# 第一部分 基础知识

算法的设计和分析，算法表达方法，设计策略，基本思想。

第一章：算法定义，例子

第二章：排序 插入（增量式） 归并（递归）（分治）

第三章：渐近表示

第四章：分治法 求解递归式

第五章：概率分布，随机化算法

## 第一章 算法在计算机中的作用

### 1.1 算法

【算法】良定义的计算过程 取值或集合作为输入，产生值或集合的输出

【特征】1.存在许多候选解 2.存在实际应用

【数据结构】存储和组织方式，便于访问和修改

【技术】设计算法，证明正确性，理解效率

【难题】NP问题 1.存在有效算法未知 2.存在则任意 3.存在局部最由

【并行性】

### 1.2 作为一种技术的算法

【效率】不同算法在解决相同问题时效率不同

【技术】系统性能依赖于硬件速度和有效算法

## 第二章 算法基础

伪代码-证明正确性，分析运行时间-记号

### 2.1 插入排序

【循环不变式】初始为真，某次迭代之前为真，迭代之后为真，终止时一个性质证明正确

【伪代码】缩进，循环，注释，赋值，变量，数组，对象，指针，串联，参数，返回，短路

### 2.2 分析算法

【资源】内存，带宽，硬件资源，计算时间

【RAM模型】random-access machine 指令一条接一条执行，没有并发

【常见指令】算术指令（加法，减法，乘法，除法，取余，向下取整，向上取整）

数据移动指令（装入，存储，复制）

控制指令（条件与无条件转移，子程序调用与返回）

【输入规模】每个问题指出所使用的输入规模度量

【运行时间】执行的基本操作数或步数

【插入函数】最好情况an+b 最坏情况an2+bn+c

【只求最坏情况】1.最坏情况是上界 2.有时最坏情况经常出现 3.平均情况与最坏情况一样

【增长量级】最坏情况运行时间Θ

### 2.3 设计算法

#### 2.3.1 分治法

【分治模式】1.分解 2.解决 3.合并

#### 2.3.2 分析分治算法

【递归式】 T(n)= (n<=c时) Θ（1） （其他）=aT(n/b)+D(n)+C(n)

【递归树】 结论 分治法 时间复杂度：Θ（nlgn）

## 第三章 函数的增长

### 3.1 渐近符号

Θ为渐近紧确界 O为渐近上界 Ω为渐近下界 o为非渐近紧确上界 ω为非渐近紧确下界

【Θ记号】Θ(g(n))={f(n):存在c1,c2和n0，使得对所有n>=n0,0<=c1g(n)<=f(n)<=c2g(n)}

c1置为稍小于最高阶系数，c2置为稍大于最高阶系数

【O记号】O(g(n))={f(n):存在c和n0，使得对所有n>=n0,有0<=f(n)<=cg(n)}

Θ包含O

【Ω记号】Ω(g(n))={f(n):存在c和n0，使得对所有n>=n0,有0<=cg(n) <=f(n)}

【定理3.1】 f(n)= Θ(g(n))🡸🡺f(n)=O(g(n)) 且f(n)= Ω(g(n))

【等式和不等式中的渐近记号】渐进记号代表匿名函数

【o记号】o(g(n))={f(n):对任意c>0,存在n0，使得对所有n>=n0,有0<=f(n)<cg(n)}

【ω记号】ω(g(n))={f(n):对任意c>0,存在n0，使得对所有n>=n0,有0<=cg(n) <f(n)}

f(n)= ω(g(n))🡸🡺g(n)=o(f(n))

【性质】 传递性ΘOΩoω

自反性ΘOΩ

对称性Θ

转置对称性Oo

【类比】 O a<=b Ω a>=b Θ a=b o a<b ω a>b <=> 可能都不成立

### 3.2 标准记号与常用函数

单调性

向下取整或向上取整

模运算

多项式

指数

对数

阶乘

多重函数

多重对数函数

斐波那契数

## 第四章 分治策略

递归情况，基本情况

代入法，递归树法，主方法

### 4.1 最大子数组问题

暴力求解方法 Ω(n2)

