第2章 递归与分治策略

- 本章主要知识点：
  - 2.1 递归的概念
  - 2.2 分治法的基本思想
  - 2.3 二分搜索技术
  - 2.4 大整数的乘法
  - 2.5 Strassen矩阵乘法
  - 2.6 棋盘覆盖
  - 2.7 合并排序
  - 2.8 快速排序
  - 2.9 线性时间选择
  - 2.10 最接近点对问题
  - 2.11 循环赛日程表
- 计划授课时间：6~8课时

2.1 递归的概念

➢ 直接或间接地调用自身的算法称为**递归算法**。用函数自身给出定义的函数称为**递归函数**。

一个递归问题

调用自己

- 从前有座山，山上有座庙，庙里有个老和尚讲故事，讲的是
  - 从前有座山，山上有座庙，庙里有个老和尚讲故事，讲的是
    - 从前有座山，山上有座庙，庙里有个老和尚讲故事，讲的是

…… ……

2.1 递归的概念---阶乘函数

- 例1 阶乘函数
- 可递归地定义为：

$$n! = \begin{cases} 1 & n = 0 \\ n(n-1)! & n > 0 \end{cases}$$

- 其中：
  - $n=0$时，$n!=1$为边界条件
  - $n>0$时，$n!=n(n-1)!$为递归方程
- 边界条件与递归方程是递归函数的二个要素，递归函数只有具备了这两个要素，才能在有限次计算后得出结果。

任何大于1的自然数n阶乘表示方法：

n!=1×2×3×……×n

2.1 递归的概念---阶乘函数

$$n! = \begin{cases} 1 & n = 0 \\ n(n-1)! & n > 0 \end{cases}$$

```
int Factorial(int n)
{
   if (n==0) return 1;
   return n*Factorial(n - 1);
}
```

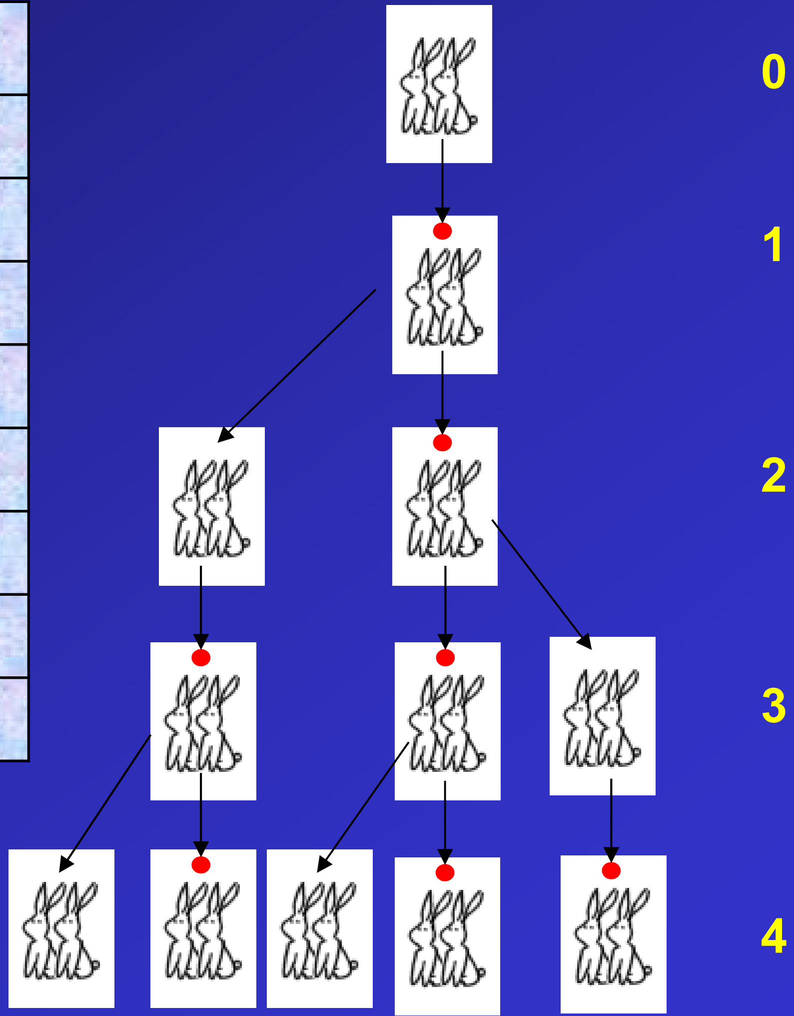
2.1 递归的概念--*Fibonacci*数列

- 例2: *Fibonacci*数列
  - 问题引入
    - 裴波那契（*Fibonacci leonardo*，约1170-1250）是意大利著名数学家.
    - 在他的著作《算盘书》中许多有趣的问题，最富成功的问题是著名的“**兔子繁殖问题**”：如果每对兔子每月繁殖一对子兔，而子兔在出生后第二个月就有生殖能力，试问一对兔子一年能繁殖多少对兔子？
  - 问题分析

