一个小问题-走楼梯

问题：

一个小青蛙🐸

呱呱，呱呱呱

来到台阶前

一跳跳2阶

有5个台阶。假设小青蛙一次要么跳1阶，要么跳2阶。

问到达第五阶有多少钟跳法?

注意（1，2） 和（2，1）是两种不同的走法

（小青蛙所处平地在第0阶）

方法观察规律

在算时间时，
顺便算出了答案：

```
n=1   1    此程序的运行时间
n=2   2    此程序的运行时间
n=3   3    此程序的运行时间
n=4   5    此程序的运行时间
n=5   8    此程序的运行时间
n=6   13       此程序的运行
n=7   21       此程序的运行
n=8   34       此程序的运行
n=9   55       此程序的运行
n=10      89       此程序
n=11      144      此程序
n=12      233      此程序
n=13      377      此程序
n=14      610      此程序
n=15      987      此程序
n=16      1597     此程序
n=17      2584     此程序
n=18      4181     此程序
n=19      6765     此程序
n=20      10946    此程序
n=21      17711    此程序
n=22      28657    此程序
n=23      46368    此程序
n=24      75025    此程序
n=25      121393       此
n=26      196418       此
n=27      317811       此
n=28      514229       此
n=29      832040       此
n=30      1346269      此
n=31      2178309      此
n=32      3524578      此
n=33      5702887      此
n=34      9227465      此
n=35      14930352     此
n=36      24157817     此
n=37      39088169     此
n=38      63245986     此
n=39      102334155    此
n=40      165580141    此
n=41      267914296    此
n=42      433494437    此
n=43      701408733    此
n=44      1134903170
n=45      1836311903
n=46      2971215073
n=47      4807526976
n=48      7778742049
```

1 2 3 5 8 13 21....

找找规律?

观察数学规律

1 2 3 5 8 13 21....

规律

除前**两项**为 1 2外，其余项=相邻后2项之和

数学

$$f(x) = \begin{cases} 1 & n = 1 \\ 2 & n = 2 \\ f(x-1) + f(x-2) & n \geq 3 \end{cases}$$

