时候，选手很难在较短的时间里建立起正确的递推关系。当然，Catalan数类的问题也可以用搜索的方法来完成，但是，搜索的方法与利用递推关系的方法比较起来，不仅效率低，编程复杂度也陡然提高。

Ⅴ.第二类Stirling数【拓展】

在五类典型的递推关系中，第二类Stirling是最不为大家所熟悉的。也正因为如此，我们有必要先解释一下什么是第二类Strling数。

【定义2】n个有区别的球放到m个相同的盒子中，要求无一空盒，其不同的方案数用S(n,m)表示，称为第二类Stirling数。

下面就让我们根据定义来推导带两个参数的递推关系——第二类Stirling数。

解：设有n个不同的球，分别用b1,b2,......bn表示。从中取出一个球bn，bn的放法有以下两种：

①bn独自占一个盒子；那么剩下的球只能放在m-1个盒子中，方案数为S2(n-1,m-1)；

②bn与别的球共占一个盒子；那么可以事先将b1,b2,......bn-1这n-1个球放入m个盒子中，然后再将球bn可以放入其中一个盒子中，方案数为mS2(n-1,m)。

综合以上两种情况，可以得出第二类Stirling数定理：

【定理】$S2(n,m)=mS2(n-1,m)+S2(n-1,m-1) \quad (n>1,m1)$

边界条件可以由定义2推导出：

$S2(n,0)=0$；$S2(n,1)=1$；$S2(n,n)=1$；$S2(n,k)=0(k>n)$。

第二类Stirling数在竞赛中较少出现，但在竞赛中也有一些题目与其类似，甚至更为复杂。读者不妨自己来试着建立其中的递推关系。

小结：通过上面对五种典型的递推关系建立过程的探讨，可知对待递推类的题目，要具体情况具体分析，通过找到某状态与其前面状态的联系，建立相应的递推关系。

【例7】(1998合肥市竞赛复试第二题)同一平面内的n(n<500)条直线，已知有p(p>=2)条直线相交于同一点，则这n条直线最多能将平面分割成多少个不同的区域？

解：这道题目与第一部分中的平面分割问题十分相似，不同之处就在于线条的曲直以及是否存在共点线条。由于共点直线的特殊性，我们决定先考虑p条相交于一点的直线，然后再考虑剩下的n-p条直线。首先可以直接求出p条相交于一点的直线将平面划分成的区域数为2p个，然后在平面上已经有k(k>=p)条直线的基础上，加上一条直线，最多可以与k条直线相交，而每次相交都会增加一个区域，与最后一条直线相交后，由于直线可以无限延伸，还会再增加一个区域。所以$f_m=f_{m-1}+m$ (m>p)，边界条件在前面已经计算过了，是$f_p=2p$。虽然这题看上去有两个参数，但是在实际做题中会发现，本题还是属于带一个参数的递推关系。

3、平面分割

同一平面内有n（n≤500）条直线，已知其中p（p≥2）条直线相交于同一点，则这n条直线最多能将平面分割成多少个不同的区域？

【输入格式】

两个整数n（n≤500）和p（2≤p≤n）。

【输出格式】

一个正整数，代表最多分割成的区域数目。

【输入样例】Surface.in

12 5

【输出样例】Surface.out

73

4、骨牌铺法

有1×n的一个长方形，用一个1×1、1×2和1×3的骨牌铺满方格。例如当n=3时为1×3的方格。此时用1×1、1×2和1×3的骨牌铺满方格，共有四种铺法。如下图：

