算法分析与设计

屈 婉 玲

qwl@pku.edu.cn

计算思维与人才培养

- 2006年3月周以真(Jeannette M. Wing，卡内基·梅隆大学计算机系系主任)首次提出Coputational Thinking的概念：运用计算机科学的基础概念去求解问题、设计系统和理解人类的行为，它包括了涵盖计算机科学之广度的一系列思维活动。

数学思维与工程思维的互补与融合：抽象与实现

编程 ⇒ 评价程序 优化程序 ⇒ 问题分析 方法确定

技能：会做　　能力：做得好　　思维：认知、方法论

三种基本的思维

- 三种思维的共同特点：

  用语言文字表达、有语法与语义规则、推理逻辑

	实验思维	理论思维	计算思维
起源	物理学	数学	计算机科学
过程步骤	1.实验观察归纳建立简单数学公式 2.导出数量关系 3.实验验证	1.定义概念 2.提出定理 3.给出证明	1.建模(约简、嵌入、转化、仿真、...） 2.抽象与分解,控制系统复杂性 3.自动化实现...
特点	解释以往现象 无矛盾 预见新的现象	公理集 可靠协调推演规则 正确性依赖于公理	结论表示有限性 语义确定性 实现机械性

北京大学

算法与计算思维

- 算法课程是训练计算思维的重要课程；涉及到对问题的抽象，建模，设计好的求解方法，复杂性的控制，...
- 可计算性与计算复杂性： 形式化、确定性、有限性，抽象与逻辑证明
- 算法设计与分析：抽象建模、归纳、正确性证明、效率分析、…

课程简介

课程名称：

算法分析与设计

Design and Analysis of Algorithms

课号：

0A002

基本目的：

- 掌握组合算法设计的基本技术
- 掌握算法分析的基本方法
- 了解计算复杂性理论的基本概念及其应用

课程主要内容

近似算法 | NP 完全理论简介 | 随机算法	问题处理策略 计算复杂性理论
算法分析与问题的计算复杂性	算法分析方法
分治策略 | 动态规划 | 贪心算法 | 回溯与分支限界	算法设计技术
数学基础、数据结构	基础知识

教 材

书名：

《算法设计与分析》

作者：

屈婉玲, 刘田, 张立昂, 王捍贫

出版社：

清华大学出版社

出版时间：

2011，2013重印

参考书

1. Jon Kleinberg, Eva Tardos, Algorithm Design, Addison-Wesley, 清华大学出版社影印版，2006.
2. Thomas H. Cormen, Charles E. Leiserson, Ronald L.Rivest, Introduction to Algorithms(Second Edtion), The MIT Press 2009.
3. 张立昂，可计算性与计算复杂性导引（第**3**版），北京大学出版社，2011.

4*. 堵丁柱，葛可一，王洁，计算复杂性导论，高教出版社 2002.

5*. Sanjeev Arora, Boaz Barak, Computational Complexity: A Modern Approach, Cambridge University Press，2009.