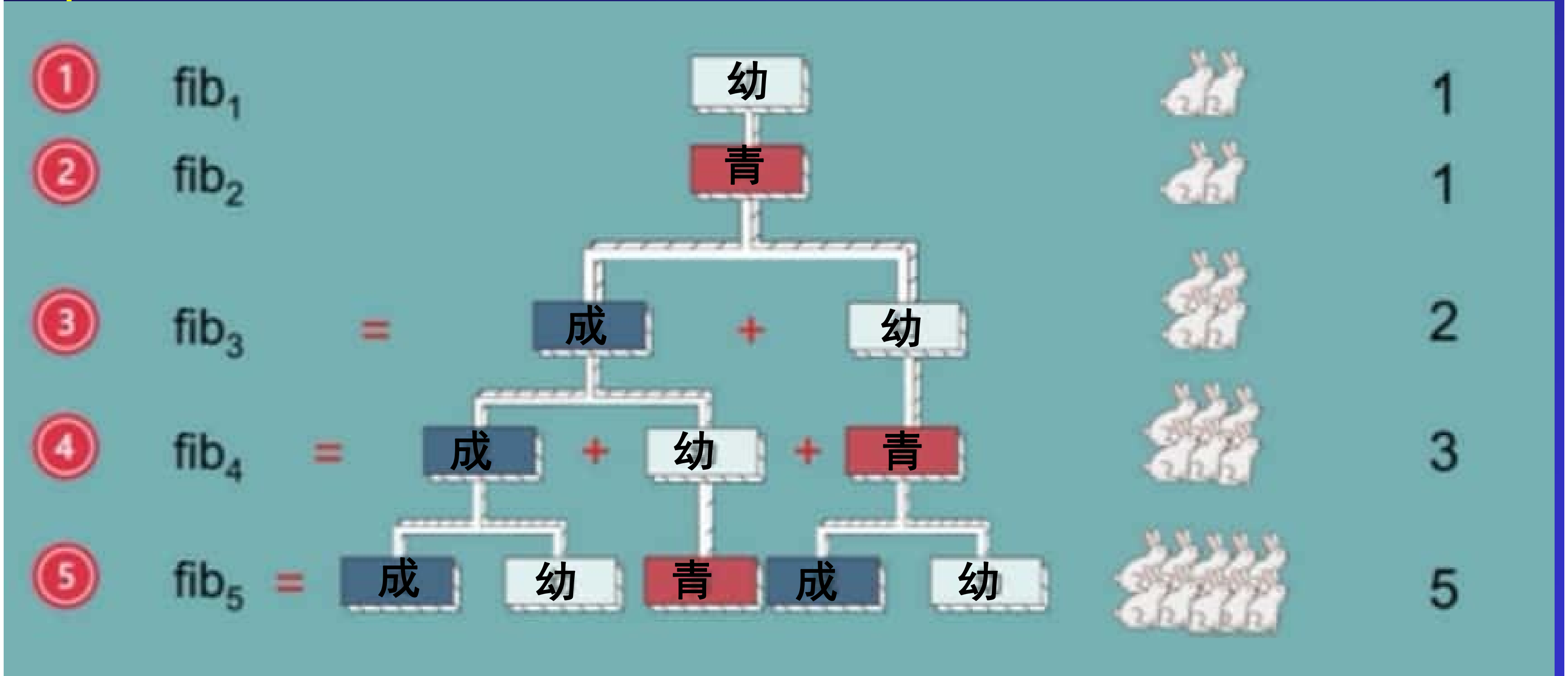
2.1 递归的概念--*Fibonacci*数列

月份	初生	成熟	总数
0	1	0	1
1	0	1	1
2	1	1	2
3	1	2	3
4	2	3	5
5	3	5	8
6	5	8	13
……	……	……	……

$$F(n)=\begin{cases}1 & n=0\\ 1 & n=1\\ F(n-1)+F(n-2) & n>1\end{cases}$$



2.1 递归的概念--*Fibonacci*数列



月份	初生	成熟	总数
0	1	0	1
1	0	1	1
2	1	1	2
3	1	2	3
4	2	3	5
5	3	5	8
6	5	8	13
……	……	……	……

$$F(n)=\begin{cases}1 & n=0\\ 1 & n=1\\ F(n-1)+F(n-2) & n>1\end{cases}$$

2.1 递归的概念

- 数列的特点
  - 数列的增长速度
  - 构造一个新数列
  - 自然科学中的若干实例

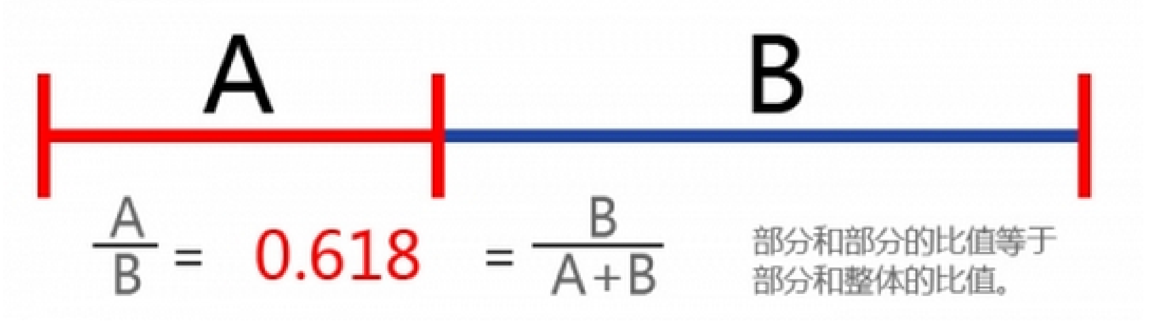
$$\frac{1}{1},\frac{1}{2},\frac{2}{3},\frac{3}{5},\frac{5}{8},\frac{8}{13},\frac{13}{21},\frac{21}{34}\cdots\cdots\frac{F_n}{F_{n+1}}\cdots\cdots$$

$$\lim_{n\to\infty}\frac{f(n)}{f(n+1)}=\frac{\sqrt{5}-1}{2}=0.618\ldots(\text{黄金分割数})$$

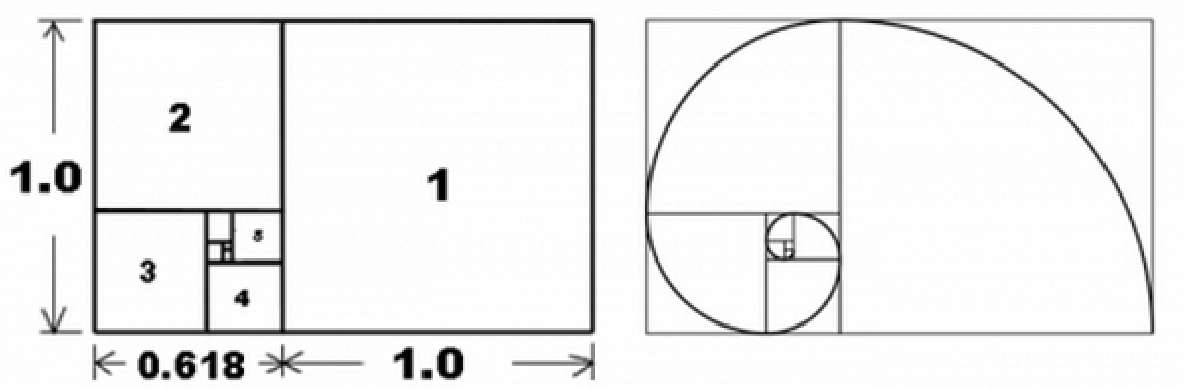
PS：黄金分割的美学价值

➢ 黄金分割率：

0.618或者1.618，这个数字是否觉得似曾相识。这其实是一个数学比例关系（说到数学，不要先着急晕哦，知道咱们做设计得对计算都不敏感，呵呵），即把一条线段分为两部分，此时短段与长段之比恰恰等于长段与整条线之比，其数值比为1:1.618或0.618:1。

