**1** 引言**(ch1)**

1. 什么是算法及其特征；

确定性 有限性 正确性 通用性 输入输出

程序和算法不一样

操作系统不是算法 而是程序

2.问题实例和问题规模。

Problem-问题-描述问题

Instance of a problem-问题实例-有具体实例

**2** 算法初步**(ch2)**

1. 插入排序算法；

循环不变式证明：初始化 保持 终止

其中插入不能使用二分

2.算法复杂性及其度量

(1)时间复杂性和空间复杂性；

(2)最坏、最好和平均情形复杂性；

3.插入排序的最坏、最好和平均时间；

最坏和平均相同 最好是n

4.归并排序算法及其时间复杂性。

分治

Merge算法（哨兵）：Merge(A,q,r)

1. 描述一个算法O(nlgn)时间复杂度，使得找出来S中存在两个相加和等于x的元素

(利用归并算法，同时设置双指针，一个指针指向子数组的头部，一个指向子数组的尾部，我觉得需要反证法证明一下一定经过那一对)

1. 在归并排序中对小数组采用插入排序（O(nk+nlg(n/k))）
2. 第三版的逆序对算法

（利用归并排序，设置双指针，同时指向两个子数组的末尾，第一子数组的指针固定，第二子数组的指针往前移动）

**3** 函数增长率**(ch3)**

1. 渐近记号 O 、Ω 、θ 的定义及其使用；
2. θ定义中的两个全部是正常量，并且需要对所有n>n0都成立
3. O定义是以其中函数为上界的集合
4. Ω与b相反
5. 小O与小Ω的定义也要注意一下
6. 并非所有函数都是可以渐进比较
7. 标准复杂性函数及其大小关系；

a. 多重对数函数

3.和式界的证明方法。

**4** 递归关系式**(ch4** ，**Sch1)**

1. a. 最大子数组问题：找出相加是最大的子数组，用二分的是O(nlgn) ，对4.1-5只需要记录之前最小[1:i]的子数组和即可，然后再记录当前数组和，就可以时间复杂度到O(n)

b. Stranssen 矩阵乘法,先算了一波S矩阵，再递推算出七个P矩阵（这里涉及7次子矩阵乘法）最后加减得出结果。（4.2-3.如果矩阵大小非2的次幂，就普通的分奇偶性讨论），（4.2-4，递归树考虑每层代价和叶子结点处理代价）

1.替换法：

(1)猜测解数学归纳法证明；（边界情况可直接设初始情况为1；减去低阶项）

(2)变量变换法。

2.迭代法：

(1)展开法；

(2)递归树法。

3.主定理。

4.补充 1：递归与分治法(sch1)

- 递归设计技术；

- 递归程序的非递归化；

- 算法设计：

(1)Fibonacci 数； (2)生成全排列；

(3)二分查找； (4)大整数乘法；

(5)Stranssen 矩阵乘法； (6)导线和开关(略)。

**5** 堆排序**(ch6)**

1 堆的概念和存储结构；

堆的形态实际上是完全二叉树（对应于一个数组）

2.堆的性质和种类；大顶堆 小顶堆

3.堆的操作：建堆；整堆；

MAX-HEAPIFY（A,i）用于维护堆，且只需要该结点的两个子树均需要是堆

时间复杂度为O（lgn）

BUILD-MAX-HEAP(A)用于建立堆，从非叶子结点开始递归使用MAX-HEAPIFY自下而上保持堆结构。复杂度为O(n)

该定理证明依赖于：堆最多包含高度为 h的结点个数为（n/2^(h+1)）。

Delete-Heap用于删除

1. 堆排序算法和时间复杂性；

heapsort加上建堆，每一次取出顶，把最右下角的元素置顶，再调整，O(n)+nO(lgn)=O(nlgn)

5.优先队列及其维护操作。

这里的优先队列采用队列实现，插入关键词，返回最大值，返回最大值并删除，关键词增大。

**6** 快速排序**(ch7)**

1.快速排序算法及其最好、最坏时间和平均时间；最好O(nlgn)，最差O(n2)

2.随机快速排序算法及其期望时间；

3.Partition 算法。Partion算法O(n)

随机快排平均时间复杂度证明

**7** 线性时间排序**(ch8)**

1.基于比较的排序算法下界： Ω (nlogn)；

2.计数排序适应的排序对象、算法和时间；

计数排序要求范围已知，所以总时间复杂度为O(n+k)

3.基数排序适应的排序对象、算法和时间；

基数排序使用条件是必须为d位数，可以使用计数排序，需要排序的稳定的，时间复杂度为O(d(n+k))