北京大学

学习安排

教学方式：

以课上讲授为主

视频答疑

书面作业

成绩评定：

平时成绩：50%

期末笔试：50%

视频答疑

http://vclassroom.pku.edu.cn/qwl

定期安排时间

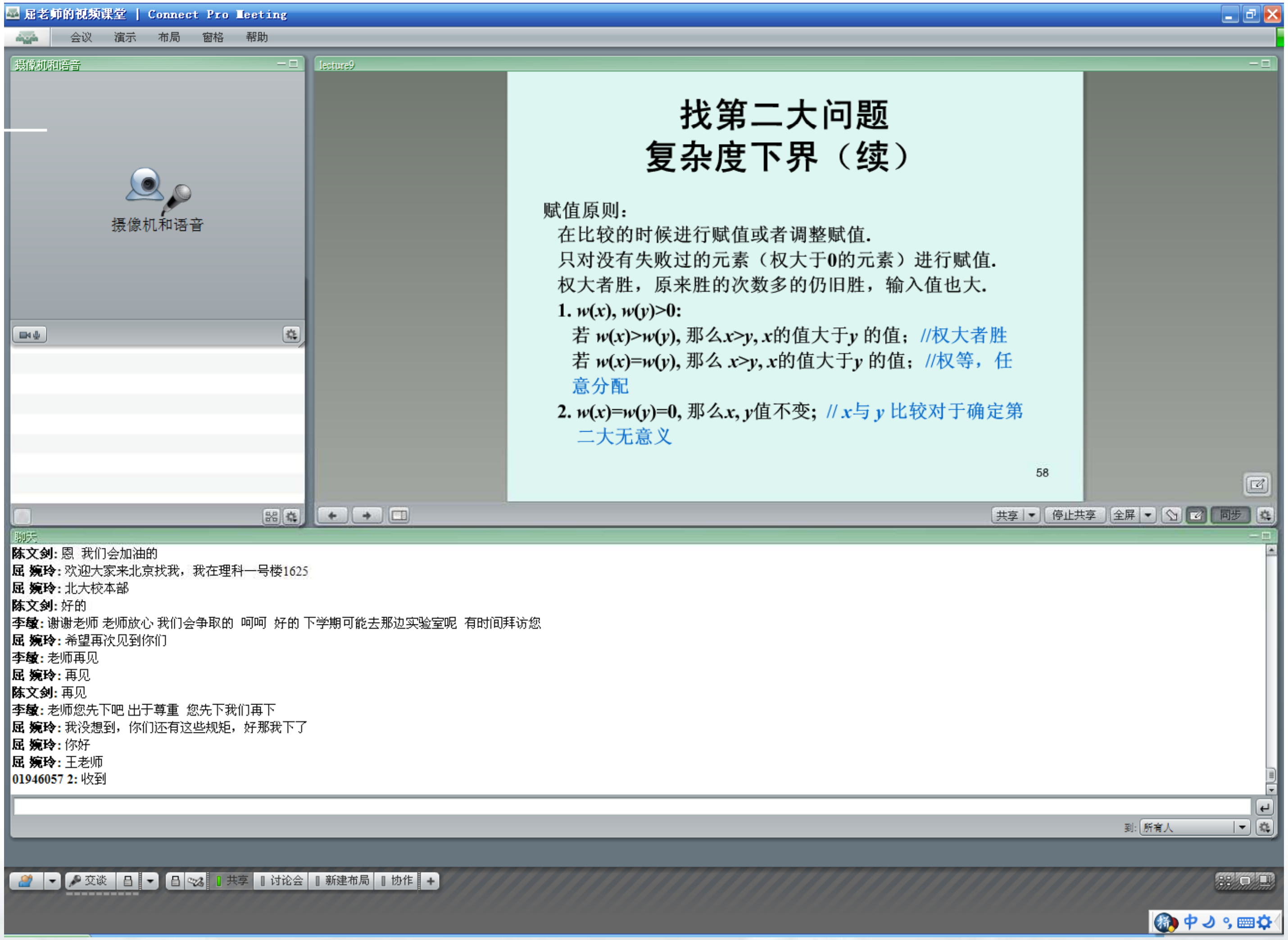
以客人身份用真实姓名进入

在线答疑：

- 语音
- 打字
- 白板
- 上传文件
- **PPT**播放

北京大学

视频教室界面


屈老师的视频课堂 | Connect Pro Meeting
会议 演示 布局 窗格 帮助
摄像机和语音
lecture9
找第二大问题
复杂度下界（续）
赋值原则：
在比较的时候进行赋值或者调整赋值.
只对没有失败过的元素（权大于0的元素）进行赋值.
权大者胜，原来胜的次数多的仍旧胜，输入值也大.
1. w(x), w(y)>0:
若 w(x)>w(y), 那么x>y, x的值大于y 的值；//权大者胜
若 w(x)=w(y), 那么 x>y, x的值大于y 的值；//权等，任意分配
2. w(x)=w(y)=0, 那么x, y值不变；// x与 y 比较对于确定第二大无意义
58
共享 停止共享 全屏 同步
聊天
陈文剑: 恩 我们会加油的
屈 婉玲: 欢迎大家来北京找我，我在理科一号楼1625
屈 婉玲: 北大校本部
陈文剑: 好的
李敏: 谢谢老师 老师放心 我们会争取的 呵呵 好的 下学期可能去那边实验室呢 有时间拜访您
屈 婉玲: 希望再次见到你们
李敏: 老师再见
屈 婉玲: 再见
陈文剑: 再见
李敏: 老师您先下吧 出于尊重 您先下我们再下
屈 婉玲: 我没想到，你们还有这些规矩，好那我下了
屈 婉玲: 你好
屈 婉玲: 王老师
01946057 2: 收到
到: 所有人
交谈 共享 讨论会 新建布局 协作

引言 为什么要学算法

- 几个例子
  - 调度问题
  - 排序问题
  - 货郎问题
- 算法研究的内容和目标
  - 算法设计技术
  - 算法分析方法
  - 问题复杂度的界定
  - 计算复杂性理论
- 算法研究的重要性

几个例子

例1：求解调度问题

任务集 $S=\{1, 2, \ldots, n\}$，

第 j 项加工时间：$t_j, \in Z^+, j=1, 2, \ldots, n$

一个可行调度方案：$1, 2, \ldots, n$ 的排列 $i_1, i_2, \ldots, i_n$

求：总等待时间最少的调度方案，即求

S的排列$i_1, i_2, \ldots, i_n$使得 $\min\{\sum_{k=1}^{n}(n-k+1)t_{i_k}\}$

求解方法

贪心策略：加工时间短的先做

如何描述这个方法？这个方法是否对所有的实例都能得到最优解？如何证明？这个方法的效率如何？

例2：投资问题

问题：

m元钱，投资给n个项目，效益函数 $f_i(x)$，表示第 i 个项目投入x元钱的效益，i=1,2,…,n. 如何分配每个项目的钱数使得总效益最大？

令x_i 是第 i 个项目的钱数

$$\max \sum_{i=1}^{n} f_i(x_i),$$

$$\sum_{i=1}^{n} x_i = m, \quad x_i \in \mathrm{N}$$

采用什么算法？效率怎样？

蛮力算法的代价

Stirling公式： $n! = \sqrt{2\pi n}\,(\frac{n}{e})^n (1 + \Theta(\frac{1}{n}))$

非负整数解 $<x_1, x_2, \ldots, x_n>$ 的个数估计：

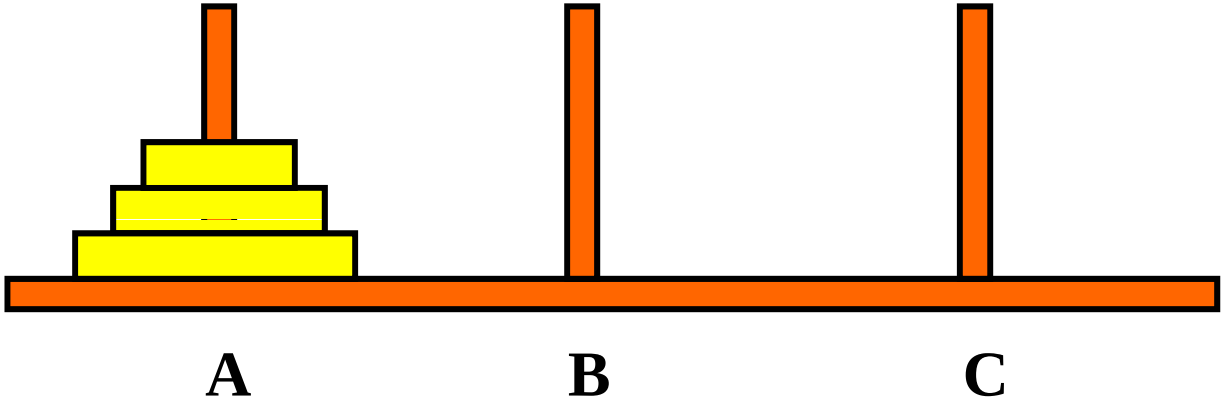
$$C_{m+n-1}^{m} = \frac{(m+n-1)!}{m!(n-1)!} = \Omega((1+\varepsilon)^{m+n-1})$$

蛮力算法——穷举法代价太大

能否利用解之间的依赖关系找到更好的算法？

结论：需要算法设计技术

例3 Hanoi塔问题



$T(n) = 2\,T(n-1) + 1$，$T(1) = 1$，解得 $T(n) = 2^n - 1$

1秒移1个，64个盘子要多少时间？（5000亿年），千万亿次/秒，4个多小时.

思考：是否存在更好的解法?

Reve难题：Hanoi塔变种，柱数增加，允许盘子相等.