信息学理解数学概念之

分段函数

描述不同数据输入的情况下函数的**答案**或**计算规则**的数学方法

观察数学规律

每一项可以通过前两项直接计算得出

伪代码

```
a[1]=1,a[2]=2;
FOR i=3...n:
    a[i]=a[i-1]+a[i-2];
```

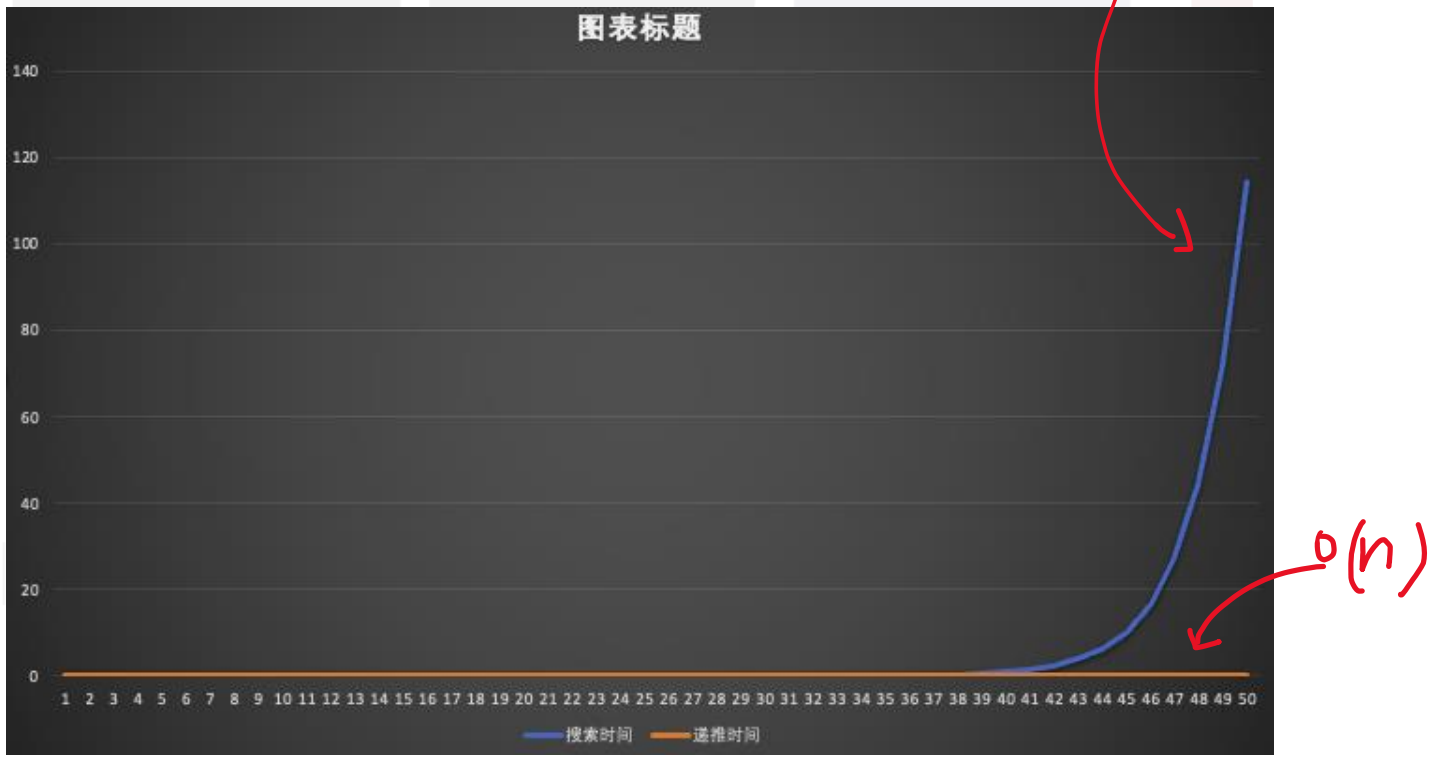
数组存储 → 初始化 → 循环刷表

代码耗时

N=50

0.00000200 s

递归方法：114s



根据已有信息推出

到达任意一节台阶的方案数。

规律：
要么从前1阶，走一步到
要么从前2阶，走两步到

到达n阶台阶的方法数
必然是前两阶方法数的和
(分类加法原理)

递推关系式

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

信息学

递推

西南大学附属中学校

信息奥赛教练组

递推过程：

递推关系+ 初始化 + 循环刷表 = 答案

递推关系：数据项间的关系（例如： $f(x)=\begin{cases}1 & n=1\\ 2 & n=2\\ f(x-1)+f(x-2) & n\geq 3\end{cases}$ **）**

递推方式

从已知条件出发逐步推到问题结果，此种方法叫**顺推**。
从问题出发逐步推到已知条件，此种方法叫**逆推**。

走楼梯变形1

- 还是走楼梯
- 只不过这次可以一次走1 or 2 or 3步

递推思路：

递推关系 f[n]=f[n-1]+ f[n-2]+ f[n-3]

初始化 f[1]=1，f[2]=2，f[3]=4

刷表 FOR i=4…n:
f[i]=f[i-1]+f[i-2]+f[i-3]

有炸弹的走楼梯

- 还是1/2步的楼梯
- 但是，特定楼梯上有炸弹（输入给定）不能走到炸弹上

递推思路：

递推关系

IF boom[n]==0:

f[n]=f[n-1]+ f[n-2]+ f[n-3]

ELSE

f[n]=0

N号位上没有炸弹

可以赋值为其他值表示不可能走到么？
不能-1，必须0的原因
1 没法走到n，方法数为0
2 不影响后续计算

分析

- 还是1/2步的楼梯
- 但是，特定楼梯上有炸弹（输入给定）不能走到炸弹上

递推思路：

初始化 f[1]=1，f[2]=2，f[3]=4

分析

- 还是1/2步的楼梯
- 但是，特定楼梯上有炸弹（输入给定）不能走到炸弹上

递推思路：

思维完备性！

初始化 f[1]=1，f[2]=2，f[3]=4

IF boom[1]==1: f[1]=0

IF boom[2]==1: f[2]=0

IF boom[3]==1: f[3]=0

例题：昆虫繁殖

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2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）

a[i]表示第i个月成虫
b[i]表示第i个月卵

第x个月的成虫= 第x-y个月出生卵+第x-1个月成虫

递推关系

a[i]=a[i-1]+b[i-2] //2个月前的卵成虫

b[i]=a[i-x]*y//每对虫生产y对卵

分析

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2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）

