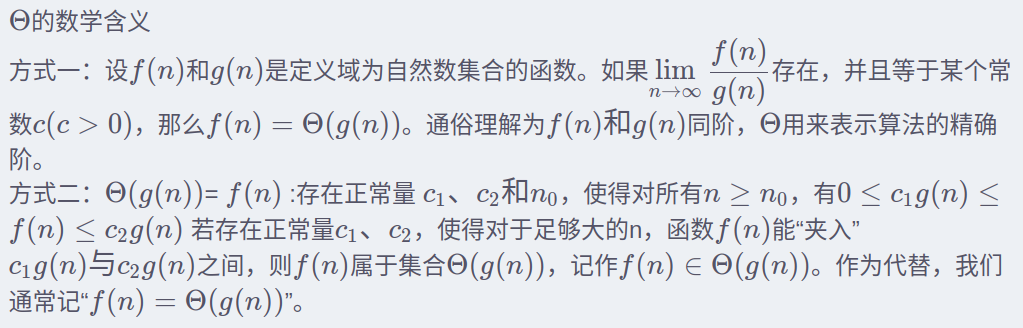
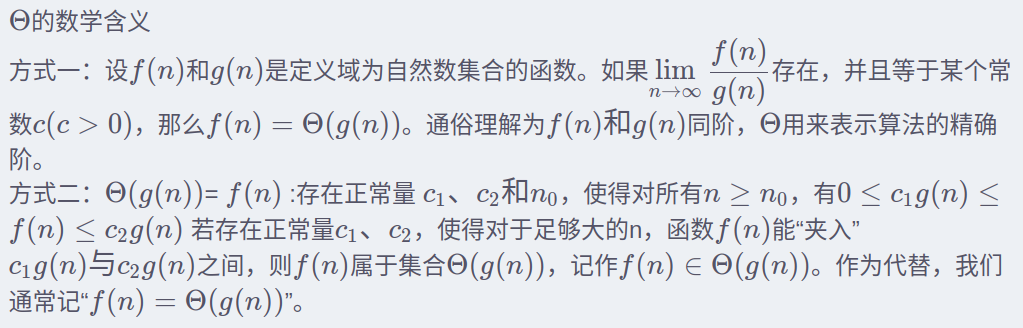
1. 算法基础
2. 插入排序（第六章）
3. 分析算法（概念）
4. 随机访问机（Random-Access Machine, RAM）：一种单处理器计算模型，没有并发操作；
5. RAM模型指令：算术指令（加、减、乘、除、取余、向上/下取整）；数据移动指令（装入、存储、复制）；控制指令（条件/无条件转移、子程序调用与返回）；未列出指令（RAM模型灰色区域，视情况而定）；
6. 输入规模：依赖于研究问题；排序或计算离散傅里叶变换规模是输入的项数；两个整数相乘规模是表示数字的二进制位数；算法的输入是图则规模需要两个参数（顶点数和边数）；
7. 运行时间：执行的基本操作数或步数，假定执行每行伪代码需要常量时间；给定输入规模，一个算法的运行时间也可能依赖于输入，如最佳情况和最坏情况；运行时间对给定的输入是固定的（随机算法除外）；
8. 最坏情况与平均情况：我们往往只研究最坏情况运行时间；算法最坏情况运行时间给出任何输入的运行时间的上界（，最好情况的运行时间给出任何输入运行时间的下界）；平均情况一般与最坏情况相同；
9. 增长量级：只对增长率或增长量级感兴趣，所以①只考虑公式中最重要的项，因为n很大时低阶项不重要；②忽略最重要项的系数，因为对大输入常量因子不如增长率重要；
10. 设计算法——分治法（第四章）和归并排序（第六章）
11. 函数的增长
12. 渐进记号
13. 渐进记号，如，表示一个集合,函数，通常记为

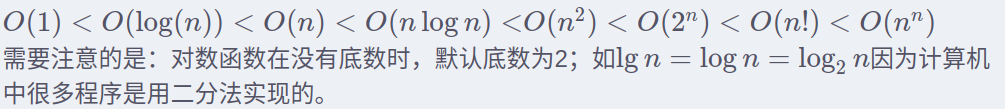
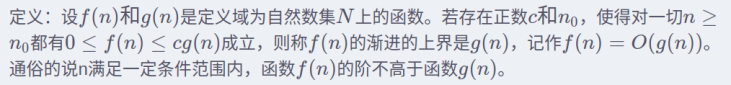
①渐近紧确界记号： Θ（big-theta）

对所有时，函数乘一个常量因子可等于，我们称是的一个渐近紧确界（asymptotically tight bound）；的定义要求每个成员均渐近非负，即当足够大时，非负;渐近正函数就是对所有足够大的n均为正的函数;