其他变种：奇偶盘号分别放置

例4 排序算法的评价

已有的排序算法：考察元素比较次数

插入排序、冒泡排序：最坏和平均状况下都为$O(n^2)$

快速排序：最坏状况为$O(n^2)$，平均状况下为$O(n\log n)$

堆排序、二分归并排序：最坏和平均状况下都为$O(n\log n)$

…

问题

那个排序算法效率最高？

是否可以找到更好的算法？排序问题的计算难度如何估计？

例5 货郎问题

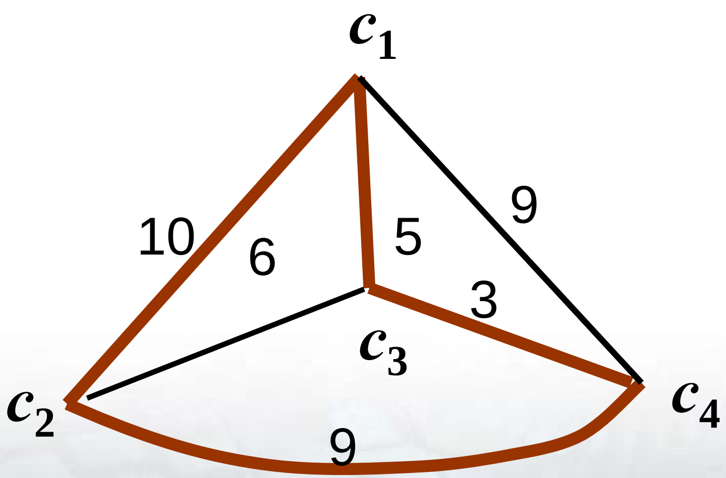
货郎问题：

有穷个城市的集合 $C = \{ c_1, c_2, \ldots, c_m \}$, 距离

$$d(c_i,c_j) = d(c_j,c_i) \in Z^+, \quad 1 \le i < j \le m$$

求 $1, 2 \ldots, m$ 的排列 $k_1, k_2, \ldots, k_m$ 使得

$$\min\{\sum_{i=1}^{m-1} d(c_{k_i}, c_{k_{i+1}}) + d(c_{k_m}, c_{k_1})\}$$



现状：至今没有找到有效的算法，
存在大量问题与它难度等价

问题：是否存在有效算法？
如何处理这类问题？

Algorithm + Data Structure = Programming

好的算法

- 提高求解问题的效率
- 节省存储空间

需要解决的问题

问题→寻找求解算法	算法设计技术
算法→算法的评价	算法分析方法
算法类→问题复杂度的评价	问题复杂性分析
问题类→能够求解的边界	计算复杂性理论

理论上的可计算：
可计算性理论

研究目标

确定什么问题是可计算的，即存在求解算法

合理的计算模型

已有的：递归函数、**Turing**机、λ演算、**Post**系统等

条件：计算一个函数只要有限条指令

每条指令可以由模型中的有限个计算步骤完成

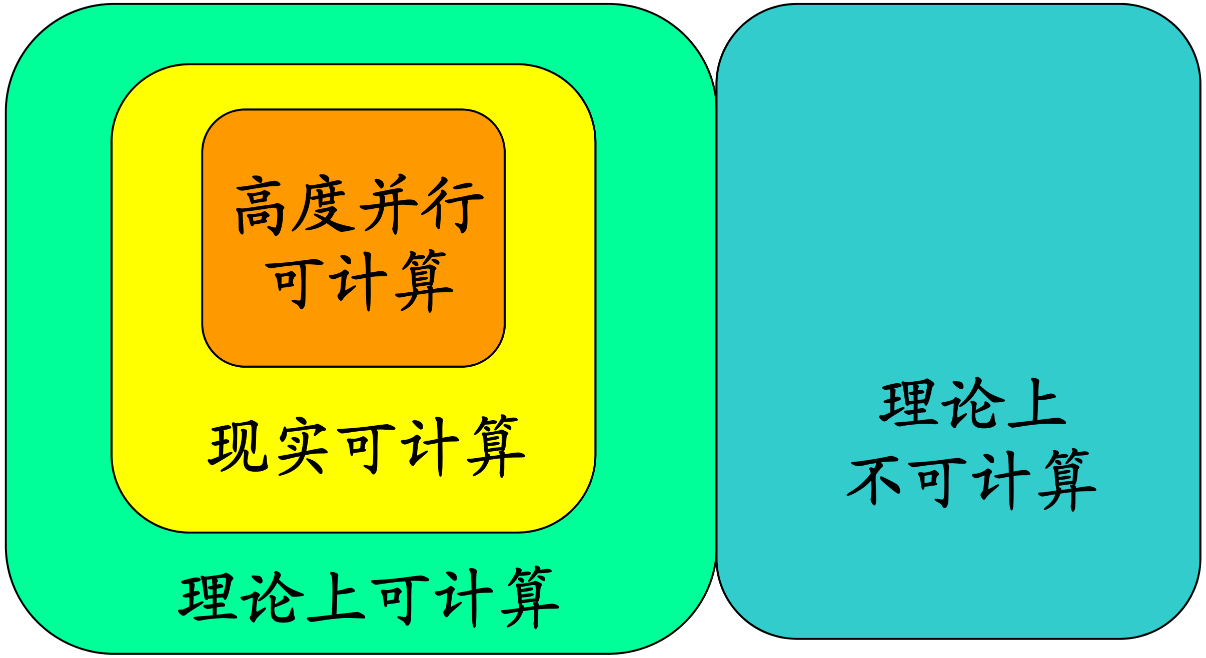
指令执行的过程是确定的

核心论题：**Church-Turing论题**

如果一个函数在某个合理的计算模型上可计算，那么它在**Turing**机上也是可计算的

可计算性是不依赖于计算模型的客观性质

理论上与现实上可计算性



算法至少具有指数时间：理论上可计算——难解的

多项式时间的算法：现实上可计算——多项式时间可解的

对数多项式时间的算法：高度并行可解的

多项式时间的算法

多项式时间的算法

时间复杂度函数为$O(p(n))$的算法，其中$p(n)$是n的多项式

不是多项式时间的算法

不存在多项式 $p(n)$使得该算法的时间复杂度为 $O(p(n))$

包含指数时间甚至更高阶的算法

多项式函数与指数函数

时间复杂度函数	问题规模					
	10	20	30	40	50	60
n	10^{-5}	$2*10^{-5}$	$3*10^{-5}$	$4*10^{-5}$	$5*10^{-5}$	$6*10^{-5}$
n^2	10^{-4}	$4*10^{-4}$	$9*10^{-4}$	$16*10^{-4}$	$25*10^{-4}$	$36*10^{-4}$
n^3	10^{-3}	$8*10^{-3}$	$27*10^{-3}$	$64*10^{-3}$	$125*10^{-3}$	$216*10^{-3}$
n^5	10^{-1}	3.2	24.3	1.7 分	5.2 分	13.0 分
2^n	.001 秒	1.0 秒	17.9 分	12.7 天	35.7 年	366 世纪
3^n	.059 秒	58 分	6.5 年	3855 世纪	$2*10^8$ 世纪	$1.3*10^{13}$ 世纪