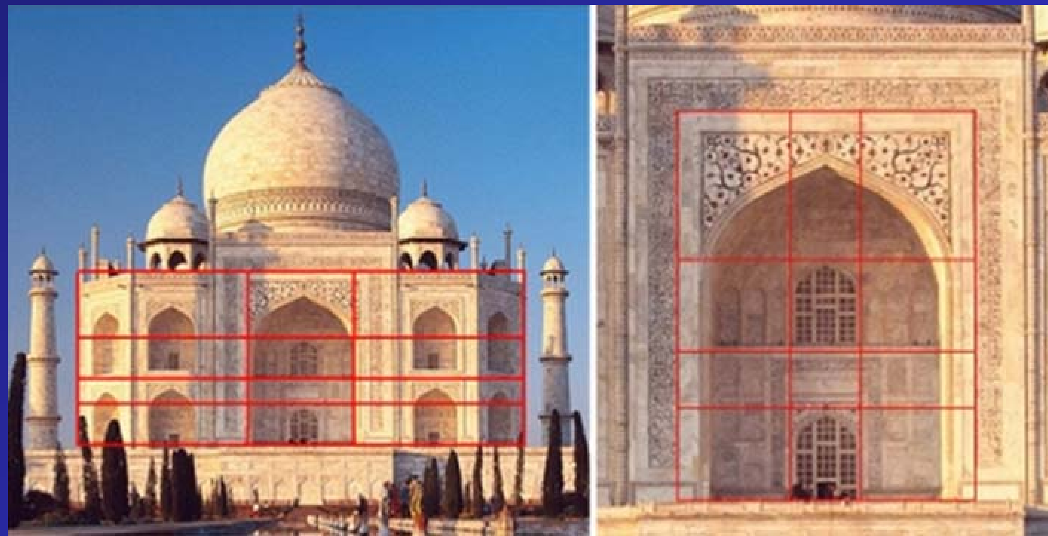


➢ 黄金矩形：长宽之比为黄金分割率，即矩形的长边为短边 1.618倍



PS：黄金分割的美学价值

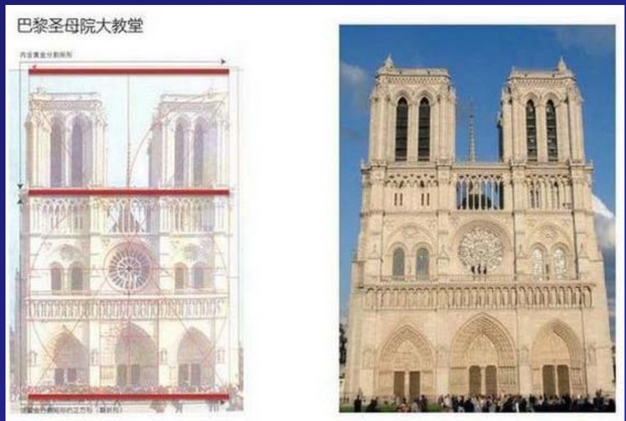
- 印度的泰姬陵：

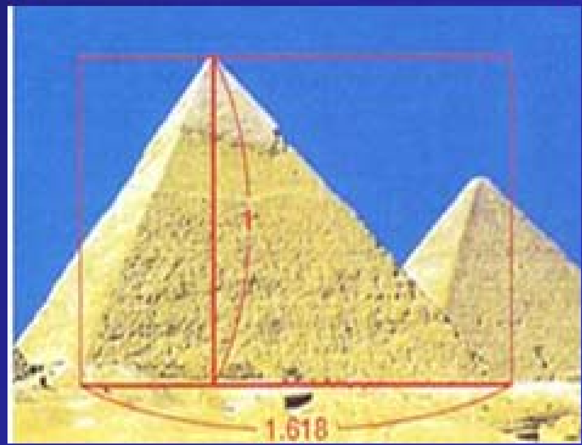


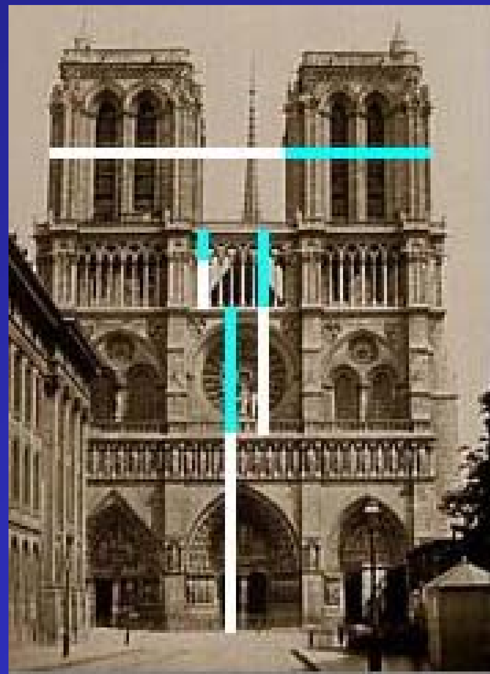
- 希腊雅典的巴特农神庙：

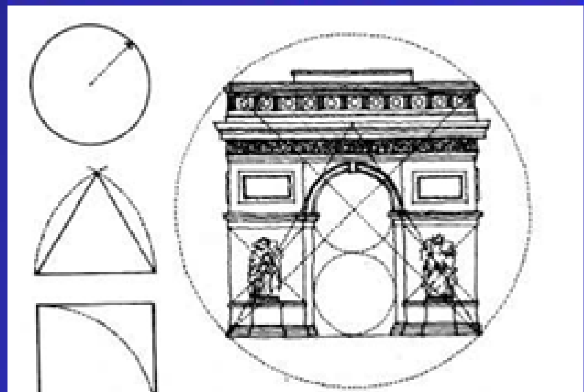
PS：黄金分割的美学价值

- 古埃及金字塔
- 巴黎圣母院
- 法国埃菲尔铁塔

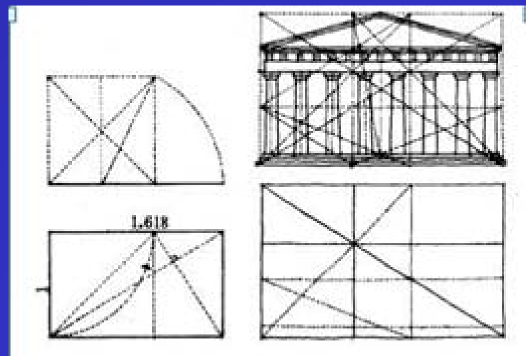








a 对巴黎星形广场凯旋门所作的几何分析

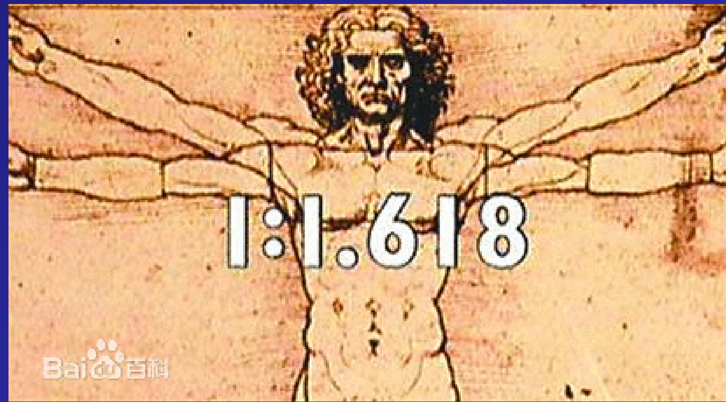


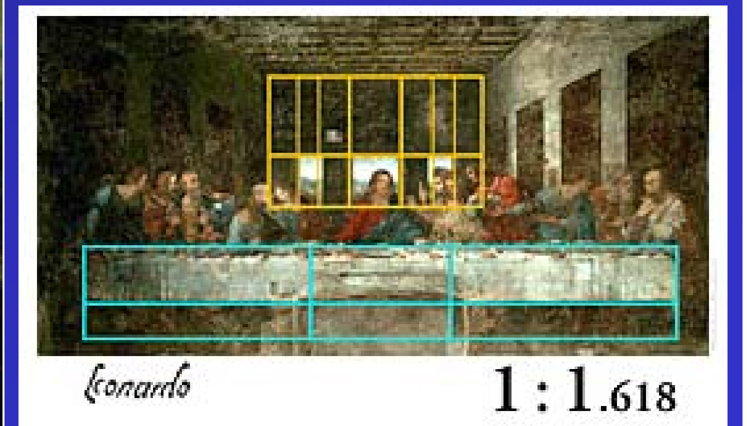
b 对帕提农神庙所作的几何分析

- 大多数门窗的宽长之比也是0.618；

PS：黄金分割的美学价值

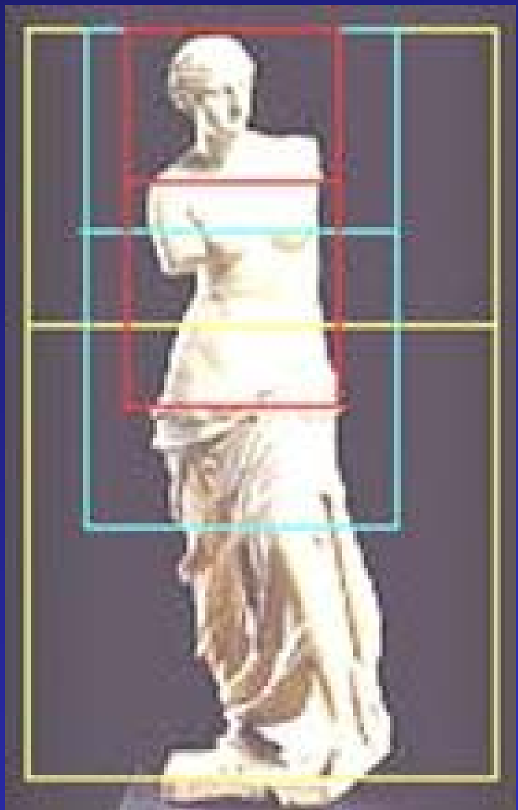
- 《维特鲁威人》
- 《蒙娜丽莎》
- 《最后的晚餐》

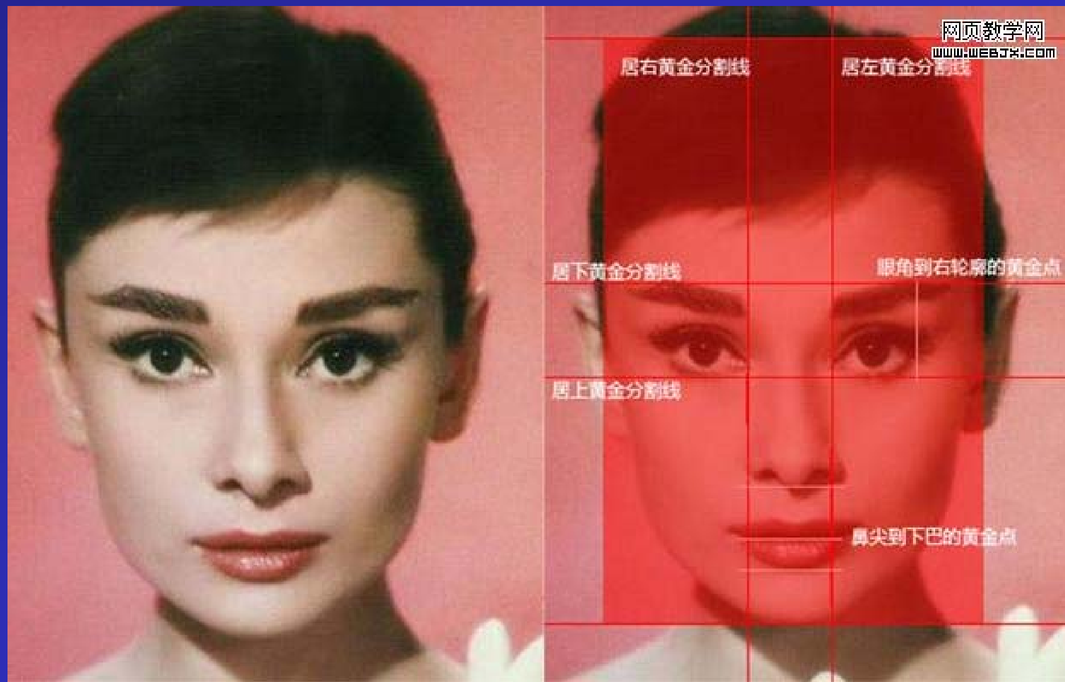




PS：黄金分割的美学价值

- 古希腊维纳斯女神塑像及太阳神阿波罗的形象都通过故意延长双腿，使之与身高的比值为0.618，从而创造艺术美。

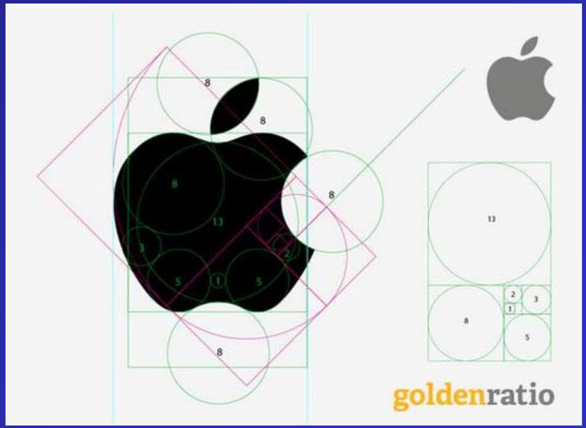




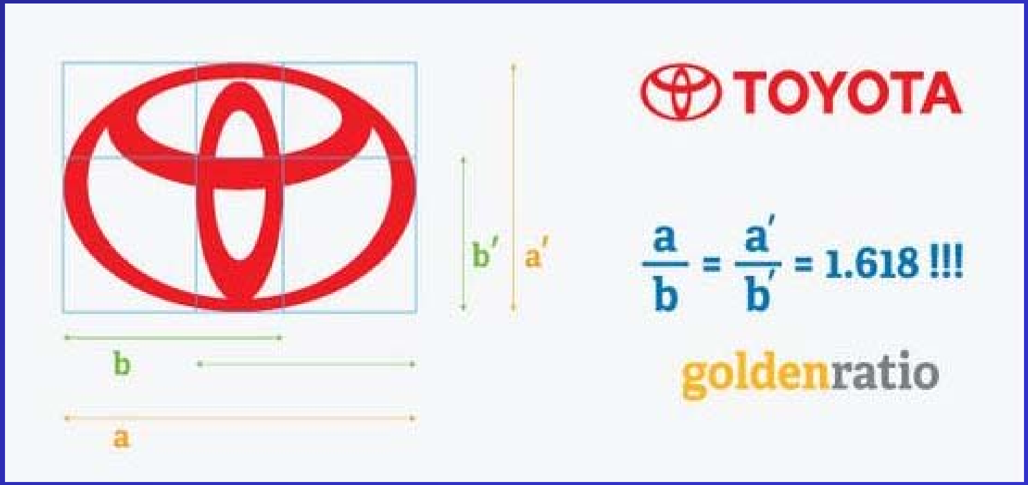
奥黛丽赫本

PS：黄金分割的美学价值

- **Apple** Apple的Logo具有完美的平衡，映射到Logo上的轮廓是在直径上遵循斐波那契数列的圆形。



- **丰田** Toyota的Logo由三个椭圆组成。一个基于φ的网格。该网格的网格线间隔遵循黄金分割率φ



2.1 递归的概念--*Fibonacci*数列

- 定义及解法

$$F(n)=\begin{cases}0 & n=0\\ 1 & n=1\\ F(n-1)+F(n-2) & n>1\end{cases}$$

$$F(n)=\frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n+1}-\left(\frac{1-\sqrt{5}}{2}\right)^{n+1}\right)$$

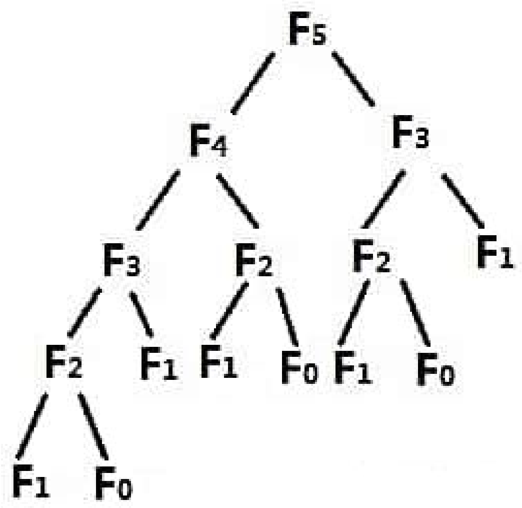
$$\begin{vmatrix}F_{n+1} & F_n\\ F_n & F_{n-1}\end{vmatrix}=\begin{vmatrix}1 & 1\\ 1 & 0\end{vmatrix}^n$$

2.1 递归的概念--*Fibonacci*数列

$$F(n)=\begin{cases} 0 & n=0 \\ 1 & n=1 \\ F(n-1)+F(n-2) & n>1 \end{cases}$$

```
long long Fib(int n)//递归
{
        if (n < 2)
        {
                return n;
        }
        return Fib(n - 1) + Fib(n - 2);
}
```

我们把 F_5 作为树的根节点， F_4 和 F_3 作为左右两个叶子节点，继续向下递归，左节点 F_4 继续向下分解为 F_3 和 F_2 ，右节点 F_3 继续向下分解为 F_2 和 F_1 ，依此类推，如下图所示：



因此，该算法的时间复杂度为 $O(2^n)$。

2.1 递归的概念--*Fibonacci*数列

➤非递归解法：

```
int Fibonacci(int n) {
        if (n<=0) {
            return 0;
        }
        if (n==1) {
            return 1;
        }
        int min=0;
        int max=1;
        int i=2;
        int result=0;
        while (i<=n) {
            result=min+max;
            min=max;
            max=result;
            ++i;
        }
        return result;
    }
```

如果说前面的递归解法是自顶向下将大问题拆解成小问题求解，那么循环解法则是逆向思维，自底向上，先求出小问题的解，再向上一步一步向上求取最终问题的解。

单层循环，时间复杂度为$O(n)$

2.1 递归的概念--*Fibonacci*数列

➢矩阵连乘法：

根据斐波那契数列自身的性质，我们可以构造如下方等式关系：

$$\begin{cases} F_2 = F_1 + F_0 \\ F_1 = F_1 \end{cases}$$

使用矩阵表示上述等式关系，即

$$\begin{bmatrix} F_2 \\ F_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} F_1 \\ F_0 \end{bmatrix}$$

那么

$$\begin{bmatrix} F_3 \\ F_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} F_2 \\ F_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^2 \times \begin{bmatrix} F_1 \\ F_0 \end{bmatrix}$$

依次乘下去，可得一般形式

$$\begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \times \begin{bmatrix} F_1 \\ F_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \times \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