O((b/r)(n+2^r))，r为选取的一个数位2^r-1，此时只需要r=lgn可以O(n)

4.桶排序适应的排序对象、算法和时间。

均匀独立分布的数据在某个区间上，

**8** 中位数和顺序统计**(ch9)**

1. 最大和最小值的求解方法；

都是O(n)，还有一个求第二小的元素（第二大），直接用二叉树可以做到只需要n+lgn-2次数的比较。

1. 期望时间为线性的选择算法；

用Random-select随机算法基于random-partition即可

1. 最坏时间为线性的选择算法及其时间分析。

知道算法运作，以及证明其时间复杂度

**9** 红黑树**(ch13)**

1. 红黑树的定义和节点结构；

根是黑色，叶节点是黑色，一个结点是红色，两个子节点为黑色，每个结点到所有后代叶子结点的简单路径上均包含相同数目的黑色的结点。

1. 黑高概念；

从一个结点开始到达一个叶节点的简单路径的黑色结点个数为黑高。

1. 一棵 *n* 个内点的红黑树的高度至多是 2log(*n*+1)；

依赖于一个引理，bh(x)的黑高的内部结点至少包含2^bh(x)-1个内部结点。

黑高至少为h/2，所以n>=2^(h/2)-1

1. 左旋算法；
2. 插入算法的时间、至多使用 2 次旋转；

插入看叔结点，如果是红直接父叔那一层变换成黑色，然后指针往上移到祖父结点即可，如果是黑则需要转两次即可。

6.删除算法的时间、至多使用 3 次旋转。

删除O(lgn)

**10** 数据结构的扩张**(ch14)**

1. 动态顺序统计：

首先每一个结点维护一个size的值，size代表其中子树的结点个数。

扩展红黑树，选择问题(中序遍历的rank，给定 Rank 求相应的元素，O(lgn))，Rank 问题(求元素 x 在集合中的 Rank，只需要)

插入：维护size的操作，第一阶段下降的过程时间复杂度为O(lgn)，第二阶段插入只涉及到旋转，且旋转的次数最多为2次，所以为O(1)。

删除：第一阶段删除重构树的结构，可以得到从删除结点到根节点的一条简单路径，然后更新O(lgn)，第二阶段则是最多三次旋转，所以O(1)