多项式函数与指数函数

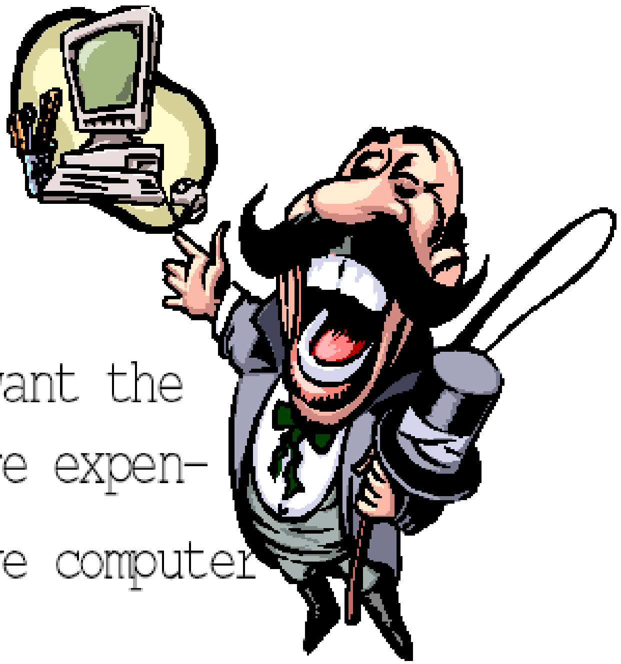
时间复杂度函数	1小时可解的问题实例的最大规模		
	计算机	快100倍的计算机	快1000倍的计算机
n	N_1	$100\ N_1$	$1000\ N_1$
n^2	N_2	$10\ N_2$	$31.6\ N_2$
n^3	N_3	$4.64\ N_3$	$10\ N_3$
n^5	N_4	$2.5\ N_4$	$3.98\ N_4$
2^n	N_5	$N_5 + 6.64$	$N_5 + 9.97$
3^n	N_6	$N_6 + 4.19$	$N_6 + 6.29$

北京大学

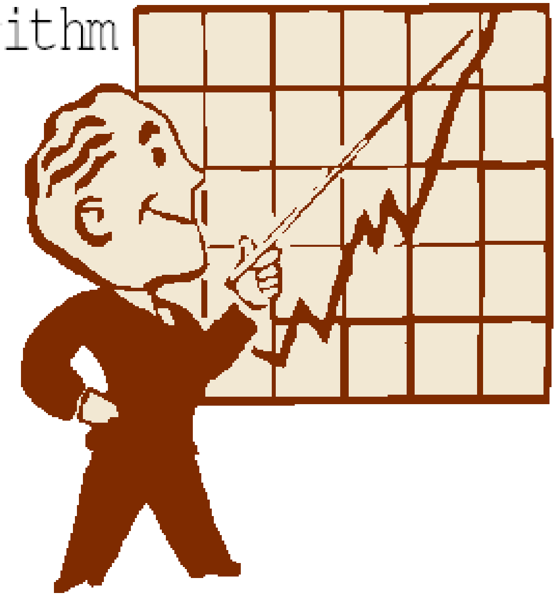
10亿次/秒机器求解的问题

- 快速排序算法给10万个数据排序, 运算量约为 $10^5 \times \log_2 10^5 \approx 1.7\times 10^6$, 仅需$1.7\times 10^6/10^9=1.7\times 10^{-3}$秒.
- Dijkstra算法求解1万个顶点的图的单源最短路径问题, 运算量约为$(10^4)^2=10^8$, 约需$10^8/10^9=0.1$秒.
- 回溯法解100个顶点的图的最大团问题, 运算量为 $100\times 2^{100}\approx 1.8\times 10^{32}$, 需要$1.8\times 10^{32}/10^9=1.8\times 10^{21}$秒 $=5.7\times 10^{15}$年, 即5千7百万亿年!
- 1 分钟能解多大的问题. 1分钟60秒, 这台计算机可做用快速排序算法可给2×10^9(即, 20亿)个数据排序, 用Dijkstra算法可解2.4×10^5个顶点的图的单源最短路径问题. 而用回溯法一天只能解41个顶点的图的最大团问题