初始条件 显然前x个月 a[ ]=1 ,所以循环从x+1 开始到z+1

递推关系

a[i]=a[i-1]+b[i-2] //2个月前的卵成虫

b[i]=a[i-x]*y//每对虫生产y对卵

代码


```
long long a[10010];
long long b[10010];
int main()
{
    int x, y, z;
    cin >> x >> y >> z;
    for (int i = 1; i <= x; i++) {
        a[i] = 1;
        b[i] = 0;
    }
    for (int i = x + 1; i <= z + 1; i++) {
        a[i] = a[i - 1] + b[i - 2];
        b[i] = a[i - x] * y;
    }
    cout << a[z + 1];
    return 0;
}
```

递推的解题思路

从答案入手：思考需要什么信息获得这个答案。此为逆推。

昆虫繁殖：根据题目信息确定递推公式（数据流动方向）

走楼梯：走到第n阶只能从n-1和n-2出发。

从数据入手：思考如何逐步计算，获得答案。此为顺推。

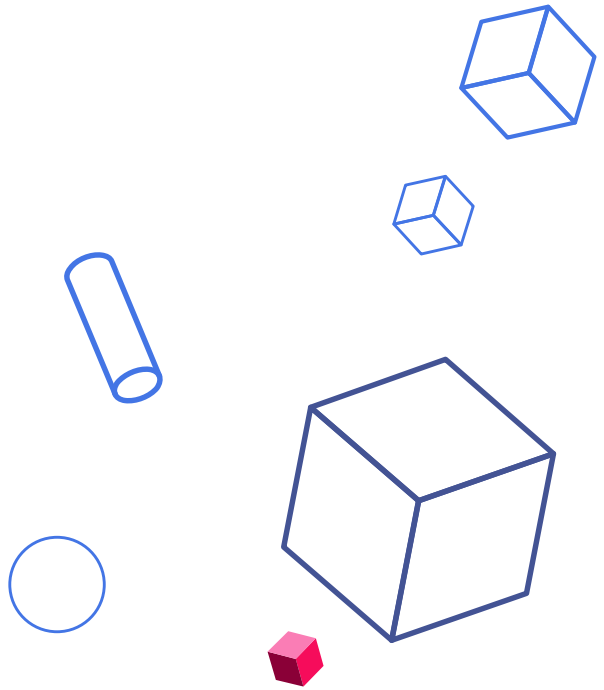
兔子繁殖：暴力计算出整体数据，找规律（发现和走楼梯同一类型）

递推与递归的区别

递推：从初值出发反复进行某一运算得到所需结果。从已知到未知，从小到大（比如每年长高9cm，20年180，30后270）

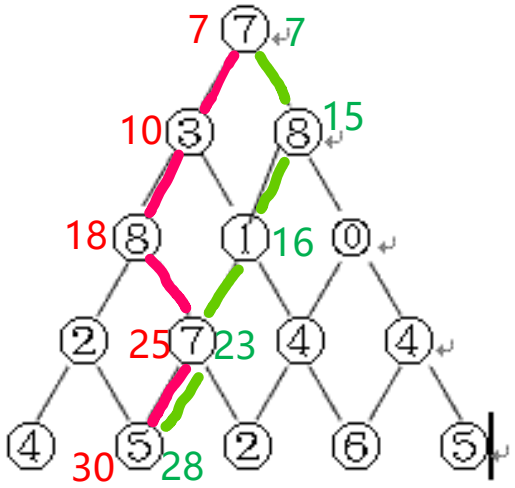
递归：从所需结果出发不断回溯前一运算直到回到初值再递推得到所需结果。从未知到已知，从大到小，再从小到大