现在求$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$的特征值。

使

$$\mathbf{Det}\left(\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)$$

$$= \lambda^2 - \lambda - 1 = 0$$

解出$\lambda_1 = \frac{1+\sqrt{5}}{2}$，$\lambda_2 = \frac{1-\sqrt{5}}{2}$

所以矩阵$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$的特征向量为

$$\vec{x}_1 = \begin{bmatrix} \frac{1+\sqrt{5}}{2} \\ 1 \end{bmatrix}$$

$$\vec{x}_2 = \begin{bmatrix} \frac{1-\sqrt{5}}{2} \\ 1 \end{bmatrix}$$

线性无关

两个特征值为 λ_1=1.618，λ_2=-0.618

→ $O(1.618^n)$

2.1 递归的概念--*Fibonacci*数列

➢矩阵连乘法：

根据斐波那契数列自身的性质，我们可以构造如下方等式关系：

$$\begin{cases} F_2 = F_1 + F_0 \\ F_1 = F_1 \end{cases}$$

使用矩阵表示上述等式关系，即

$$\begin{bmatrix} F_2 \\ F_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} F_1 \\ F_0 \end{bmatrix}$$

那么

$$\begin{bmatrix} F_3 \\ F_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} F_2 \\ F_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^2 \times \begin{bmatrix} F_1 \\ F_0 \end{bmatrix}$$

依次乘下去，可得一般形式

$$\begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \times \begin{bmatrix} F_1 \\ F_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \times \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

现在寻找方法使$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n = T\Lambda^n T^{-1}$

这里Λ为对角阵

由相似对角化原理知

$$T = \begin{bmatrix} \frac{1+\sqrt{5}}{2} & \frac{1-\sqrt{5}}{2} \\ 1 & 1 \end{bmatrix}$$

$$T^{-1} = \begin{bmatrix} \frac{1}{\sqrt{5}} & \frac{-1+\sqrt{5}}{2\sqrt{5}} \\ -\frac{1}{\sqrt{5}} & \frac{1+\sqrt{5}}{2\sqrt{5}} \end{bmatrix}$$

此时$T^{-1}AT = \Lambda = \begin{bmatrix} \frac{1+\sqrt{5}}{2} & 0 \\ 0 & \frac{1-\sqrt{5}}{2} \end{bmatrix}$

从而 $\begin{bmatrix} a_{n+2} \\ a_{n+1} \end{bmatrix} = T\Lambda^n T^{-1} \begin{bmatrix} a_2 \\ a_1 \end{bmatrix}$

$$= \begin{bmatrix} \frac{1+\sqrt{5}}{2} & \frac{1-\sqrt{5}}{2} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \frac{1+\sqrt{5}}{2} & 0 \\ 0 & \frac{1-\sqrt{5}}{2} \end{bmatrix}^n \begin{bmatrix} \frac{1}{\sqrt{5}} & \frac{-1+\sqrt{5}}{2\sqrt{5}} \\ -\frac{1}{\sqrt{5}} & \frac{1+\sqrt{5}}{2\sqrt{5}} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

换元，取一个行得到

$$a_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

$n \in N_+$

这就是斐波那契数列的通项公式。

我们也可以发现一个全是整数的数列通项公式竟然有无理数。
而且和黄金分割率有关。

$\rightarrow$ O(logn)

2.1 递归的概念--*Fibonacci*数列

- 三种解法的比较
  - 解法1： $O(1.618^n)$
  - 解法2： $O(n)$
  - 解法3： $O(logn)$

fib(110)：

$O(1.618^n)$	→	10^{22} 次运算
$O(n)$	→	111 次运算
$O(logn)$	→	7 次运算

2.1 递归的概念

- 例2 Fibonacci数列
- 无穷数列1，1，2，3，5，8，13，21，34，55，…，被称为Fibonacci数列。它可以递归地定义为：

$$F(n)=\begin{cases} 1 & n=0 \\ 1 & n=1 \\ F(n-1)+F(n-2) & n>1 \end{cases}$$

- 第n个Fibonacci数可递归地计算如下：

```
public static int fibonacci(int n)
  {
     if (n <= 1) return 1;
     return fibonacci(n-1)+fibonacci(n-2);
  }
```