两种选择



rich man



smart mathematician

问题的复杂度分析

多项式时间可解的问题 ***P***

存在着解***P*** 的多项式时间的算法

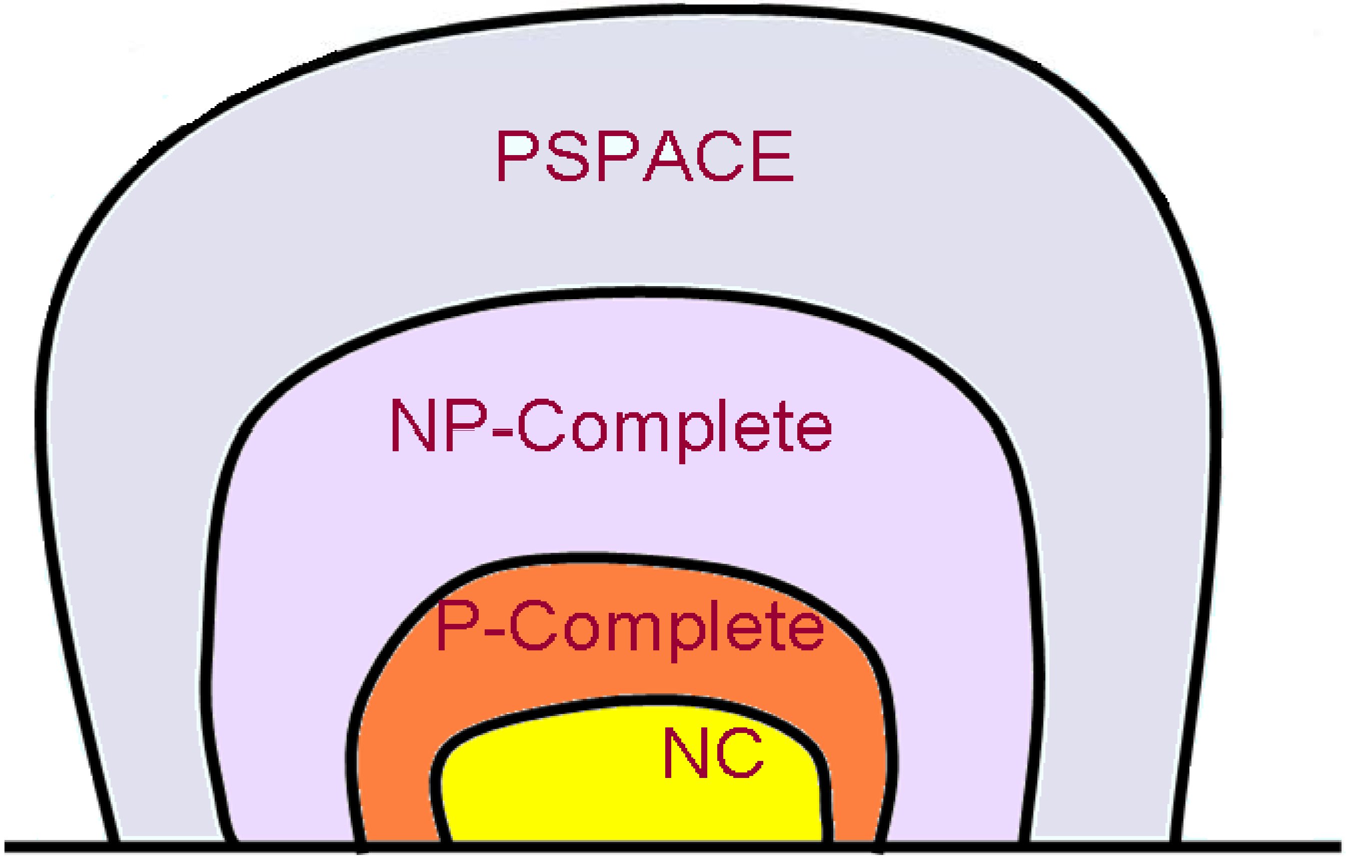
难解的问题***P***

不存在解***P*** 的多项式时间的算法

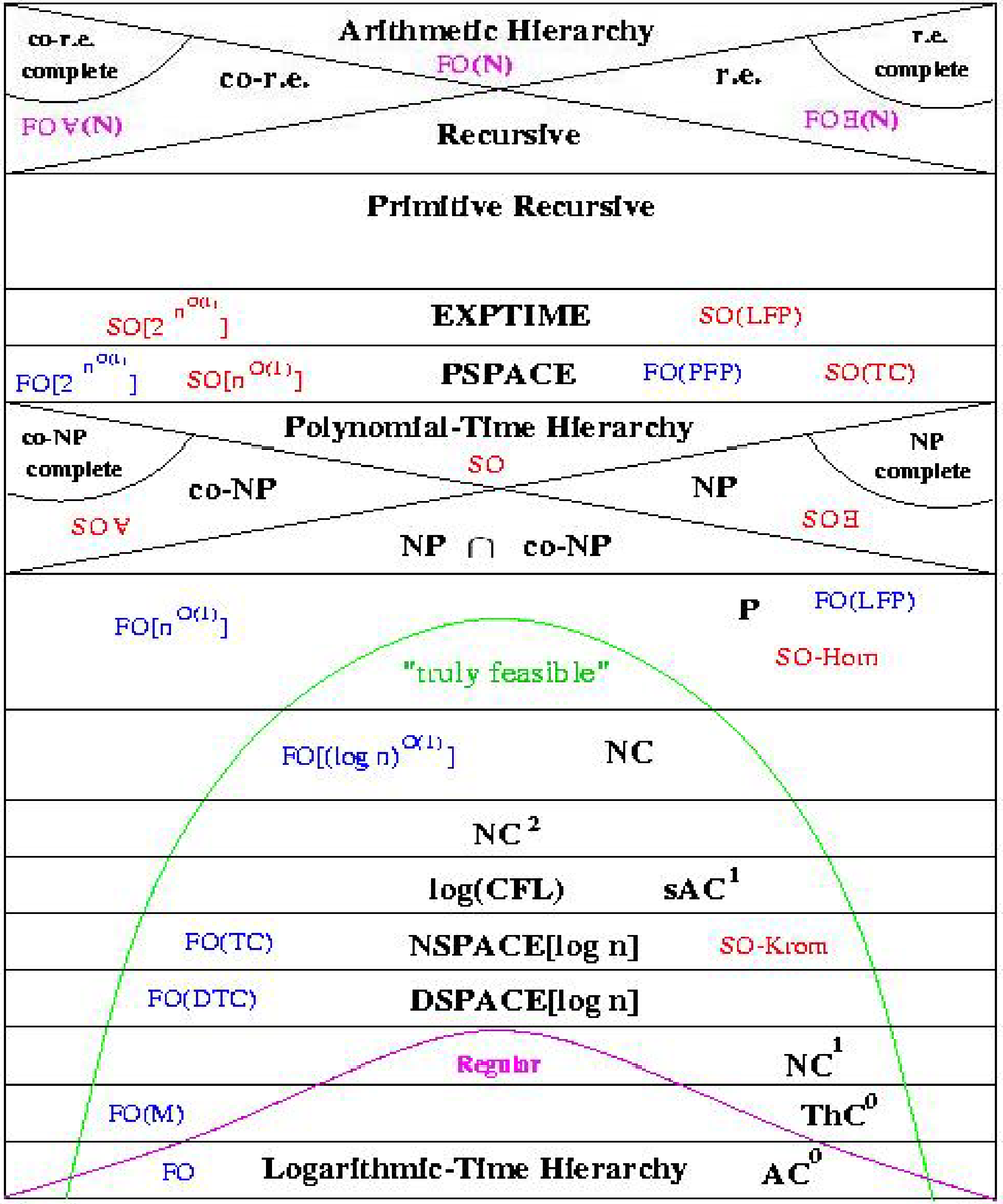
实际上可计算的问题

多项式时间可解的问题

计算复杂性类的谱系



计算复杂性类的谱系



Arithmetic Hierarchy
co-r.e. complete
co-r.e.
FO(N)
r.e.
r.e. complete
FO∀(N)
FO∃(N)
Recursive
Primitive Recursive
$SO[2^{n^{O(1)}}]$
EXPTIME
SO(LFP)
$FO[2^{n^{O(1)}}]$
$SO[n^{O(1)}]$
PSPACE
FO(PFP)
SO(TC)
Polynomial-Time Hierarchy
co-NP complete
SO
NP complete
co-NP
NP
SO∀
SO∃
NP ∩ co-NP
$FO[n^{O(1)}]$
P
FO(LFP)
"truly feasible"
SO-Horn
$FO[(\log n)^{O(1)}]$
NC
NC^2
log(CFL)
sAC^1
FO(TC)
NSPACE[log n]
SO-Krom
FO(DTC)
DSPACE[log n]
Regular
NC^1
FO(M)
ThC^0
FO
Logarithmic-Time Hierarchy
AC^0

