# 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

n/2+1…n 是叶结点，只需对n/2 …1执行MAX-HEAPIFY就行

循环不变量证明

更紧确的界 O(n)

### 6.4 堆排序算法

将根结点与当前序列最后一位交换，调用MAX-HEAPIFY

### 6.5 优先队列

每个元素有一个关键字

INSERT, MAXIMUM, EXTRACT-MAX, INCREASE-KEY

【应用】作业调度，基于事件驱动的模拟器

## 第七章 快速排序

最坏时间复杂度Θ(n2) 期望时间复杂度为Θ(nlgn) 常数因子很小 原址排序

### 7.1 快速排序的描述

【分解】根据选取元素将数组化为左右两半，左边都小于，右边都大于

【解决】递归调用分别对两边排序

【合并】原址排序不需合并

【PARTITION划分】选取主元，通过交换将小的放在左边，大的放在右边

循环不变量证明、

### 7.2 快速排序的性能

运行时间-依赖-划分平衡

【最坏情况划分】有一边为0个 T(n)=T(n-1)+ Θ(n)= Θ(n2)

【最好情况划分】划分为n/2和n/2+1 T=2T(n/2)+ Θ(n)= Θ(nlgn)

【平衡划分】假设9:1划分 T(n)=T(9n/10)+T(n/10)+cn

递归在log10/9(n)= Θ(lgn)终止 时间复杂度为Θ(nlgn)

### 7.3 快速排序的随机化版本

【随机抽样RANDOMIZED】随机选择一个元素作为主元

### 7.4 快速排序分析

#### 7.4.1 最坏情况分析

#### 7.4.2 期望运行时间

？？？

## 第八章 线性时间排序

O(nlgn)算法：归并排序，堆排序，快速排序 【比较排序】

### 8.1 排序算法的下界

【决策树模型】完全二叉树，表示给定输入规模，特定排序算法对所有元素的比较操作

比较排序算法最坏情况比较次数等于决策树高度

【定理8.1】最坏情况下，任何比较排序算法都需要Ω(nlgn)次比较

### 8.2 计数排序

C[1…k]记录下标数字的个数，迭代累加为对应位置下标

B[1…n]下标为C,值为A C递减

时间复杂度为Θ(k+n) k=O(n)时 运行时间为Θ(n)

稳定性

### 8.3 基数排序

【基数排序】卡片排序机 一次只能查看一列

按最低有效位进行排序 每位数排序必须是稳定的

【引理8.3】n个d位数 每个数有k个可能值 时间复杂度Θ(d(n+k))

【引理8.4】n个d位数，r<=b 时间复杂度Θ((b/r)(n+2r))

b<lgn r<=b (n+2r)= Θ(n)

b>lgn r<lgn时 (n+2r)= Θ(n)

r>lgn时 (n+2r)=(bn/lgn)

通常 b=O(lgn) r≈lgn 运行时间为Θ(n)

【与快排比较】常数项较大，依赖于具体实现和底层硬件特性

快排更能有效使用硬件缓存

### 8.4 桶排序

计数排序假设输入数据属于小区间内的整数，桶排序假设输入随机产生

【桶排序】所有元素位于0~1，需要临时数组[0…n-1]存放链表，每个元素存在在对应桶中，采用插入排序

时间代价 T(n)= Θ(n)+ΣO(ni2)=…=Θ(n)

## 第九章 中位数和顺序统计量

第i个顺序统计量为集合中第i小的元素

中位数是集合的中点元素 下中位数：(n+1)/2 上中位数：(n+2/2)

直观：通过O(nlgn)排序，将下标为i的元素输出

### 9.1 最小值和最大值

依次遍历集合中每个元素，时间复杂度为O(n)

【同时找最大值和最小值】 直观法：2n-2次 每次取一对，小的和min比 大的和max比

### 9.2 期望为线性时间的选择算法

每个步骤，选取主元，如果大于i在左边找，小于i在右边找

最坏情况Θ(n2) 期望为Θ(n)

### 9.3 最坏情况为线性时间的选择算法

1.划分分组，每组5个数

2.取中位数

3.找出中位数的中位数x

4.按照x划分输入数组

5.递归查找

最差时间复杂度是线性的

# 第三部分 数据结构

使用指针的简单数据结构表示动态集合

栈，队列，链表和有根树

### 10.1 栈和队列

【栈】后进先出

PUSH压入 POP弹出

【队列】先进后出

ENQUEUE入队 DEQUEUE出队

### 10.2 链表

【链表】链表的顺序由对象的指针决定，数组的顺序由数组的下标决定

【双向链表】包含next和prev关键字

【链表查找】运行时间Θ(n)

【链表插入】头插法O(1)

【链表删除】O(n)