2.1 递归的概念--*Fibonacci*数列

- 思考：
  - 楼梯问题
    - 有一楼梯共有n阶，上楼可以一步上一阶，也可以一步上两阶。
    - 编一个程序，计算共有多少种不同的走法？

$$S(n)=\begin{cases}1 & n=1\\ 2 & n=2\\ S(n-1)+S(n-2) & n>2\end{cases}$$

➢思考：

- 数字转换字符串问题 百度面试真题
  - 将给定的数转换为字符串，原则如下：1对应 a，2对应b，…..26对应z，例如12258可以转换为"abbeh", "aveh", "abyh", "lbeh" and "lyh"，个数为5，编写一个函数，给出可以转换的不同字符串的个数。

```
Please input the serial numbers:
1234

Total number of strings: 3
abcd
lcd
awd
```

```
Please input the serial numbers:
11020

Total number of strings: 6
aab
kb
ajb
aat
kt
ajt
```

```
Please input the serial numbers:
12235

Total number of strings: 5
abbce
lbce
avce
abwe
lwe
```

➢每个位置两种选择：

▪ 数字转换字符串问题

百度面试真题

1. 将当前单位数字翻译
2. 当前和下一位数字集合成两位数翻译（两位数需要小于26）

➢递归结束条件：

1. *最后一个数字时：只能以单个字符翻译，结束，返回1；*
2. *对于其他情况：起码都还剩余两个数字，所以我们先判断两位数字是不是满足[10,25]，满足的话说明有两种选择，返回f(i+1) + f(i+2)，否则只有一种选择，返回f(i+1);*
3. *特殊情况：当i为倒数第二个数时，即下标为n-2，此时如果最后两位数字构成的两位数满足要求（[10,25]），则有两种选择：分别是两个数字单独翻译以及两个数字合成一个翻译，应该返回2，即返回f(i+1) + f(i+2)，这时f(i+1)为1，而i+2超出n的范围了，应该返回1。递归结束条件进行修改，i>=n-1退出递归，返回1。*

2.1 递归的概念----*Ackerman*函数

- 例3 Ackerman函数
- 当一个函数及它的一个变量是由函数自身定义时，称这个函数是**双递归函数**。Ackerman函数A(n，m)定义如下：
- 前2例中的函数都可以找到相应的非递归方式定义。
- 但本例中的Ackerman函数却无法找到非递归的定义。

$$
\begin{cases}
A(1,0)=2 & \\
A(0,m)=1 & m\geq 0 \\
A(n,0)=n+2 & n\geq 2 \\
A(n,m)=A(A(n-1,m),m-1) & n,m\geq 1
\end{cases}
$$

$$n!=1\cdot 2\cdot 3\cdot \Lambda \cdot (n-1)\cdot n$$

$$F(n)=\frac{1}{\sqrt{5}}\left[\left[\frac{1+\sqrt{5}}{2}\right]^{n+1}-\left[\frac{1-\sqrt{5}}{2}\right]^{n+1}\right]$$

2.1 递归的概念----*Ackerman*函数

- *Ackerman*函数
  - $A(n,0) = n+2$
  - $A(n,1) = 2n$
  - $A(n,2) = 2^n$。
  - $A(n,3) = 2^{2^{2^{\cdot^{\cdot^{2}}}}}$（$n$ 个 2）

$$A(n,m)=\begin{cases} 2 & n=1, m=0 \\ 1 & n=0, m\geq 0 \\ n+2 & n\geq 2, m=0 \\ A(A(n-1,m),m-1) & n,m\geq 1 \end{cases}$$

  - $A(n,4)$的增长速度非常快，以致于没有适当的数学式子来表示这一函数。

$$A(3,4) = 2^{2^{2^{\cdot^{\cdot^{2}}}}}\quad(65536 \text{ 个 } 2)$$

2.1 递归的概念----*Ackerman*函数

```
int Ackerman(int n,int m)
{
  if( n= =1&&m= =0 )
        return 2;
  else if (n= =0&&m>=0)
           return 1;
        else if(n>=2&&m= =0)
                 return n + 2;
             else if(n>=1&&m>=1)
                      return Ackerman(Ackerman(n - 1,m),m - 1);
}
```

$$A(n,m)=\begin{cases}2 & n=1,m=0\\ 1 & n=0,m\geq 0\\ n+2 & n\geq 2,m=0\\ A(A(n-1,m),m-1) & n,m\geq 1\end{cases}$$

应用：路径压缩算法中，在集合的查找过程中将树的深度降低。

2.1 递归的概念----排列问题

- 设A={$a_1,a_2,\ldots,a_n$}是要进行排列的n个元素的集合，
- n=1　　输出a_1
- n=2　　输出a_1a_2

　　　　　　　a_2a_1
- n=3　　输出$a_3a_1a_2$

　　　　　　　$a_3a_2a_1$

　　　　　　　$a_1a_2a_3$

　　　　　　　$a_1a_3a_2$

　　　　　　　$a_2a_1a_3$

　　　　　　　$a_2a_3a_1$

分析n=3，排列按如下步骤进行：

(1) a_3之后跟a_1,a_2的所有全排列；

(2)在上述全排列里，a_3和a_1位置互换；

(3)在上述全排列里，a_3和a_2位置互换。

2.1 递归的概念----排列问题

- range(A) $\boxed{A\text{-}a_1}$ $\boxed{A\text{-}a_2}$ $\boxed{A\text{-}a_n}$

$= a_1range(A_1), a_2range(A_2),\ldots, a_n range(A_n)$

集合A用数组实现

range(A,1,n)：

递归出口：range(A, n, n)

递归调用：使得集合所有元素都可以作为前缀出现

2.1 递归的概念----排列问题

procedure range(A, k, n)
  if k=n then print(A)
    else for $i \leftarrow k$ to n do
        A(k) $\leftrightarrow$ A(i)
        call range(A,k+1,n)
        A(k) $\leftrightarrow$ A(i)
      repeat
  endif
end range