算法与计算复杂性理论进展

算法
- 概率算法
- 近似算法
- 在线算法
- 分布式算法

计算复杂性
- 概率Turing机与概率复杂性
- 近似求解的复杂性
- 参数复杂性
- 计数复杂性
- 通信复杂性

算法研究的重要性

- 算法设计与分析技术在计算机科学与技术领域有着重要的应用背景
- 算法设计分析与计算复杂性理论的研究是计算机科学技术的核心研究领域

  1966-2005期间，**Turing**奖获奖**50**人，其中**10**人 以算法设计，**7**人以计算理论、自动机和复杂性研究领域的杰出贡献获奖

  计算复杂性理论的核心课题 “**P=NP?**” 是本世纪 **7**个最重要的数学问题之一

- 通过算法设计与分析课程的训练对提高学生的素质和分析问题解决问题的能力以及计算思维有着重要的作用

第1章 基础知识

1.1 算法的基本概念

- 问题
- 算法
- 算法的时间复杂度

1.2 算法的伪码表示

1.3 数学基础知识

- 函数的渐近的界
- 序列求和
- 递推方程求解

1.1算法的基本概念

问题：需要回答的一般性提问，通常含有若干参数，对参数的一组赋值就得到问题的一个实例.

算法：是有限条指令的序列，这个指令序列确定了解决某个问题的一系列运算或操作.

算法A解问题P：把问题P的任何实例作为算法A的输入，A能够在有限步停机，并输出该实例的正确的解.

算法的时间复杂度：针对问题指定基本运算，计数算法所做的基本运算次数

最坏情况下的时间复杂度：算法求解输入规模为n的实例所需要的最长时间$W(n)$.

平均情况下的时间复杂度：在指定输入的概率分布下，算法求解输入规模为n的实例所需要的平均时间 $A(n)$.

检索问题的时间估计

检索问题

输入：非降顺序排列的数组 L，元素数为n； 数x

输出：j. 若x在L中，j 是 x 首次出现的序标；否则 $j=0$

算法： 顺序搜索

最坏情况下时间：$W(n)=n$

平均情况：假设 $x\in L$ 的概率为 p，x 在 L 不同位置等概分布.

实例集 S，实例$I\in S$的概率是 p_I, 算法对I 的基本运算次数为 t_I，平均情况下的时间复杂度为

$$A(n)=\sum_{I\in S} t_I p_I$$

$$A(n)=\sum_{i=1}^{n} i\frac{p}{n}+(1-p)n=\frac{p(n+1)}{2}+(1-p)n$$

1.2 算法的伪码描述

赋值语句：←

分支语句：if …then … [else…]

循环语句：while，for，repeat until

转向语句：goto

输出语句：return

调用：直接写过程的名字

注释：//…

实例：求最大公约数

算法1.1 Euclid(m,n)

输入：非负整数m,n，其中m与n不全为0

输出：m与n的最大公约数

1. while m>0 do
2. $r \leftarrow n$ mod m
3. $n \leftarrow m$
4. $m \leftarrow r$
5. return n

实例：改进的顺序检索

算法1.2 Search(L,x)

输入：数组 $L[1..n]$，其元素按照从小到大排列，数 x.

输出：若 x 在 L中，输出 x 的位置下标 j；否则输出0.

1. $j \leftarrow 1$
2. **while** $j \le n$ **and** $x > L[j]$ **do** $j \leftarrow j+1$
3. **if** $x < L[j]$ **or** $j > n$ **then** $j \leftarrow 0$
4. **return** j

1.3 数学基础

1.3.1 函数的渐近的界

定义1.1 设 f 和g是定义域为自然数集 N上的函数.

(1) 若存在正数 c 和 n_0使得对一切 $n \geq n_0$有$0 \leq f(n) \leq cg(n)$ 成立, 则称 $f(n)$ 的渐近的上界是 $g(n)$，记作

$$f(n) = O(g(n)).$$

(2) 若存在正数 c 和 n_0，使得对一切 $n \geq n_0$有 $0 \leq cg(n) \leq f(n)$ 成立, 则称 $f(n)$的渐近的下界是 $g(n)$，记作

$$f(n) = \Omega(g(n)).$$

(3) 若对于任意正数 c 都存在 n_0，使得当 $n \geq n_0$ 时有 $0 \leq f(n) < cg(n)$ 成立, 则记作

$$f(n) = o(g(n)).$$

函数的渐近的界（续）

(4) 若对于任意正数 c 都存在 n_0，使得当 $n \geq n_0$ 时有 $0 \leq cg(n) < f(n)$成立, 则记作

$$f(n)=\omega(g(n)).$$

(5) 若 $f(n) = O(g(n))$ 且 $f(n) = \Omega(g(n))$, 则记作

$$f(n)=\Theta(g(n)).$$

例 $f(n)=n^2+n$，则

$f(n)=O(n^2)$， $f(n)=O(n^3)$,

$f(n)=o(n^3)$,

$f(n)=\Theta(n^2)$

有关定理

定理1.1 设 f 和g是定义域为自然数集合的函数.

(1) 如果 $\lim_{n\to\infty} f(n)/g(n)$ 存在且等于某个常数c>0, 那么

$$f(n)=\Theta(g(n)).$$

(2) 如果$\lim_{n\to\infty} f(n)/g(n) = 0$，那么

$$f(n)=o(g(n)).$$

(3) 如果 $\lim_{n\to\infty} f(n)/g(n) = +\infty$，那么

$$f(n)=\omega(g(n)).$$

证明定理1.1 (1)

证 根据极限定义，对于给定的正数 ε =c/2, 存在某个n_0，只要$n \geq n_0$， 就有

$$\left|\frac{f(n)}{g(n)} - c\right| < \varepsilon \Rightarrow c - \varepsilon < \frac{f(n)}{g(n)} < c + \varepsilon$$

$$\Rightarrow \frac{c}{2} < \frac{f(n)}{g(n)} < \frac{3c}{2} < 2c$$

对所有的$n \geq n_0$， $f(n) \leq 2cg(n)$. 从而推出 $f(n)=O(g(n))$

对所有的$n \geq n_0$， $f(n) \geq (c/2)g(n)$， 从而推出 $f(n)=\Omega(g(n))$,

于是 $f(n)= \Theta(g(n))$

定理1.2-1.3

定理1.2 设 f, g, h是定义域为自然数集合的函数，

(1) 如果 $f=O(g)$ 且 $g=O(h)$，那么 $f=O(h)$.