二维递推



分析

如图所示，一个数字三角形。
请编一个程序计算从顶至底的某处的一条路径，使该路径所经过的数字的**总和最大。**
每一步可沿左斜线向下或右斜线向下走



绿色：按照贪心（当前最优）的思路选取的数
红色：正确答案

寻找其他方法解决问题

从解决**最简单的小问题**入手

什么样的问题算小问题？

分析

如图所示，一个数字三角形。
请编一个程序计算从顶至底的某处的一条路径，使该路径所经过的数字的**总和最大**。
每一步可沿左斜线向下或右斜线向下走

递推的思维模式：
小问题出发，推大问题
一层层推导问题的解决方法

1层

最大值：7

红色数字表示
走到此处时
数字和的最大值

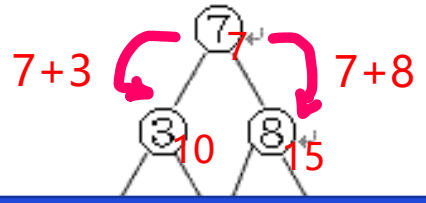
分析

如图所示，一个数字三角形。
请编一个程序计算从顶至底的某处的一条路径，使该路径所经过的数字的**总和最大**。
每一步可沿左斜线向下或右斜线向下走

递推的思维模式：
小问题出发，推大问题
一层层推导问题的解决方法

1层

2层



最大值：7

最大值：15 如何求第二层经过数字的最大值?

红色数字表示
走到此处时
数字和的最大值

分析

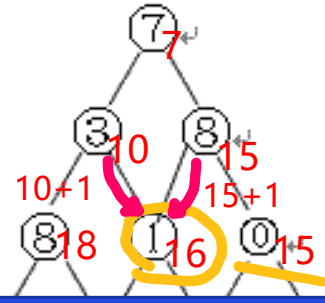
如图所示，一个数字三角形。
请编一个程序计算从顶至底的某处的一条路径，使该路径所经过的数字的**总和最大**。
每一步可沿左斜线向下或右斜线向下走

递推的思维模式：
小问题出发，推大问题
一层层推导问题的解决方法

1层

2层

3层



最大值：7

最大值：15

最大值：18

因为题目要求最大和，
所以会有一个选择过程
max(1+10 , 1+15)