【哨兵】不能降低渐近时间界，可以降低常数因子

### 10.3 指针和对象的实现

三个数组key,next,prev实现链表功能

单数组表示链表功能

【自由表】自由对象保存在一个单链表中 分配和释放

### 10.4 有根树的表示

【二叉树】p指向父节点 left指向左孩子 right指向右孩子

【分支无限制的有限树】左孩子右兄弟表示法

【其他表示法】堆表示 单数组

## 第11章 散列表

动态集合结构 支持INSERT, SEARCH, DELETE字典操作

散列表 最坏情况 查找时间为Θ(n) 平均时间为Θ(1)

数组概念的推广，为每个可能的关键字保留位置，以利用直接寻址技术

使用长度与实际存储的关键字数目不成比例的数组来存储

根据关键字计算下标，不直接把关键字作为数组下标

### 11.1 直接寻址表

动态集合每个元素取自全域U={0…m-1} 使用数组T[0…m-1]存储，

每个位置称为槽，指向关键字为k的元素

### 11.2 散列表

【直接寻址】如果全域U很大，连续空间不好找，而且很浪费

【散列表】存储需求降为Θ(|k|)

【散列函数】由关键字k计算h(k)作为散列值映射到槽位上

【冲突】两个关键字映射到同一个槽上

避免冲突：散列函数尽可能随机

解决冲突：链接法，开放寻址法

【链接法】将散列到同一槽中的元素放在一个链表中。

插入O(1) 查找O(l)(l:表的长度)

【分析】最坏情况，映射到同一槽 查找时间Θ(n)

散列函数h平均性能:关键字分布在m个槽位上的均匀程度

【简单均匀散列】任何一个给定元素等可能的散列到m个槽中的任何一个，且与位置无关

列表长度的期望=n/m

【定理11.1】 简单均匀散列，一次查找的平均时间为Θ(1+α) α为链表的期望长度

如果槽与表中元素数成正比，全部字典操作平均情况下时间为O(1)

### 11.3 散列函数

【好的散列特点】每个关键字等可能的散列到m个槽中，

将相近的关键字散列到相同槽的可能性最小化

独立于数据可能存在的形式

相似的关键字具有截然不同的散列值

将关键字转换为自然数

11.3.1 除法散列法

h(k)=k mod m 映射到m个槽中 m取2的幂附近的质数

11.3.2 乘法散列法

H(k)=m(kA mod 1) m取2的次幂 A=(sqrt(5)-1)/2 =0.618

11.3.3 全域散列法

随机选择散列函数，独立于要存储的关键字

？？？

11.3.4 开放寻址法

所有元素都存在散列表中

线性探查：h(k,i)=(h’(k)+i) mod m

依次向后查找，如果到头则循环

问题：一次群集 随着连续占用的槽增加，平均查找时间增加

二次探查：h(k,i)=(h’(k)+c1i+c2i2) mod m

偏移量依赖于序号I

二次群集：初始位置相同，探查序列也相同

双重探查：h(k,i)=(h1(k)+ih2(k)) mod m

以两种不同方式依赖于关键字k

要求：h2(k)与m互素

开放寻址法的分析

装载因子α=n/m<1 假设是均匀散列 不成功查找的期望探查次数至多为1/(1-α)

均匀散列 装载因子α 中插入一个元素至多需要1/(1-α)次探查

期望次数为1/α(ln(1/(1-α)))

### 11.5 完全散列

【完全散列】最坏情况下用O(1)次访存完成

【二次散列表】

## 第12章 二叉搜索树

操作：SEARCH,MINIMUN,MAXIMUM,PREDECESSOR,SUCCESSOR,INSERT,DELETE

作为字典，优先队列

基本操作最坏运行时间Θ(lgn)

### 12.1 什么是二叉搜索树

每个节点包含left,right,p 任何节点左子树关键字都小于它，右子树都大于它

【中序遍历】跟位于中间 【先序遍历】跟位于前面 【后序遍历】跟位于后面

遍历 时间复杂度Θ(n)

### 12.2 查询二叉搜索树

【查找】按照关键字大小，小的在左边，大的在右边 时间复杂度O(h)

【最值】 最小值：一直搜左子树 最大值：一直搜右子树

后继：大于关键字的最小关键字结点 前驱：小于关键字的最大关键字结点

查找时间复杂度O(h)

### 12.3 插入和删除

【插入】遍历找到叶节点，插入

【删除】叶节点直接删除，

只有一个孩子，直接替换

有两个孩子，找后继替换当前位置，递归旋转

动态集合操作时间复杂度O(h)

### 12.4 随机构建二叉搜索树

最坏情况高度为n-1 按随机次序插入能改善这种情况

一颗n个不同关键字随机构建的二叉树期望高度为O(lgn)