(1)节点结构的扩展；

(2)选择问题的算法；

(3)Rank 问题的算法；

(4)维护树的成本分析；

2.如何扩张一个数据结构：

扩张的步骤；

a.选择一种基础数据结构。

b. 确定基础数据结构中要维护的附加信息。

c. 检验基础数据结构上的基本修改操作能否维护附加信息。

d. 设计一些新操作。

扩张红黑树的定理；

一个结点的值取决于自己和左节点与右节点，那么在查询、删除、插入操作中，可以用O(lgn)的复杂度维护且完成操作。

3.区间树的扩张和查找算法。

区间树的扩展的定义（全部是闭区间，且额外存储的数是区间的右侧的最大值，按照区间左侧进行二叉排序红黑树），查找算法一定可以找到（否则就是没有）。

**11** 动态规划**(ch15)**

1. 方法的基本思想和基本步骤；

描述最优解结构

递归定义最优解值

自底向上的方法计算最优解值

由计算结果构造最优解

1. 动态规划和分治法求解问题的区别；

动态规划有分治法的思想，但是备份了相同子问题的解。

1. 最优性原理及其问题满足最优性原理的证明方法；

4.算法设计：

(1)多段图规划； (2)矩阵链乘法；

(3)最大子段和； (4)最长公共子序列。

（４）０１背包

**12** 贪心算法**(ch16)**

1. 方法的基本思想和基本步骤；

将问题转换为：先做出选择，再解决剩下的一个子问题

证明贪心的选择是合理的

说明选择贪心以后，贪心子问题的解和之前选择联合起来的合理性。

2.贪心算法的正确性保证：满足贪心选择性质；

3.贪心算法与动态规划的比较；

相同：都有最优子结构的性质

不同：一个有贪心选择性质，一个没有

4.两种背包问题的最优性分析：最优子结构性质和贪心选择性质； 5.算法设计：

(1)小数背包； (2)活动安排；

(3)找钱问题。

**13** 回溯法**(sch2)**

1.方法的基本思想和基本步骤；

2.回溯法是一种深度遍历的搜索；

3.术语: 三种搜索空间, 活结点, 死结点, 扩展结点, 开始结点, 终端结点； 4.两种解空间树和相应的算法框架 ；

5.算法设计：

(1)图和树的遍历； (2)*n* 后问题；

(3)0- 1 背包； (4)排列生成问题；

(5)TSP 问题。

**14** 平摊分析**(ch17)**

1.平摊分析方法的作用和三种平摊分析方法各自特点；

2.聚集分析法及应用；

3.记账分析法及应用；

4.势能法及应用。

**15** 二项堆**(ch19** in textbook version 2**)**

1. 为什么需要二项堆？ 二项堆和二叉堆上的几个基本操作时间复杂性；

因为需要降低UNION操作时间复杂度，从ｎ到ｌｇｎ。

1. 二项堆定义和存储结构；

由根度数不相同的若干二项数组成

1. 二项堆上合并操作及过程；

首先全部按顺序合起来，然后从小到大合起来。

1. Fibnacci 堆优于二项堆的操作有哪些？

合并，插入，最小值，减小关键字

5.二项堆或 Fibonacci 堆在哪些图论算法上有应用？

最小生成树与单源最短路径，最大流算法

**16** 不相交集数据结构**(ch21)**

1. 不相交数据集概念；

MAKE-SET()操作

UNION()操作

FIND-SET()操作

CONNECTED-COMPONENTS(G)操作所有聚合

SAME-COMPONENT()查看是否在同一个集合中

1. 两种实现方式：链表表示和森林表示；

基于链表的合并：

如果采用直接合并，很有可能导致O(n2)的复杂度

一种加权合并的启发式策略，时间复杂度为O(m+nlgn)，这个启发式时间复杂度证明很妙，第一次被更新的话至少有2个成员，第二次的更新至少有4个成员。。。第lgn次更新，至少有n个成员，每个元素如果有lgn次操作，那么就是O(nlgn)。M是除了MAKE-SET的操作其他的操作序列个数。

基于集合森林的合并：

有两种启发式的策略

第一：基于按秩合并的启发式策略。对千每个结点，维护一个 秩，它表示该结点高度的一个上界。在使用按秩合并策略的 UNION 操作中，我们可以让具有较小秩的根指向具有较大秩的根。

做法：为了使用按秩合并的启发式策略实现一个不相交集合森林，我们必须记录下秩的变化情况。 对于每个结点 x, 维护一个整数值 x.ra成，它代表 的高度（从 到某一后代叶结点的最长简单路 径上边的数目）的一个上界。当 凡屯:-SET 创建一个单元素集合时，这个树上的单结点有一个为 的初始秩。每一个 FIND-SET 操作不改变任何秩。 UNION 操作有两种情况，取决千两棵树的根是 否有相同的秩。如果根没有相同的秩，就让较大秩的根成为较小秩的根的父结点，但秩本身保持不 变。另一种情况是两个根有相同的秩时，任意选择两个根中的一个作为父结点，并使它的秩加1. O(mlgn)时间复杂度，证明？

第二：基于路径压缩。使用这种策略可以使查找路径中的每个结点直接指向根。路径压缩并不 改变任何结点的秩。

3.两种表示具体实现和其上操作的时间复杂性；

4.不相交集数据结构应用（在哪些图论算法上有应用）。

　ｐｒｉｍ算法以及无向图的连通分量

**17** 图论算法**(ch22-ch25)**

1. BFS 和 DFS 算法：

- 白色、灰色和黑色结点概念和作用；

BFS：白结点代表尚未接触的结点，灰色的结点表示为已经接触但只存在于BFS队列中，黑色代表其结点周围的结点已经全部接触（在BFS中为已经出队列）

总时间复杂度为O(V+E)

在广度搜索中的PI属性将图划分成广度搜索树。

DFS：白未见，灰色代表初次看见，黑色表示邻接表已经全部扫描变成黑色。　　O(V+E)时间复杂度，其中引入了时间戳概念，其中白灰之间为一个时间戳（结点的一个属性），灰黑之间为一个时间戳。

DFS性质：结点v是深度优先森林里结u的后代当且仅当结点 v在结点u为灰色的时间段里被发现。深度优先搜索的另一个重要性质是，结点的发现时间和完成时间具有所谓的括号化结构（其实很好理解，括号化定理）。、

白色路径定理） 在有向或无向图 G= (V,E) 的深度优先森林中，结点u是结点v的后代当且仅当在发现结点u的时间 u.d, 存在一条从结点u到结点v的全部由白色结点所构成的路径

树边，后向边，前向边，横向边的定义。无向图只有树边和后向边。（各种性质还是要看一眼先）

2.最小生成树：

- 安全边概念和一般算法（Generic algorithm）；

- Kruskal 算法和 Prim 算法的计算过程和计算复杂性；

Kruskal算法普通复杂度为：O(VE),斐波那契堆进行优化为O(E+VlogV),二叉堆维护则为：O(ElogV)