红色数字表示
走到此处时
数字和的最大值

分析

如图所示，一个数字三角形。
请编一个程序计算从顶至底的某处的一条路径，使该路径所经过的数字的**总和最大**。
每一步可沿左斜线向下或右斜线向下走

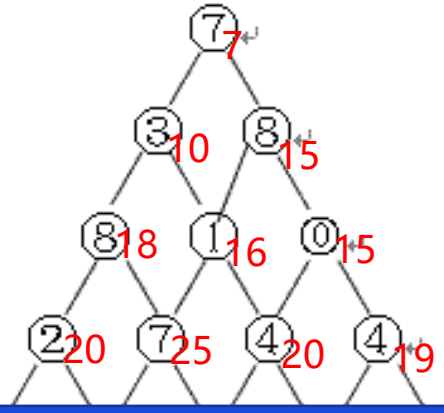
递推的思维模式：
小问题出发，推大问题
一层层推导问题的解决方法

1层

2层

3层

4层



最大值：7

最大值：15

最大值：18

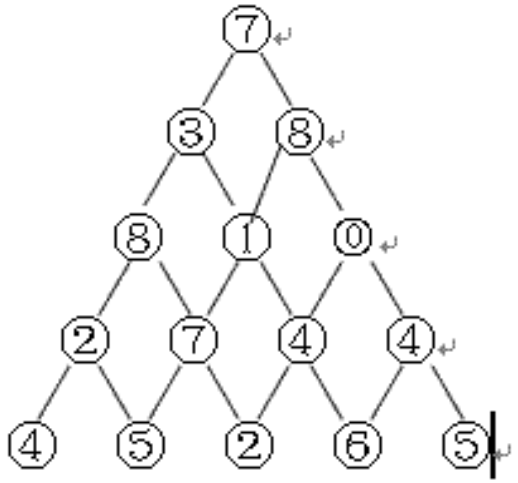
最大值：25

红色数字表示
走到此处时
数字和的最大值

分析

如图所示，一个数字三角形。
请编一个程序计算从顶至底的某处的一条路径，使该路径所经过的数字的总和最大。
每一步可沿左斜线向下或右斜线向下走

1层
2层
3层
4层
5层



记录数字三角形中的数

7
10 15
18 16 15
20 25 20 19
24 30 27 26 24

记录一层层的最值

存储

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

基础数据存储

7				
10	15			
18	16	15		
20	25	20	19	
24	30	27	26	24

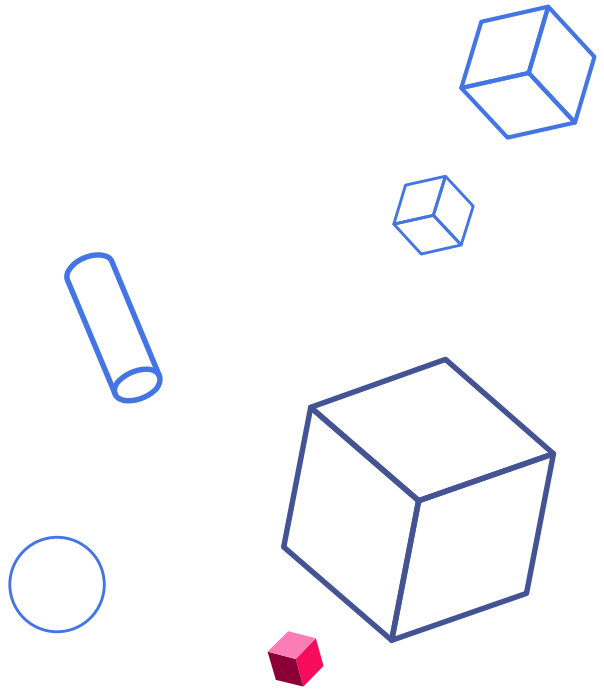
中间过程存储

代码


```
#include <bits/stdc++.h>
using namespace std;
int n, arr[30][30], f[30][30];
int main(){
    //读入
    cin >> n;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= i; j++) {
            scanf("%d", &arr[i][j]);//cin>>arr[i][j];
        }
    }

    //边界条件赋值
    f[1][1] = arr[1][1];
    for (int i = 1; i <= n; i++) {
        f[i][1] = f[i - 1][1] + arr[i][1];
        f[i][i] = f[i - 1][i - 1] + arr[i][i];
    }
    //递推
    for (int i = 2; i <= n; i++) {
        for (int j = 1; j <= i; j++) {
            f[i][j] = max( f[i - 1][j - 1], f[i - 1][j] ) +
arr[i][j];
                // 决定信息流动方向
                // 是从左上下来还是从上方下来
        }
    }
    //打擂台
    int ans = f[n][1];
    for (int i = 2; i <= n; i++) {
        ans = max( ans, f[n][i] );
    }
    cout << ans;
    return 0;
}
```

FZOI
西大附中信息学竞赛
High School Affiliated to Southwest University

西南大学附属中学
High School Affiliated to Southwest University



递推经典模型

斐波那契数列

$$f(x) = \begin{cases} 1 & n = 1 \\ 2 & n = 2 \\ f(x-1) + f(x-2) & n \geq 3 \end{cases}$$

来自意大利数学家：
斐波那契的《算盘书》
1202年

L.Fibonacci, 1170-1250

爬楼梯
兔子繁殖
骨牌问题

抽象总结

递推公式形如

$f(x) = k_1*f(x-a_1) + k_2*f(x-a_2) \ldots$

是斐波那契数列类问题

第二类斯特林数 Stirling

将n个不同小球放入m个相同盒子并没有空盒的方案数
可用$S_2(n,m)$表示

把1～n个球放入m个盒子中，
假设前n-1个球已经妥善放置，考虑第n个球的放法：
1 要么n单独一盒: $S_2(n-1,m-1)$
2 要么n和其他球一起: $m*S_1(n-1,m)$

递推公式： **$S_2(n,m) = S_2(n-1,m-1) + m*S_1(n-1,m)$**

第二类斯特林数 Stirling

将n个不同小球放入m个相同盒子并没有空盒的方案数
可用$S_2(n,m)$表示

递推公式： $S_2(n,m) = S_2(n-1,m-1) + m*S_2(n-1,m)$
边界条件?