递归出口，打印整个数组A。

A(i)与A(k)值互换

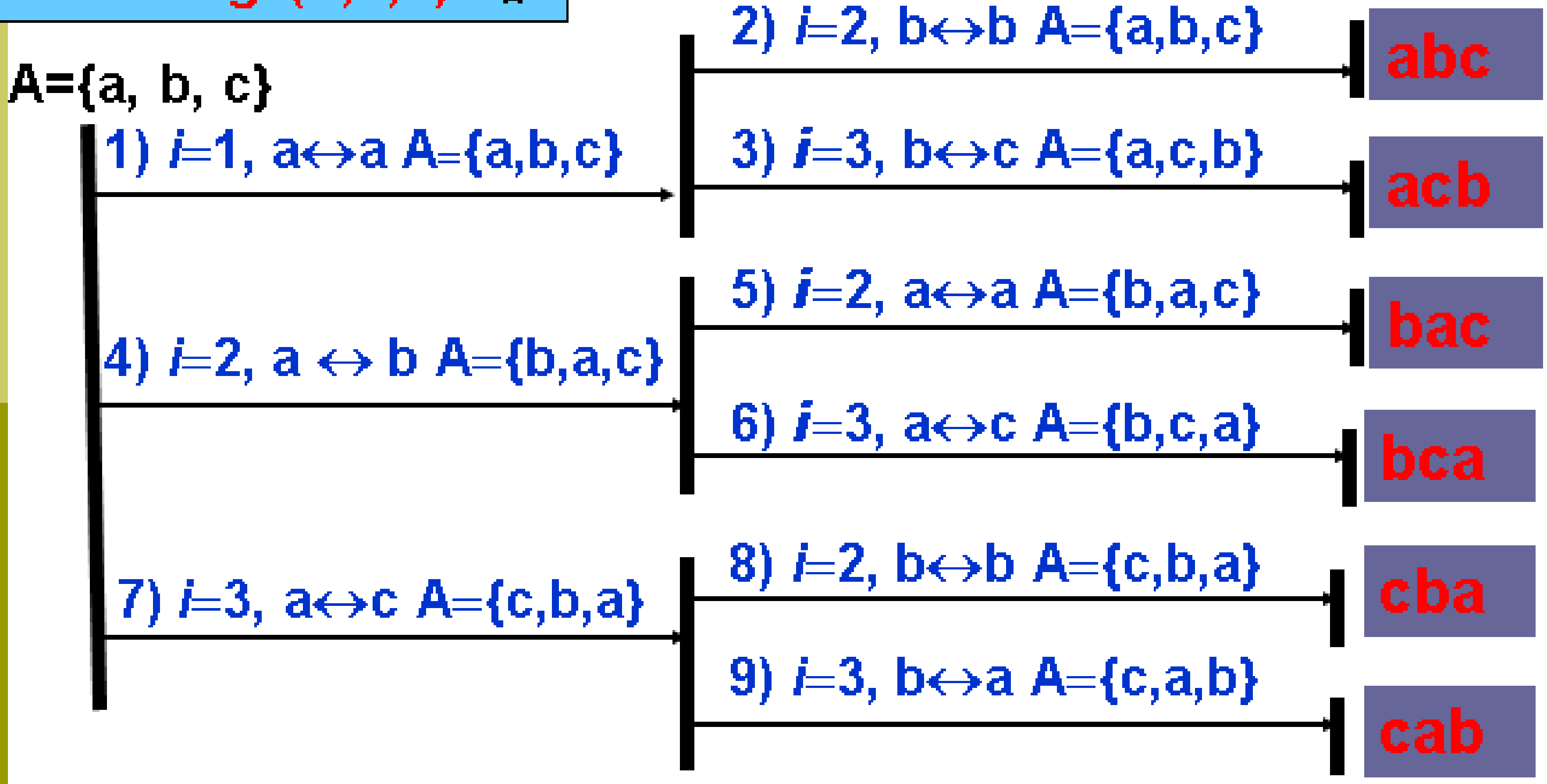
缺省值变化时不回传，交换返回原始状态。

call range(A,1,n)

```
range(A,1,3)
If 1=3 then print(A)
else
  for i ←1 to 3 do
    A(1)↔A(i);
   call range(A,2,3) 略
```

```
range(A,2,3)…
for i←2 to 3 do
      A(2)↔A(i);
      call range(A,3,3) 略
```

```
range(A,3,3)
If 3=3 print(A) 略
```



2.1 递归的概念----整数划分

- 任何一个大于1的自然数n，总可以拆分成若干个小于n的自然数之和，试求n的所有拆分。

- n=2　2=1+1
- n=3　3=1+2
  - =1+1+1
- n=4　4=1+3
  - =1+1+2
  - =1+1+1+1
  - =2+2

2.1 递归的概念----整数划分

- 分析：
- 将最大加数n_1不大于m的划分个数记作$q(n,m)$

$$q(n,m)=\begin{cases}1 \cdots\cdots\cdots\cdots & m=1 \vee n=1 \\ q(n,n) \cdots\cdots\cdots & n<m \\ 1+q(n,n-1) \cdots\cdots\cdots & n=m \\ q(n,m-1)+q(n-m,m) \cdots & 1<m<n\end{cases}$$

正整数n的划分数
p(n)=q(n,n)。

6；
5+1；
4+2，　　4+1+1；
3+3，　　3+2+1，　　3+1+1+1；
2+2+2，　　2+2+1+1，　　2+1+1+1+1；
1+1+1+1+1+1;

2.1 递归的概念----整数划分

$$q(n,m)=\begin{cases}1 \cdots\cdots & m=1 \vee n=1 \\ q(n,n) \cdots\cdots & n<m \\ 1+q(n,n-1) \cdots\cdots & n=m \\ q(n,m-1)+q(n-m,m) \cdots\cdots & 1<m<n\end{cases}$$

```
int q(int n,int m)
{    if((n<1)||(m<1)) return 0;
     if(n==1||m==1)  return 1;
     if(n<m)         return q(n,n);
     if(n==m)        return q(n,m-1)+1;
     return q(n,m-1) + q(n-m,m);
 }
```

2.1 递归的概念----简单的0/1背包问题

例：$m=20,\ n=5,$

$(m_1, m_2, m_3, m_4, m_5) = (3,5,8,9,10)$

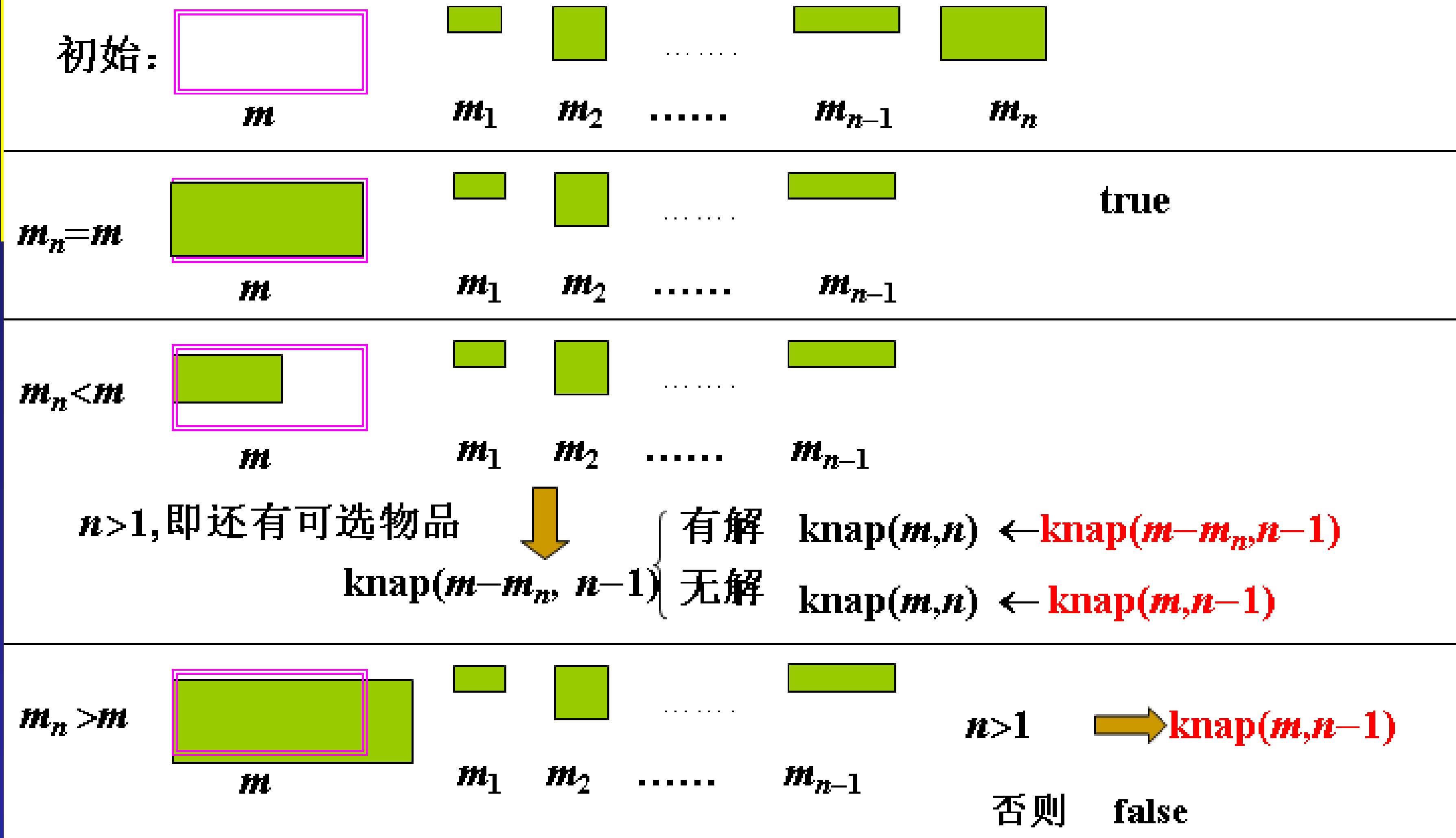
$(x_1,x_2,x_3,x_4,x_5)=(1,0,1,1,0)$

$m=18?$ $m=28?$

注：对于第*i*件物品要么取，要么舍，不能取一部分，因此这个问题可能有解，也可能无解。

布尔函数

问题分析 knap(m, n)



初始：
m
m_1 m_2 …… m_{n-1} m_n
m_n=m
true
m
m_1 m_2 …… m_{n-1}
m_n<m
m
m_1 m_2 …… m_{n-1}
n>1,即还有可选物品
knap($m-m_n$, $n-1$)
有解 knap(m,n) ←knap($m-m_n$,$n-1$)
无解 knap(m,n) ← knap(m,$n-1$)
m_n >m
m
m_1 m_2 …… m_{n-1}
n>1
knap(m,$n-1$)
否则 false

2.1 递归的概念---Hanoi塔传说

- 在贝拿勒斯（在印度北部）的圣庙里，一块黄铜板上插着三根宝石针。印度教的主神梵天在创造世界的时候，在其中一根针上从下到上地穿好了由大到小的**64**片金片，这就是所谓的汉诺塔(Tower of Hanoi）。
- 不论白天黑夜，总有一个僧侣在按照下面的法则移动这些金片：一次只移动一片，不管在哪根针上，小片必须在大片上面。僧侣们预言，当所有的金片都从梵天穿好的那根针上移到另外一根针上时，世界就将在一声霹雳中消灭，而梵塔、庙宇和众生也都将同归于尽。