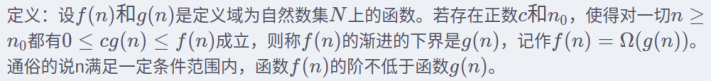
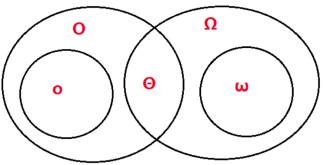
②渐近上界记号：O(big-oh)

根据符号的定义，用它评估算法的复杂度得到的只是问题规模充分大时的一个上界;这个上界的阶越低，评估越精确，越有价值;渐近上界包括渐进紧确的上界和非渐进紧确的上界；



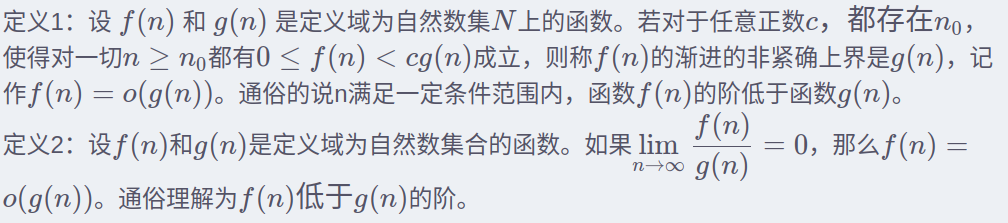
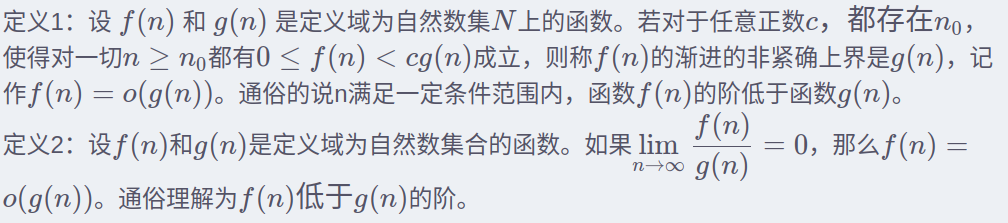
③渐近下界记号：(big-omege)

当我们说运行时间为时，指存在一个属于的函数，使得对的任意值（不论什么规模，哪一种输入），其运行时间的上界都是；插入排序的运行时间介于和；插入排序的最坏情况运行时间为或；

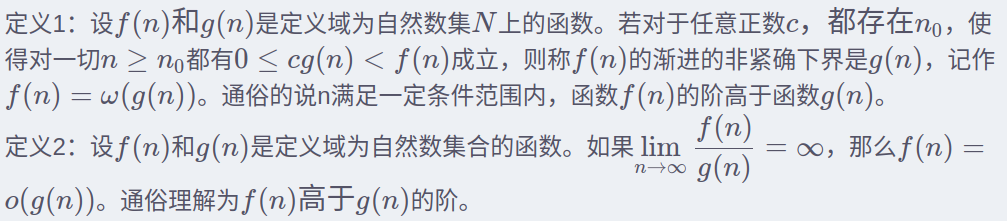
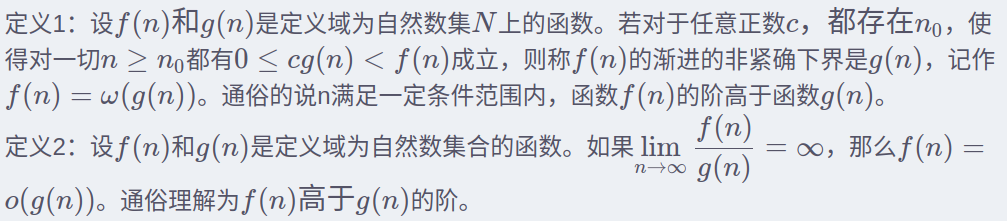


④非渐近紧确上界：o(小-oh)

是渐近紧确的，而是非紧确上界;



⑤非渐近紧确下界：ω(小-omege)