## 第13章 红黑树

### 13.1 红黑树的性质

每个节点5个属性：color,key,left,right,p

【性质】1.每个节点或是红色的，或是黑色的

2.根节点是黑色的

3.每个叶节点(NIL)是黑色的

4.如果一个节点是红色的，则它的两个子节点是黑色的

5.每个节点，到所有叶节点包含相同的黑色节点

黑色节点个数称为黑高bh(x)

n个内部节点红黑树的高度至多为2lg(n+1)

### 13.2 旋转

左旋，右旋 时间复杂度O(1)

### 13.3 插入

情况1 叔节点是红色

情况2 叔节点是黑色，z是右孩子

情况3 叔节点是黑色，z是左孩子

循环不变量证明

时间复杂度 O(lgn)

### 13.4 删除

情况1：x的兄弟节点w是红色的

情况2：x的兄弟节点w是黑色的，且w两个子节点都是黑色

情况3：x的兄弟节点w是黑色的，w左孩子是红色，右孩子是黑色

情况4：x的兄弟节点w是黑色的，w右节点是红色

时间复杂度：O(lgn)

## 第14章 数据结构的扩张

创造出一类全新类型的数据结构

### 14.1 动态顺序统计

红黑树 O(lgn)时间内确定顺序统计量，计算一个元素的秩

【顺序统计树】x.size = left.size + right.size + 1

【秩】中序遍历树时输出的位置

找出第i小的，计算根节点的秩，大于i左边迭代，小于i右边迭代

循环不变量证明秩的计算

### 14.2 如何扩张数据结构

步骤： 1.选择一种基础数据结构

2.确定基础数据结构中要维护的附加信息

3.检验基础数据结构上的基本修改操作能否附加信息

4.设计一些新操作

### 14.3 区间树

？？？

# 第四部分 高级设计和分析技术

技术：分治，随机化，递归，动态规划，贪心算法，摊还算法

## 第15章 动态规划

分治：分解成不相交的子问题，递归求解然后组合求出原问题

动归：求解最优化问题，很多可行解，希望找出最优解

步骤： 1.刻画一个最优解的结构特征

2.递归地定义最优解的值

3.计算最优解的值，通常采用自底向上的方法

4.利用计算出的信息构造一个最优解

### 15.1 钢条切割

【问题】给定长度为n的钢条和价格表，求切割方案，使得收益最大

长度n的钢条有2n-1种不同切割方案

rn=max(pn,r1+rn-1,r2+rn-2,…,rn-1+r1)

【最优子结构】最优解由相关子问题最优解组合而成，子问题可以独立求解

问题：反复调用相同参数值进行递归调用，时间复杂度很高

动态规划解决：每个子问题求解一次，将结果保存

【时空权衡】付出额外的内存空间来节约计算时间 。将指数时间转化为多项式时间

实现方式：1.带备忘的自顶向下法 2.自底向下法

【子问题图】每个顶点对应一个子问题，有向边需要用到子问题的解

运行时间与顶点和边的数量呈线性关系

【重构解】不仅保留最优解，也保留对应的切割方案

### 15.2 矩阵链乘法

【问题】n个矩阵链乘，求完全括号化方案，使得乘积所需的乘法次数最少

【卡塔兰数】相似的递归公式产生的序列，增长速度为Ω(4n/n3/2)

### 15.3 动态规划原理

【要素】最优子结构，子问题重叠

【最优子结构】最优解包含子问题的最优解

1. 做出一个选择，这次选择会产生一个或多个待解的子问题
2. 假定知道哪种选择会得到最优解
3. 确定最优解的选择会产生的子问题，如何更好刻画子问题空间
4. 每个子问题的解都是本身的最优解

【不同结构体现】1.涉及多少子问题 2.多少种选择

【子问题无关】一个子问题的解不影响另一个子问题的解

【重叠子问题】递归算法反复求解相同的子问题

### 15.4 最长公共子序列

【LCS问题】给定两个序列，求最长公共子序列

### 15.5 最优二叉搜索树

【最优二叉搜索树】每个关键字有对应搜索频率，构造二叉搜索树使得期望搜索代价最小

## 第十六章 贪心算法

更简单高效算法，每一步选择当时最佳的选择，即贪心算法

### 16.1 活动选择问题

【问题】一个教室多个活动，希望选出最大兼容活动集

### 16.2 贪心算法原理

【0-1背包问题】

### 16.3 赫夫曼编码

### 16.4 拟阵和贪心算法

### 16.5 用拟阵求解任务调度问题

## 第17章 摊还算法

## 第六部分 图算法

### 第22章 基本的图算法

#### 22.1 图的表示

1. 邻接链表 稀疏图

2. 邻接矩阵 稠密图

权重图

#### 22.2 广度优先搜索

#### 22.3 深度优先搜索

第23章 最小生成树

23.1 最小生成树的形成

23.2 Kruskal算法和Prim算法

第24章 单源最短路径

Bellman-Ford算法

Dijkstra算法

第25章 所有节点对的最短路径

Floyd-Warshall算法

Johnson算法

第26章 最大流