假如每秒钟一次，移完这些金片需要**5845**亿年以上。

n阶Hanoi塔问题

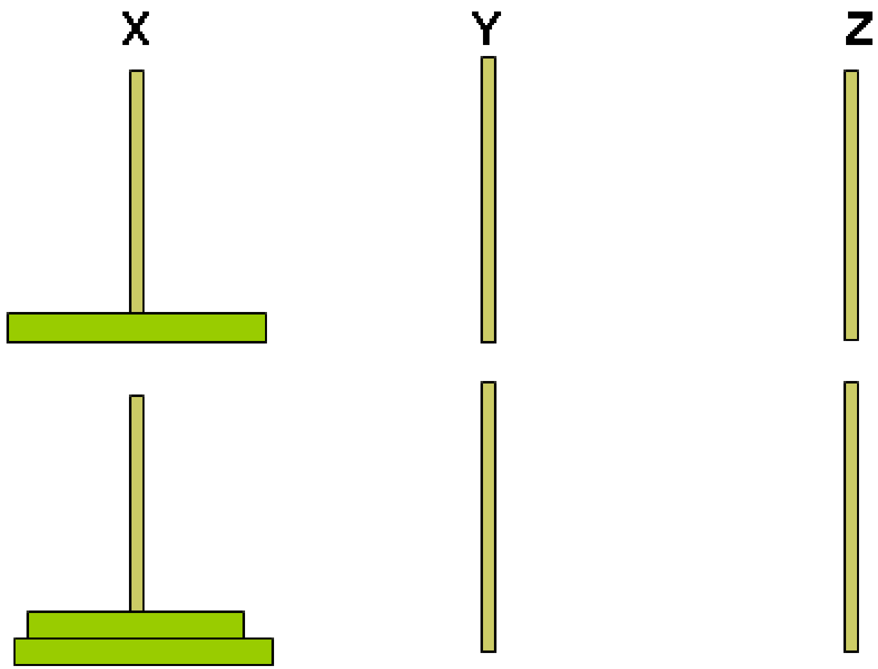
- 有n个圆盘依半径从小到大自上而下地套在柱子X上，柱子Y和Z没有圆盘。要求将X上的盘子换到Z上，每次只移动一个，且不允许将大圆盘压在小圆盘的上面。

2.1 递归的概念---Hanoi塔问题

寻找递归出口

圆盘数量　源柱　辅助柱　目标柱

Hanoi(n,X,Y,Z)



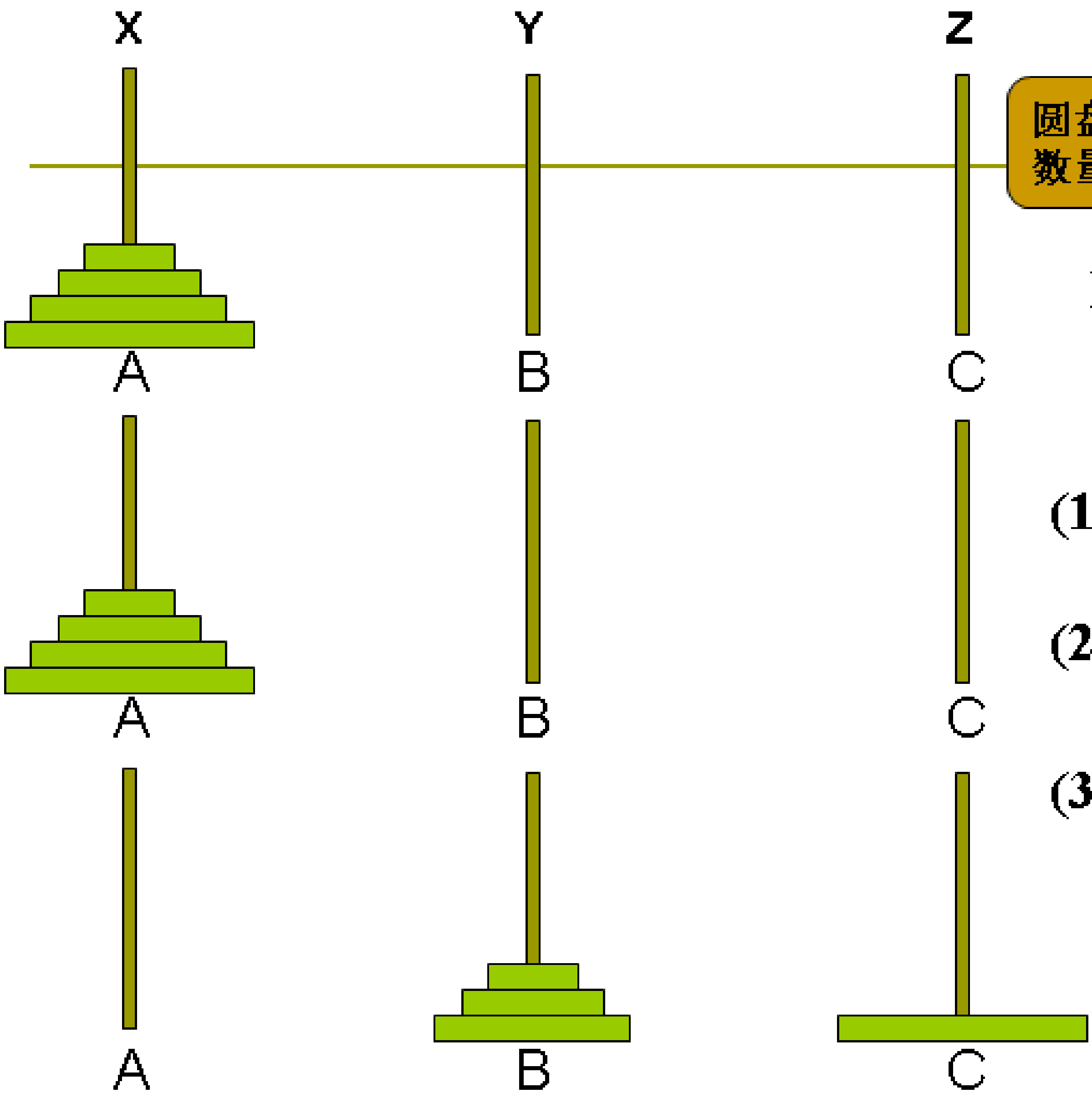
(1) n=1, $X^1 \rightarrow Z$

(2) n=2, $X^1 \rightarrow Y$

$X^2 \rightarrow Z$

$Y^1 \rightarrow Z$

2.1 递归的概念---Hanoi塔问题



圆盘数量 | 源柱 | 辅助柱 | 目标柱

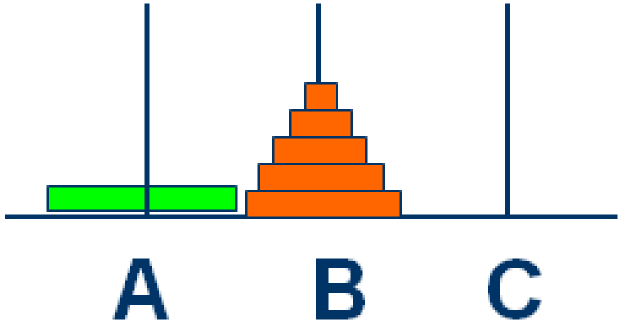
Hanoi(n,X,Y,Z)

(1) Hanoi(n-1,X,Z,Y)

(2) X ^{n}→Z

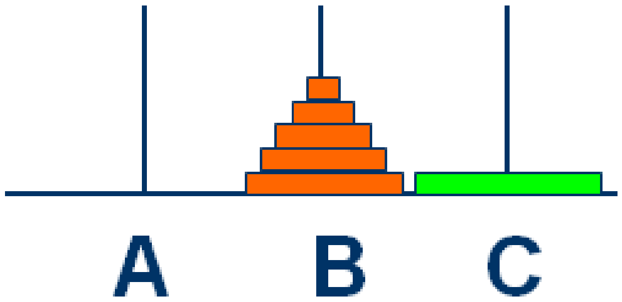
(3) Hanoi(n-1,Y,X,Z)

2.1 递归的概念---Hanoi塔问题



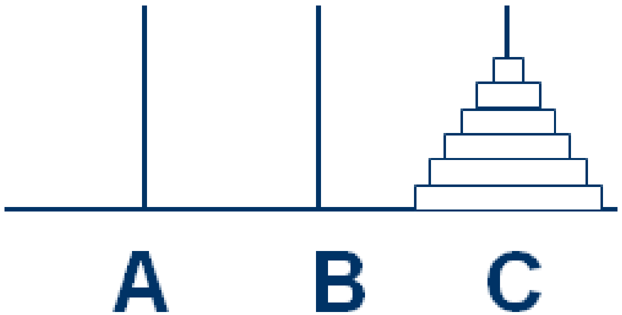
Hanoi(n-1,A,C,B)

= +



Move(n,A,C)