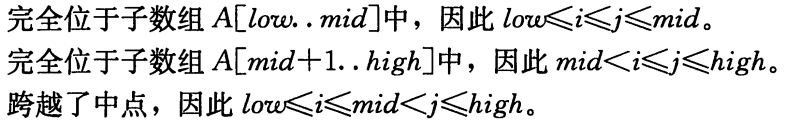
1. 等式和不等式中的渐进记号

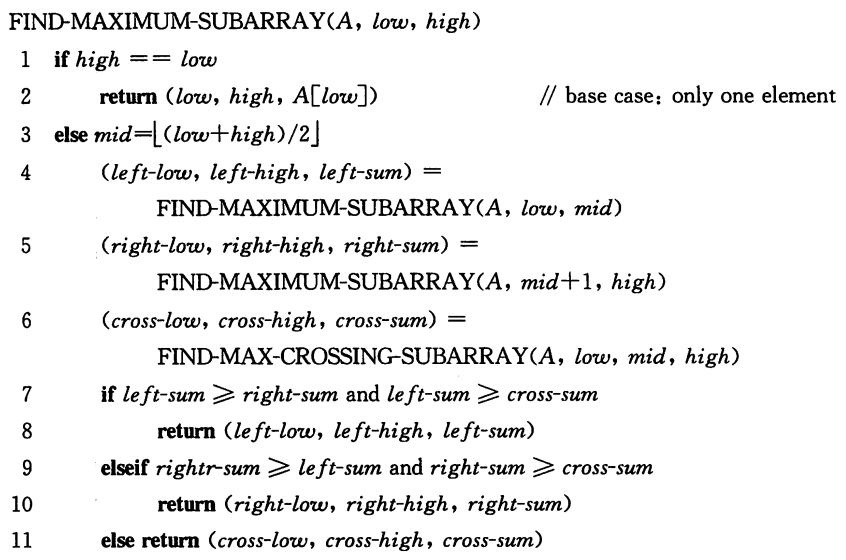
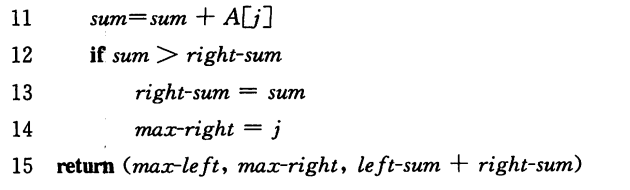
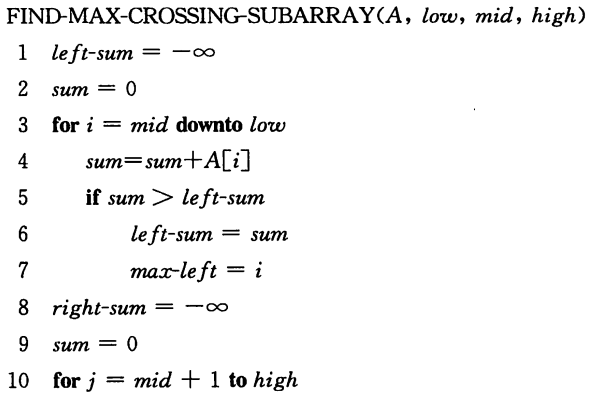
渐进记号在公式中表示不关注名称的匿名函数；渐进记号出现在等式左边表示任意，出现在等式右边表存在；对于任意等号左边的匿名函数，总存在一个等号右边的匿名函数使得等式成立；

1. 各种性质

①传递性；②自反性；③对称性；④转置对称性；⑤三分性（不是所有函数都可渐进比较）

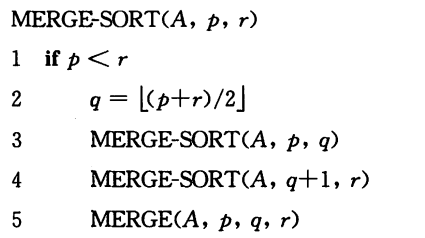
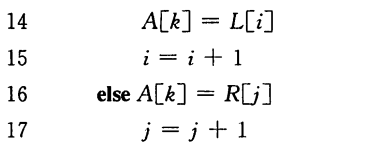
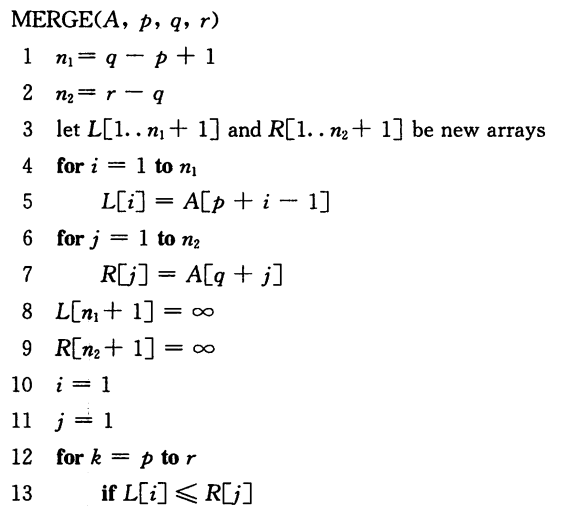
1. 标准记号与常用函数
2. 分治策略
3. 概述
4. 层递归步骤：①分解Divide（将问题划分为一些子问题）；②解决Conquer（递归求解子问题）；③合并Combine（将子问题的解组合成原问题的解）
5. 子问题种类：①递归情况（Recursive Case）：子问题足够大需要递归求解时；②基本情况（Base Case）：子问题小到直接求解；③合并步骤：除与原问题形式相同规模更小的子问题外，需要求解的与原问题不同的子问题，这类子问题求解视作合并的一部分；
6. 递归式（Recurrence）：一个等式或不等式，通过更小输入上的函数值来描述一个函数；通常忽略边界条件（向上/下取整）；
7. 求解递归式或渐进界的方法：①代入法；②递归树法；③主方法；
8. 不等式递归式：如，仅描述了的上界，因此可以用大符号来表示解；，仅描述了的下界，因此可以用大符号来表示解；
9. 最大子数组问题
10. 问题描述：某时刻可以买入一支股票并在之后某时刻卖出，目标时最大化收益；
11. 暴力解法：尝试每对买进卖出日期组合，只要卖出在买入之后；则n天共有种日期组合；处理每对日期的开销为常量，因此运行时间为；
12. 问题变换：我们的目的是寻找一段日期，使得第一天到最后一天的价格差最大；我们将输入数据从每日价格变为每日价格变化；第i天的价格变化定义为第i天和第i-1天的价格差，称为数组A；此时问题转换为寻找A的和最大的非空连续子数组，称为最大子数组（Maximum Subarray）；只求一个最大子数组；只有当数组中包含负数时，最大子数组问题才有意义；
13. 分治法： 的任何连续子数组A[i .. j]所处的情况是以下三种情况之一；的最大子数组必然是 、 、跨越中点所有子数组中和最大者； 和仍是最大子数组问题，可以递归求解；跨越中点所有子数组并非原问题更小规模实例，因为加入了子数组必须跨越中点的限制，我们需要找出和然后将其合并；如果包含n个元素，则调用花费时间，for循环每次迭代，3-7行次，10-14行次，共计+1次；递归过程中，6行子问题并非原问题更小规模实例，因此视为合并部分。