分治法 最大子数组=max(左半边最大，右半边最大，穿过中间的数组)

时间复杂度Θ(nlgn)

### 4.2 矩阵乘法的Strassen算法

直观方法 Ω(n3)

Strassen算法:将矩阵分成4个子矩阵 通过巧妙设计将8次矩阵乘法变为7次矩阵乘法

### 4.3 用代入法求解递归式

【代入法】1.猜 2.归纳法证明

一些奇奇怪怪的技巧

### 4.4 用递归树方法求解递归式

没懂

### 4.5 用主方法求解递归式

【问题】T(n)=aT(n/b)+f(n)

规模为n的问题分解为a个子问题，每个规模为n/b，花费为T(n/b)，问题分解和合并f(n)

【主定理】 f(n)=O(nlogba-ε) T(n)= Θ(nlogba)

f(n)=Θ(nlogba) T(n)= Θ(nlogbalgn)

f(n)=Ω(nlogba+ε) T(n)= Θ(f(n))

中间是有间隙的

### 4.6 证明主定理

没懂，不考

## 第五章 概率分析和随机算法

### 5.1 雇用问题

问题：如果当前应聘者更合适，就辞退当前的办公助理。计算费用。

【最坏情况】严格递增序列

【概率分布】所有可能输入的运行时间取平均

【随机算法】输入由随机数生成器决定，随机算法的运行时间称为期望运行时间

### 5.2 指示器随机变量

【指示器随机变量】发生为1不发生为0

雇用问题 E[Xi]=1/I E[X] = O(chlnn)

### 5.3 随机算法

随机选择待选的助理，费用期望是O(chlnn)

产生优先级序列来生成随机化数组

假设所有优先级不同，则产生的输入序列均匀随机排列

循环不变式证明

### 5.4 概率分布和指示器随机变量的进一步使用

#### 5.4.1 生日悖论

1\*（1-1/n）\*(1-2/n)\*…\*(1-(k-1)/n)

k>=(1+sqrt(1+(8ln2)n)/2) n=365 k=23

#### 5.4.2 球与箱子

在给定箱子中投球的概率是1/b

每个箱子中都有球的概率是blnb

#### 5.4.3 特征序列

投n次硬币，最长连续正面的期望是lnn

#### 5.4.4 在线雇用问题

前k个计算出最大值，之后取第一个大于最大值的人

K=n/e 至少以1/e的概率成功雇用

# 第二部分 排序和顺序统计量

排序：通过对序列重排使得所有后一个比前一个大

原因：1.本身需要排序 2.许多算法作为子程序 3.现有量大 4.拓展其他问题 5.工程问题

排序算法

|  |  |  |
| --- | --- | --- |
| 算法 | 最坏情况运行时间 | 期望运行时间 |
| 插入排序 | Θ(n2) | Θ(n2) |
| 归并排序 | Θ(nlgn) | Θ(nlgn) |
| 堆排序 | O(nlgn) | - |
| 快速排序 | Θ(n2) | Θ(nlgn) |
| 计数排序 | Θ(k+n) | Θ(k+n) |
| 基数排序 | Θ(d(n+k)) | Θ(d(n+k)) |
| 桶排序 | Θ(n2) | Θ(n) |

## 第六章 堆排序

堆排序：时间复杂度O(nlgn) 具有空间原址性

### 6.1 堆

【堆】近似完全二叉树 当前元素为i 父节点 i/2 左孩子 2i 右孩子2i+1

【最大堆】每个父节点大于等于子节点，最小堆相反

【高度】Θ(lgn)

### 6.2 维护堆的性质

【MAX-HEAPIFY】维护最大堆性质的过程：当孩子比父亲大时，交换，然后递归

T(n)<=T(2n/3)+ Θ(1) = O(lgn)=h

### 6.3 建堆