与。

图 4.4.3

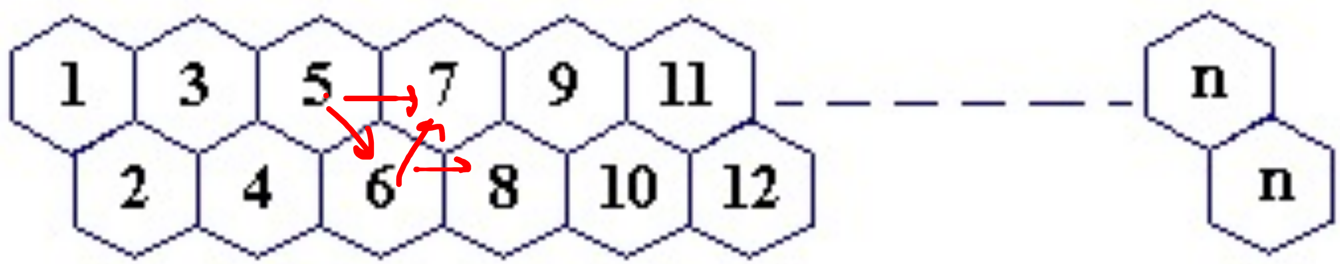
【输入样例】Domino.in
3

【输出样例】Domino.out
4

5、蜜蜂路线

【问题描述】

　　一只蜜蜂在下图所示的数字蜂房上爬动,已知它只能从标号小的蜂房爬到标号大的相邻蜂房,现在问你：蜜蜂从蜂房M开始爬到蜂房N，M<N，有多少种爬行路线？



【输入格式】

输入M，N的值。

【输出格式】

爬行有多少种路线。

【输入样例】**bee.in**

1 14

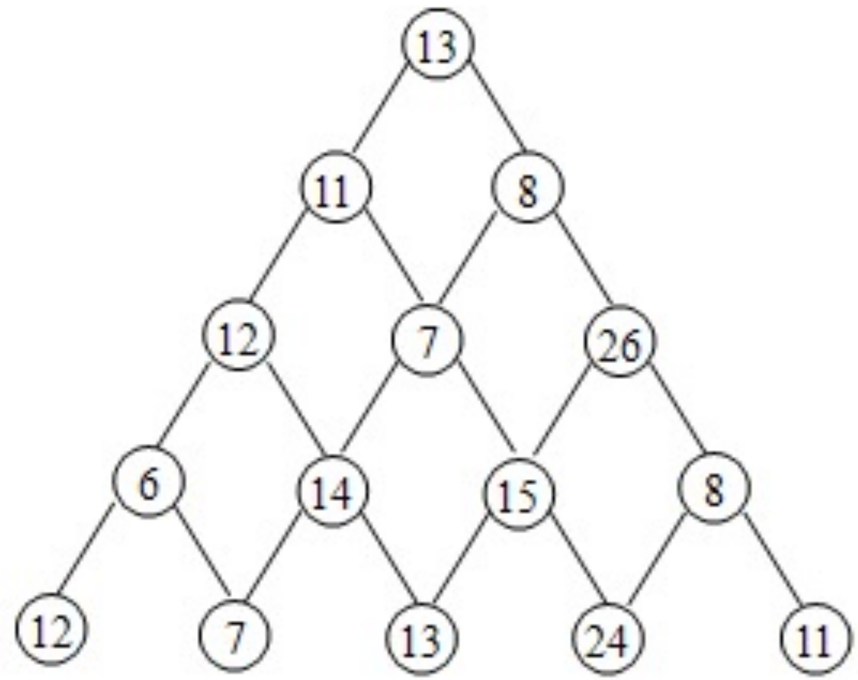
【输出样例】**bee.out**

377

6、数塔问题

【问题描述】

设有一个三角形的数塔，顶点为根结点，每个结点有一个整数值。从顶点出发，可以向左走或向右走，如图所示：



若要求从根结点开始，请找出一条路径，使路径之和最大，只要输出路径的和。

【输入格式】

第一行为n(n<10)，表示数塔的层数

从第2行至n+1行，每行有若干个数据，表示数塔中的数值。

【输出格式】

输出路径和最大的路径值。

【输入样例】tower.in

5
13
11 8
12 7 26
6 14 15 8
12 7 13 24 11

【输出样例】tower.out

86

9、极值问题

【问题描述】

已知m、n为整数，且满足下列两个条件：

① m、n∈{1， 2， …， k}，即1≤m，n≤k

②（n2－m*n－m2）2＝1

你的任务是：编程输入正整数k（1≤k≤109），求一组满足上述两个条件的m、n，并且使m2＋n2的值最大。例如，从键盘输入k=1995，则输出：m=987　n=1597。

【输入样例】**Acme.in**

1995

【输出样例】**Acme.out**

m=987

n=1597