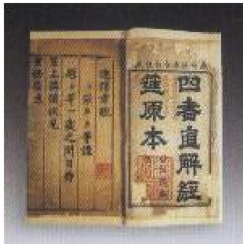
第二章 算法分析基础

1. 算法时间复杂性分析
2. 算法空间复杂性分析
3. 最优算法
4. 小 结

案例一——百鸡问题

公元5世纪末，我国古代数学家张丘建在他所撰写的《算经》中，提出了这样一个问题："鸡翁一，值钱五；鸡母一，值钱三；鸡雏三，值钱一。百钱买百鸡，问鸡翁、母、雏各几何？"

意思是公鸡每只5元、母鸡每只3元、小鸡3只1元，用100元钱买100只鸡，求公鸡、母鸡、小鸡的只数。



案例一——百鸡问题

令a为公鸡只数，b为母鸡只数，c为小鸡只数。列出约束方程：

$a+b+c=100$　　(1)

$5a+3b+c/3=100$　　(2)

$c\%3=0$　　(3)

分析: a、b、c的可能取值范围为0~100，对a、b、c的所有组合进行测试，满足约束方程的组合是问题的解。把问题转化为用 n 元钱买 n 只鸡，则上式变为：

$a+b+c=n$　　(1′)

$5a+3b+c/3=n$　　(2′)

算法1 百鸡问题

```
void chicken_question(int n,int &k,int g[],int m[],int s[])
{
    int  a,b,c;
    k = 0;
    for (a=0;a<=n;a++){
        for (b=0;b<=n;b++){
            for (c=0;c<=n;c++) {
                if ((a+b+c==n)&&(5*a+3*b+c/3==n)&&(c%3==0)) {
                    g[k] = a;
                    m[k] = b;
                    s[k] = c;
                    k++;
                }
            }
        }
    }
}
```

算法2 改进的百鸡问题

```
void chicken_problem(int n,int &k,int g[],int m[],int s[])
{int  i,j,a,b,c; k = 0; i = n/5; j = n/3;
for (a=0;a<=i;a++){
        for (b=0;b<=j;b++) {
              c = n–a–b;
              if ((5*a+3*b+c/3==n)&&(c%3==0)) {
                   g[k] = a;
                   m[k] = b;
                   s[k] = c;
                   k++;
                }
            }
     }
}
```

思考下面程序段

```
PARTITION(A， p， q)
// A是一个实数数组,p, q是该数组的上下限
{
    x←A[p]；  // A[p]被选中
    i←p ；
    for j←p+1 to q do
            if (A[j]≤x)  then {
              i←i+1；
              A[i] ↔A[j]；}// 交换A[i]和A[j]的内容
    A[p] ↔A[i]
    return i
}
```

算法分析——时间复杂性

基本概念

- 算法分析：对算法所需要的两种计算机资源——时间和空间进行估算。
- 问题规模：指输入量的多少。运行算法所需要的时间T是问题规模n的函数，记作T(n)。
- 基本语句：执行次数与整个算法的执行次数成正比的语句。

渐进符号——运行时间的上界

定义:
若存在两个正的常数c和n_0,对于任意$n \geq n_0$,都有$T(n) \leq cf(n)$,则称$T(n)=O(f(n))$ 。

大O符号描述增长率的上限，表示$T(n)$的增长最多像$f(n)$增长的那样快，这个上限的阶越低，结果越有价值。

该算法的运行时间至多是$O(f(n))$ 。

渐进符号——运行时间的上界

例如:

当有$T(n) \le 100n + n$

取$n_0 = 5$, 对任意$n \ge n_0$, 有:

$T(n) \le 100n + n = 101n$

令$c = 101$, $f(n) = n$, 有:

$T(n) \le cn = cf(n)$

所以$T(n) = O(f(n)) = O(n)$

练习:

当有$T(n) \le 19/15n^2 + 161/15n + 28$

则$T(n) = O($? $)$

渐进符号——运行时间的下界

定义

若存在两个正的常数c和n_0,对于任意$n \geq n_0$,都有$T(n) \geq cg(n)$,则称$T(n)=\Omega(g(n))$ 。

大Ω符号用来描述增长率的下限，也就是说，当输入规模为n时，算法消耗时间的最小值。与大 O符号对称，这个下限的阶越高，结果就越有价值。

该算法的运行时间至少是$\Omega(g(n))$ 。

渐进符号——运行时间的下界

例如:

当有$T(n) \geq n^2 + n \geq n^2$

取$n_0 = 1$，任意$n \geq n_0$，存在常数c=1，

$f(n)=n^2$，使得：$T(n) \geq n^2 = cf(n)$

所以, $T(n)=\Omega(g(n))$

练习:

当有$T(n) \geq 19/15n^2+161/15n+28$

则$T(n)=\Omega($ ？ $)$

渐进符号——运行时间的准确界

Θ符号(运行时间的准确界)

定义1.3 若存在三个正的常数c_1、c_2和n_0，对于任意$n \geq n_0$，都有$c_1 f(n) \geq T(n) \geq c_2 f(n)$，则称$T(n)=\Theta(f(n))$。

Θ符号意味着$T(n)$与$f(n)$同阶，用来表示算法的精确阶。

渐进符号——运行时间的准确界

例1.1 $T(n) = 3n\text{-}1$

【解答】

当$n \geq 1$时， $3n\text{-}1 \leq 3n = O(n)$

当$n \geq 1$时， $3n\text{-}1 \geq 3n\text{-}n = 2n = \Omega(n)$

当$n \geq 1$时， $3n \geq 3n\text{-}1 \geq 2n$， 则$3n\text{-}1 = \Theta(n)$

例1.2 $T(n) = 5n^2 + 8n + 1$

渐进符号——运行时间的准确界

【解答】当$n≥1$时，

$5n^2+8n+1 \le 5n^2+8n+n=5n^2+9n \le 5n^2+9n^2 \le 14n^2=O(n^2)$

当$n≥1$时，$5n^2+8n+1 \ge 5n^2=\Omega(n^2)$

当$n≥1$时，$14n^2 \ge 5n^2+8n+1 \ge 5n^2$，则$5n^2+8n+1=\Theta(n^2)$

算法时间复杂度分析——定理

练习：

1. $T(n)=5n+2$
2. $T(n)=8n^2+3n+2$
3. $T(n)=5\times 2^n+n^2$

定理1.1 若$T(n)=a_m n^m + a_{m-1} n^{m-1} + \ldots + a_1 n + a_0$（$a^m>0$），则有$T(n)=O(n^m)$，且$T(n)=\Omega(n^m)$，因此，有$T(n)=\Theta(n^m)$。

非递归算法的分析

1. 建立一个代表算法运行时间的求和表达式;
2. 用渐进符号表示这个求和表达式。

```
int ArrayMin(int a[ ], int n)
{
        min=a[0];
        for (i=1; i<n; i++)
            if (a[i]<min) min=a[i];
        return min;
}
```

非递归算法分析的一般步骤

1. 决定用哪个（或哪些）参数作为算法问题规模的度量

   可以从问题的描述中得到。

2. 找出算法中的基本语句

   通常是最内层循环的循环体。

3. 检查基本语句的执行次数是否只依赖于问题规模

   如果基本语句的执行次数还依赖于其他一些特性，则需要分别研究最好情况、最坏情况和平均情况的效率。

4. 建立基本语句执行次数的求和表达式

   计算基本语句执行的次数，建立一个代表算法运行时间的求和表达式。

5. 用渐进符号表示这个求和表达式

   计算基本语句执行次数的数量级，用大O符号来描述算法增长率的上限。

递归算法的分析

对递归算法时间复杂性的分析，关键是根据递归过程建立递推关系式，然后求解这个递推关系式。

汉诺塔益智玩具

```
void Hanoi(int n,char A,char B,char C)
{
      if (n==1)
            printf (A—>C);
     else
     {
            hanoi(n-1,A,C,B);
            printf(A—>C);
            hanoi(n-1,B,A,C);
     }
}
```

$$\mathrm{H}(1) = 1 \qquad (n=1)$$

$$\mathrm{H}(n) = 2*\mathrm{H}(n\text{-}1)+1 \qquad (n>1)$$

算法复杂度分析小结

基本技术：

- 根据循环统计基本语句次数
- 用递归关系统计基本语句次数
- 用平摊方法统计基本语句次数

选取基本语句

统计基本语句频数

计算并简化统计结果

渐近复杂度：

O、Ω、Θ

标准：

- 平均时间复杂
- 最坏时间复杂
- 最好时间复杂度

基本技术：

- 常用求和公式
- 定积分近似求和
- 递归方程求解

算法分析的核心——计数

渐近时间关注的是趋势

你认为算法是什么?

从哲学角度看：算法是解决一个问题的抽象行为序列。

从技术层面上看：算法是一个计算过程，它接受一些输入，并产生某些输出。

从抽象层次上看：算法是一个将输入转化为输出的计算步骤序列。

从宏观层面上看：算法是解决一个精确定义的计算问题的工具。