IF k==t ： $S_2(k,t) = 1$
IF k<t ： $S_2(k,t) = 0$ //不允许空盒

代码

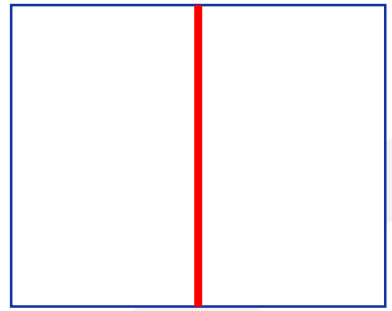

```
int main(){
    int n,m;
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        a[i][i]=1;
    }
    for(int i=1;i<=n;i++){
        for(int j=1;j<i;j++){//注意j范围从1到i
            if(j>i)break;
            a[i][j]=a[i-1][j-1] + j*a[i-1][j];
        }

    }
    cout<<a[n][m];
    return 0;
}
```

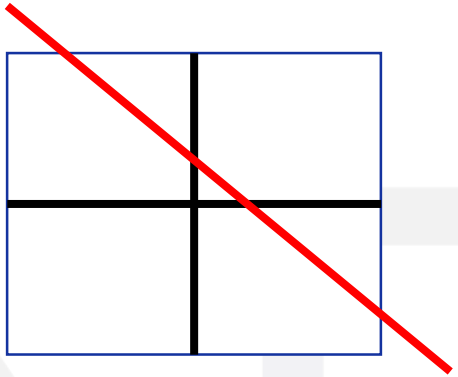
第二类斯特林数变形

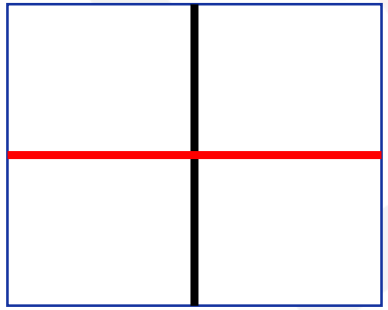
- n 个不同的球，放入 m 个无区别的盒子，不允许盒子为空。
- n 个不同的球，放入 m 个有区别的盒子，不允许盒子为空。
- n 个不同的球，放入 m 个无区别的盒子，允许盒子为空。
- n 个不同的球，放入 m 个有区别的盒子，允许盒子为空。

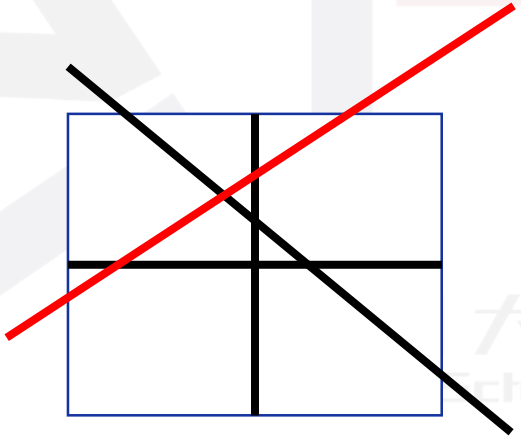
https://blog.csdn.net/u011815404/article/details/80083954

平面上有n个线段，最多能把平面分成几部分？









不难推出规律：
增加第n条线段，就会增加n个平面

递推式：f[n]=f[n-1]+n； n>1

f[1]=2;

平面上有n个圆，最多能把平面分成几部分？
(任意2个圆之间最多有2个交点)

从小例子出发，到大样本数据。

每多增加一个圆
则与原有圆最多产生2(n-1)条线段，就会组成2(n-1)个平面。
2(n-1)+原有平面数=构成一个新封闭图形数

递推公式：**a[n] =2(n-1) + a[n-1]**
初始化：a[1]=2
答案：a[n]

代码


```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;


ll a[1000];
int main(){
    int n;
    cin>>n;
    a[1]=2;
    for(int i=2;i<=n;i++){
        a[i]=a[i-1]+2*(i-1);
    }
    cout<<a[n];
    return 0;
}
```

平面上有n个椭圆，最多能把平面分成几部分?

分析：
每多增加一个椭圆
则与原有圆最多产生4(n-1)条线段，就会组成4(n-1)个平面。
4(n-1)+原有平面数=构成一个新封闭图形数

递推公式：**a[n] =4(n-1) + a[n-1]**
初始化：a[1]=2
答案：a[n]

注意3个基础操作：

1. 递推公式
2. 初始化
3. 循环计算（得到答案）

注意2个分析过程

1. 顺推：从小例子出发找规律
2. 逆推：分析答案如何计算得出

Thanks

For Your Watching