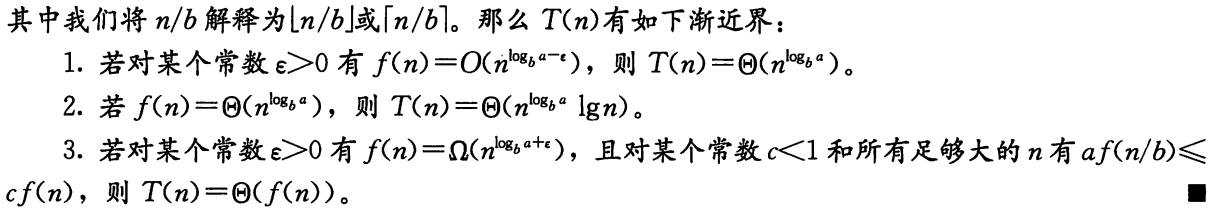
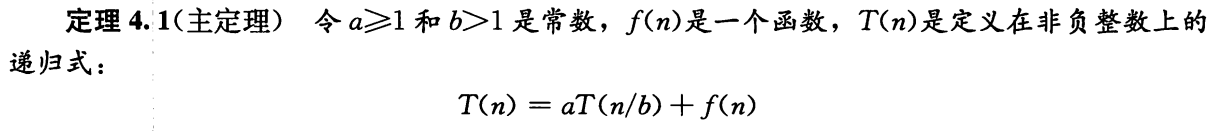


合并

1. 算法分析：假设原问题规模为2的幂，这样所有子问题规模均为整数；用表示FIND-MAXIMUM-SUBARRAY求解n个元素的最大子数组的运行时间；n=1时；当n>1时为递归情况，第1、3行为常量时间，第4、5行为规模更小实例用时，第6行调用FIND-MAX-CROSSING-SUBARRAY花费时间，第7-11行仅花费时间，因此有(n)+=(n)；最大子数组问题实际存在一个线性时间算法。
2. 归并排序
3. 问题描述：见第6章
4. 分治法：归并算法的关键是合并，调用MERGE(A,p,q,r)来完成合并；其中A是一个数组p、q、r是数组下标；认为A[p..q]和A[q+1..r]都已经排好序；需要时间；每个堆底部放一张哨兵牌；



1. 矩阵乘法的Strassen算法
2. 用代入法求解递归式
3. 用递归树方法求解递归式
4. 用主方法求解递归式
5. 主定理：



1. 主方法：①；②其中b=3/2；③由于，其中约等于0.2；④，其中不存在；
2. 证明主定理
4. 堆排序
5. 堆
6. 维护堆的性质
7. 建堆
8. 堆排序算法
9. 优先队列
10. 快速排序
11. 快速排序的描述
12. 快速排序的性能
13. 快速排序的随机化版本
14. 快速排序分析 （略）
15. 线性时间排序
16. 排序算法的下界
17. 计数排序
18. 基数排序
19. 桶排序 （略）
20. 中位数和顺序统计量
21. 最小值和最大值
22. 期望为线性时间的选择算法
23. 最坏情况为线性时间的选择算法