(2) 如果 $f=\Omega(g)$ 且 $g=\Omega(h)$，那么 $f=\Omega(h)$.

(3) 如果 $f=\Theta(g)$ 和 $g=\Theta(h)$，那么 $f=\Theta(h)$.

定理1.3 假设 f 和 g是定义域为自然数集合的函数，若对某个其它的函数 h, 我们有 $f=O(h)$ 和 $g=O(h)$，那么

$$f + g = O(h).$$

例：函数的阶

例1 设 $f(n) = n^2 - 3n$， 证明 $f(n) = \Theta(n^2)$.

证 因为

$$\lim_{n\to+\infty}\frac{f(n)}{n^2} = \lim_{n\to+\infty}\frac{\frac{1}{2}n^2 - 3n}{n^2} = \frac{1}{2}$$

根据定理1.1有 $f(n) = \Theta(n^2)$.

可以证明：多项式函数，幂函数的阶低于指数函数

$n^d = o(r^n)$，$r>1$，$d>0$

基本函数类

阶的高低

至少指数级：$2^n, 3^n, n!, \ldots$

多项式级：$n, n^2, n\log n, n^{1/2}, \ldots$

对数多项式级：$\log n, \log^2 n, \ldots$

$$2^{2^n},\quad n!,\quad n2^n,\quad (\log n)^{\log n} = \Theta(n^{\log\log n}),$$

$$n^3,\quad \log(n!) = \Theta(n\log n),\quad n$$

$$\log n,\quad \sqrt{\log n},\quad \log\log n,$$

$$n^{1/\log n} = \Theta(1)$$

对数函数

符号：

$$\log n = \log_2 n, \quad (\lg n = \log_2 n)$$

$$\log^k n = (\log n)^k$$

$$\log\log n = \log(\log n)$$

性质：

$$\log_b n = o(n^\alpha) \quad \alpha > 0$$

$$a^{\log_b n} = n^{\log_b a}$$

$$\log_k n = \Theta(\log_l n)$$

阶乘

Stirling公式
$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1+\Theta\left(\frac{1}{n}\right)\right)$$

$n! = o(n^n)$， $n! = \Omega(2^n)$

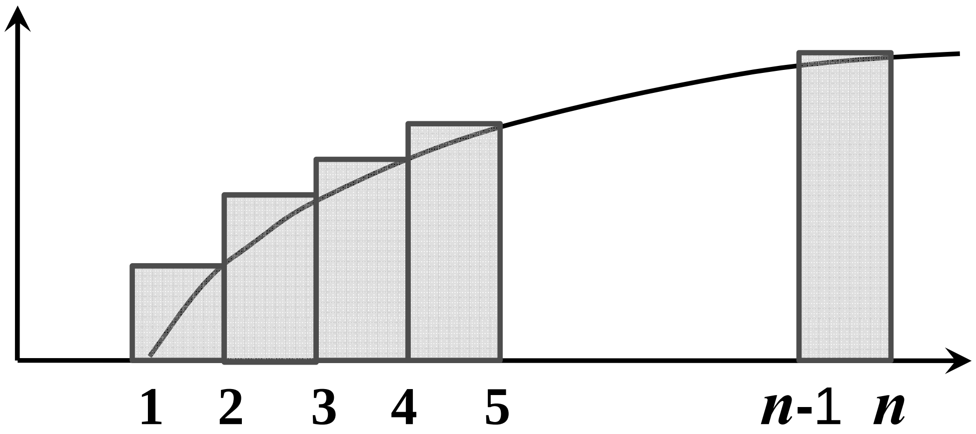
$\log(n!) = \Theta(n\log n)$

$$\log(n!) = \sum_{k=1}^{n}\log k$$

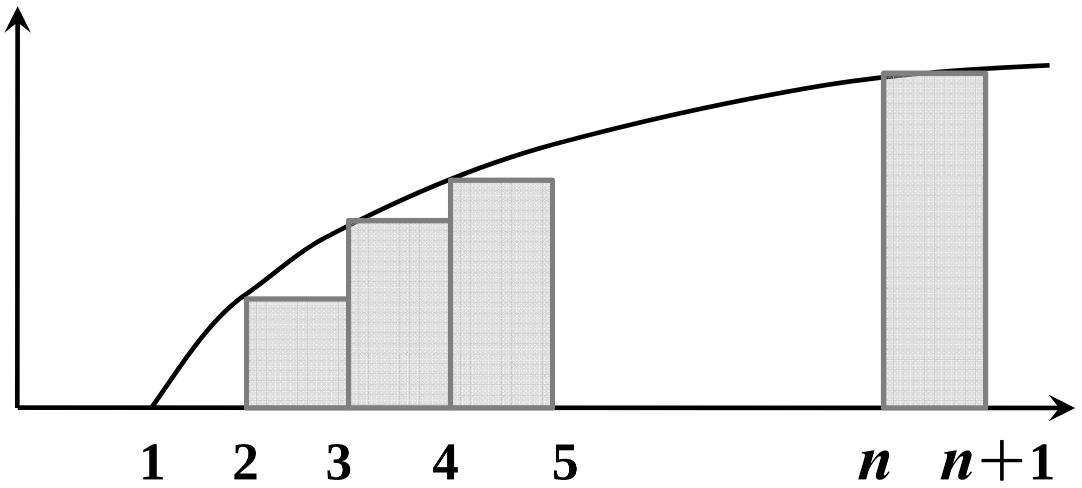
$$\geq \int_1^n \log x dx$$

$$= \log e(n\ln n - n + 1)$$

$$= \Omega(n\log n)$$



阶乘（续）



$$\log(n!) = \sum_{k=1}^{n} \log k \le \int_{2}^{n+1} \log x dx = O(n \log n)$$

例2：函数的阶

按照阶从高到低对以下函数排序：

$$\log^2 n,\quad 1,\quad n!,\quad n2^n,\quad n^{1/\log n},\quad (3/2)^n,\quad \sqrt{\log n},\quad (\log n)^{\log n},$$

$$2^{2^n},\quad n^{\log\log n},\quad n^3,\quad \log\log n,\quad n\log n,\quad n,\quad 2^{\log n},\quad \log n,\quad \log(n!)$$