Prim普通：利用堆进行维护，复杂度为O(ElgE)

- 两种贪心算法的贪心策略和贪心选择性质。

#

3.单源最短路径（略）：

- 单源最短路径*δ* (s, v)和短路径上界 d[v]概念。

- 边松弛技术及其一些性质。

- 三种问题算法的计算过程及其时间复杂度： Bellman-Ford 算法、DAG 算法和

Dijkstra 算法。

Bellman-Ford 更新V-1次边松弛技术，最后一次如果有变化则表明FAULSE，时间复杂度O(VE)

Dijkstra 边权值要求全部为非负，更新时间复杂度在O(E+V)

DAG算法，边值可以为负值，但是可以进行拓扑排序，即有向图没有圈，此时直接从拓扑排序进行松弛操作，O(V+E)

4. 所有点对最短路径（略）：

- 为什么能转换为矩阵乘法？

普通算法O(n4)

- 基于矩阵乘法的较慢和快速算法的时间复杂度；

改进算法仍然需要计算矩阵乘法但是次数达到的是O(lgn)级别

- Floyd-Warshall Algorithm 的思路和时间复杂度；

算法复杂度是O(n3)，需要知道算法运行步骤。

- Johnson Algorithm 适应的问题及其时间复杂度。

边值可以取负权值，但是不能有负权值的圈，时间复杂度为：

O(V2lgV+VE)

**18 6** 数论算法**(ch31)**

1. gcd(a, b)及其表示成 a, b 线性组合方法；

其实就是O(lgb)的时间复杂度

1. Euclid’sAlg. 的运行时间；

(Lame 定理） 对任意整数 k~l, 如果 a>b>=l, b<F(k+1)，则该算法调用次数小于K次数。

1. 线性模方程的求解方法；

有解的条件，以及求解方式：算出来一个特解，再进行下一步计算。

1. 中国余数定理及其相应线性同余方程组的求解；

好解

1. RSA 算法过程及正确性基础；

选择的数需要与(q-1)(p-1)互素

6.简单素数测试算法和伪素数测试算法；

简单素数测试方法为试除法，时间复杂度在O(sqrt(n))

伪素数的测试方法为：a^(n-1) = 1 (mod n)，基于a的伪素数。

**卡迈克尔数**的定义是对于合数n，如果对于所有与n互质的正整数b，都有同余式b^ (n-1)≡1 (modn)成立，则称合数n为Carmichael数。

7.MR 算法的改进措施和算法复杂性。

每次考虑其开根号后的剩余，如果是+1，则继续，否则-1停止换数字，如果非平方则为合数

**19** 串匹配**(ch32)**

1. 朴素的串匹配算法及其时间复杂度；
2. Rabin-Karp 串匹配算法及其时间复杂度；

实际上就是采用散列表操作，需要模数，所以可能存在伪散列数，此时需要O(m)再次进行检测，所以最差情况是O(m(n-m+1)+m)

平均情况大概是O(m+n)

1. 有限自动机串匹配算法及其时间复杂度；

预处理建造出自动机，普通算法需要O(m3 Σ)

改进后：这需要 O(m Σ) 的预处理时间和 O(n) 的匹配时间。

4.KMP 串匹配算法及其时间复杂度。

O(m+n)时间复杂度，next数组

**20** 模型和 **NPC(ch34)**

1.算法的严格定义；

算法可解代表在图灵机上可解。

被图灵机在上面严格定义。

1. 几种计算模型的语言识别能力；

有限状态自动机：识别正则文法生成的语言

下推自动机：识别上下文文法无关生成的语言

线性有界自动机：识别上下文相关的生成语言

图灵机：识别短语结构文法生成的所有语言

1. 两类图灵机模型；

确定型，非确定型

固定输出，有随机

5． P 问题、 NP 问题和 NP 完全问题的定义及 P 归约。

P: 确定型 多项式 判定

NP：确定型 多项式 验证

非确定型 多项式 可解

NP完全（NPC）：全部NP可以规约成NPC

**21** 随机算法**(Sch3**，略**)**

(不考的)

1.随机算法的定义；

2.随机算法的分类： Las Vegas 和 Monte Carlo；

(1)QuickSort 属于 Las Vegas 类型； (2)MinCut 属于 Monte Carlo 类型。

3.一般设计风范(略)

4.算法设计：

(1)随机取样； (2)随机串匹配；

(3)格点逼近。