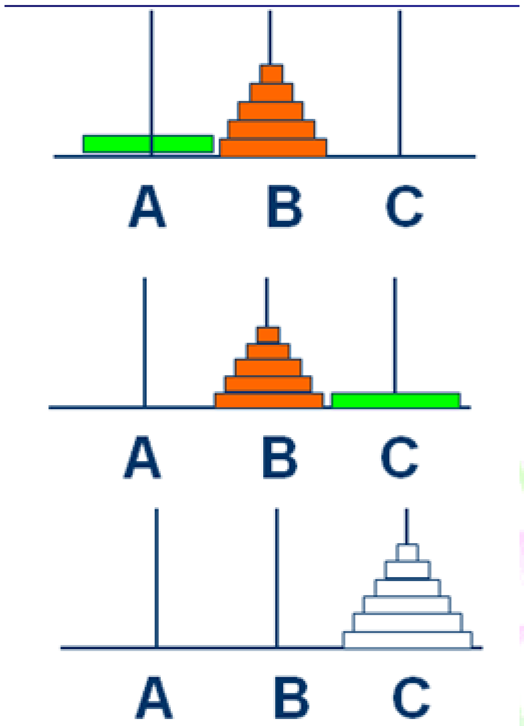
+



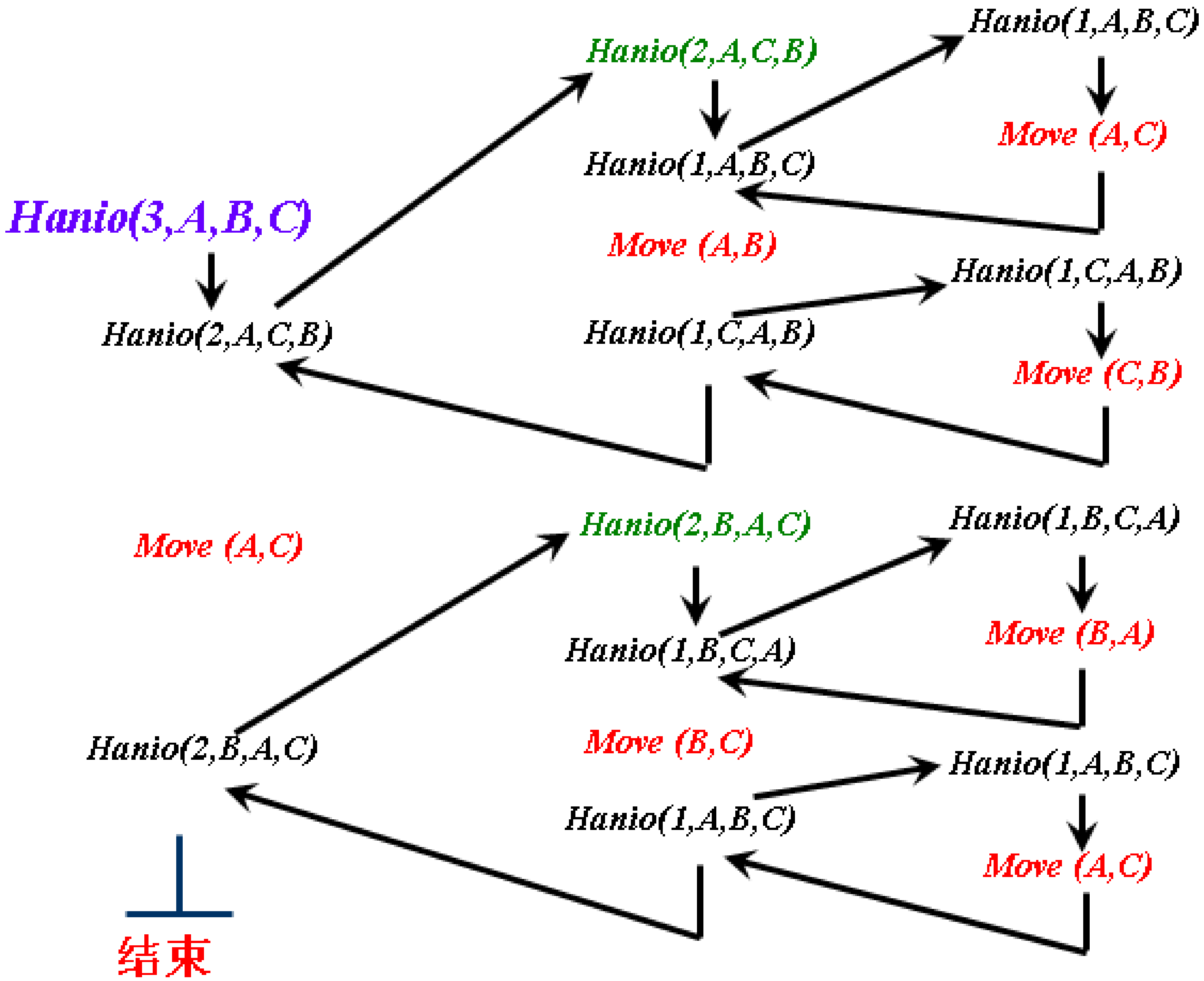
Hanoi(n-1,B,A,C)

递归求解

```
void hanoi(int n, int a, int b, int c)
  { if   (n== 1)          move(a,b);
     else   {
                 hanoi(n-1, a, c, b);
                 move(a,b);
                 hanoi(n-1, c, b, a);
             }
  }
```



递归函数的运行轨迹


Hanio(3,A,B,C)
Hanio(2,A,C,B)
Hanio(2,A,C,B)
Hanio(1,A,B,C)
Hanio(1,A,B,C)
Move (A,C)
Move (A,B)
Hanio(1,C,A,B)
Hanio(1,C,A,B)
Move (C,B)
Move (A,C)
Hanio(2,B,A,C)
Hanio(1,B,C,A)
Hanio(1,B,C,A)
Move (B,A)
Move (B,C)
Hanio(2,B,A,C)
Hanio(1,A,B,C)
Hanio(1,A,B,C)
Move (A,C)
结束

2.1 递归的概念---Hanoi塔问题

这个问题有一个简单的解法。假设塔座A、B、C排列成一个三角形，A——>B——>C——>A构成一顺时针循环。在移动圆盘的过程中，若是奇数次移动，则将最小的圆盘移到顺时针方向的下一塔座上；若是偶数次移动，则保持最小的圆盘不动。而其他两个塔座之间，将较小的圆盘移到另一塔座上去。

2.1 递归的概念---Hanoi塔问题

时间复杂性分析:

- 规模为*n*的*Hanoi(n)*问题，可以分解为2个规模为*n-1*的*Hanoi(n-1)*问题和一个*Move* 操作。
- 所以，*n*个盘子的移动次数为:

$$T(n)=\begin{cases}1 & n=1\\ 2T(n-1)+1 & n>1\end{cases}$$

$$\Rightarrow T(n)=2^n-1$$

若n=64，则移动次数为$2^{64}-1$

$2^{64}-1=18,446,744,073,709,551,615$

2.1 递归的概念---Hanoi塔问题

- $2^{64}-1=18,446,744,073,709,551,615$是个什么概念？
  - 实例1：
    - 假设每秒钟移动一次，一年约31556926秒，
    - 计算表明：移动64个盘子需要5800多亿年。
  - 实例2：
    - 国王的麦子问题
      - 一个高4米、宽10米的粮仓装麦子，这个粮仓有3000万公里长，能绕地球赤道700圈，可以把地球全部表面（包括海洋）铺上2米厚的小麦层！它相当于全世界2000多年小麦产量的总和.

2.1 递归的概念---查找数组最大数据项

- 练习：在数组a[0],……, a[n-1]的n个项中找出最大数据项。用递归算法来实现。

```
item max(item a[],int l,int r)
{ item u,v; int m=(l+r)/2;      //取数组下标的中点
  If (l= =r)
     return a[l]; //只有一个元素的情况，直接返回
  u=max(a,l,m);                 //递归地对左半部分取最大值
  v=max(a,m+1,r);               //递归地对右半部分取最大值
  if (u>v)
     return u;
  else return v;
}
```

算法是将数组a[1],…,a[r]分成a[1],…a[m]和a[m+1],…a[r]两部分，分别求出每一部分的最大元素（递归地），并返回较大的那一个作为整个数组的最大元素。

2.1 递归的概念

- 在运行递归算法时：系统需要在运行被调用算法之前完成三件事：

1. 将所有实参指针，返回地址等信息传递给被调用算法；
2. 为被调用算法的局部变量分配存储区；
3. 将控制转移到被调用算法的入口。

- 在从被调用算法返回调用算法时：系统也相应地要完成三件事：

1. 保存被调用算法的计算结果；
2. 释放分配给被调用算法的数据区；
3. 依照被调用算法保存的返回地址将控制转移到调用算法。

2.1 递归的概念

- 递归小结
  - **优点**：结构清晰，可读性强，而且容易用数学归纳法来证明算法的正确性，因此它为设计算法、调试程序带来很大方便。
  - **缺点**：递归算法的运行效率较低，无论是耗费的计算时间还是占用的存储空间都比非递归算法要多。
  - **解决方法**：在递归算法中消除递归调用，使其转化为非递归算法。