结果：

$$2^{2^n},\quad n!,\quad n2^n,\quad (3/2)^n,\quad (\log n)^{\log n} = n^{\log\log n},$$

$$n^3,\quad \log(n!) = \Theta(n\log n),\quad n = \Theta(2^{\log n}),$$

$$\log^2 n,\quad \log n,\quad \sqrt{\log n},\quad \log\log n,$$

$$n^{1/\log n} = \Theta(1)$$

函数阶的渐近性质

例3　PrimalityTest(n)

输入：n, n为大于 2 的奇整数
输出：true 或者false

1．$s \leftarrow \sqrt{n}$
2．for $j \leftarrow 2$ to s
3．　　if j 整除 n
4．　　　　then return false
5．return true

假设计算 $\sqrt{n}$ 可以在$O(1)$时间完成，可以写$O(\sqrt{n})$,
不能写$\Theta(\sqrt{n})$

取整函数

$\lfloor x \rfloor$：表示小于等于x的最大的整数

$\lceil x \rceil$：表示大于等于x的最小的整数

取整函数具有下述性质：

(1) $x-1 < \lfloor x \rfloor \le x \le \lceil x \rceil < x+1$

(2) $\lfloor x+n \rfloor = \lfloor x \rfloor + n$, $\lceil x+n \rceil = \lceil x \rceil + n$，其中$n$为整数

(3) $\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor = n$

(4) $\left\lceil \frac{\left\lceil \frac{n}{a} \right\rceil}{b} \right\rceil = \left\lceil \frac{n}{ab} \right\rceil$, $\left\lfloor \frac{\left\lfloor \frac{n}{a} \right\rfloor}{b} \right\rfloor = \left\lfloor \frac{n}{ab} \right\rfloor$

求和公式

基本求和公式

$$\sum_{k=1}^{n} a_k = \frac{n(a_1 + a_n)}{2}, \quad \{a_k\}为等差数列$$

$$\sum_{k=0}^{n} aq^k = \frac{a(1-q^{n+1})}{1-q}, \quad \sum_{k=0}^{n} x^k = \frac{1-x^{n+1}}{1-x},$$

$$\sum_{k=0}^{\infty} aq^k = \frac{a}{1-q} (q<1)$$

$$\sum_{k=1}^{n} \frac{1}{k} = \ln n + O(1)$$

估计和式上界的方法

放大法：

1. $\sum_{k=1}^{n} a_k \le n a_{\max}$

2. 假设存在常数 $r<1$，使得 对一切 $k \ge 0$有 $a_{k+1}/a_k \le r$ 成立

$$\sum_{k=0}^{n} a_k \le \sum_{k=0}^{\infty} a_0 r^k = a_0 \sum_{k=0}^{\infty} r^k = \frac{a_0}{1-r}$$

求和实例

例4 求和

(1) $$\sum_{k=1}^{n-1}\frac{1}{k(k+1)}$$

(2) $$\sum_{t=1}^{k} t2^{t-1}$$

解 (1) $$\sum_{k=1}^{n-1}\frac{1}{k(k+1)}=\sum_{k=1}^{n-1}\left(\frac{1}{k}-\frac{1}{k+1}\right)$$

$$=\sum_{k=1}^{n-1}\frac{1}{k}-\sum_{k=1}^{n-1}\frac{1}{k+1}=\sum_{k=1}^{n-1}\frac{1}{k}-\sum_{k=2}^{n}\frac{1}{k}=1-\frac{1}{n}$$

求和实例

(2) $$\sum_{t=1}^{k} t2^{t-1} = \sum_{t=1}^{k} t(2^{t} - 2^{t-1})$$

$$= \sum_{t=1}^{k} t2^{t} - \sum_{t=1}^{k} t2^{t-1}$$

$$= \sum_{t=1}^{k} t2^{t} - \sum_{t=0}^{k-1} (t+1)2^{t}$$

$$= \sum_{t=1}^{k} t2^{t} - \sum_{t=0}^{k-1} t2^{t} - \sum_{t=0}^{k-1} 2^{t}$$

$$= k2^{k} - (2^{k} - 1)$$

$$= (k-1)2^{k} + 1$$

实例

例5 估计 $\sum_{k=1}^{n}\frac{k}{3^k}$ 的上界.

解 由 $a_k=\frac{k}{3^k},\quad a_{k+1}=\frac{k+1}{3^{k+1}}$

得
$$\frac{a_{k+1}}{a_k}=\frac{1}{3}\frac{k+1}{k}\le\frac{2}{3}$$

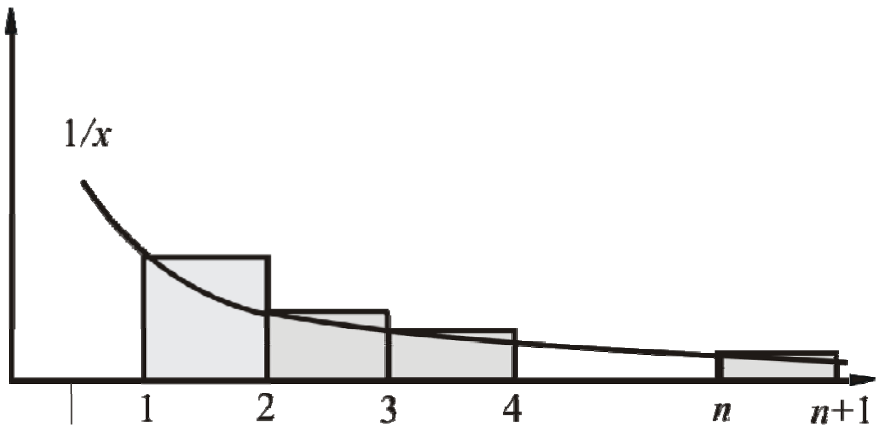
$$\sum_{k=1}^{n}\frac{k}{3^k}\le\sum_{k=1}^{\infty}\frac{1}{3}\left(\frac{2}{3}\right)^{k-1}=\frac{1}{3}\frac{1}{1-\frac{2}{3}}=1$$

估计和式渐近的界

估计 $\sum_{k=1}^{n}\frac{1}{k}$ 的渐近的界.

$$\sum_{k=1}^{n}\frac{1}{k} \geq \int_{1}^{n+1}\frac{dx}{x}$$

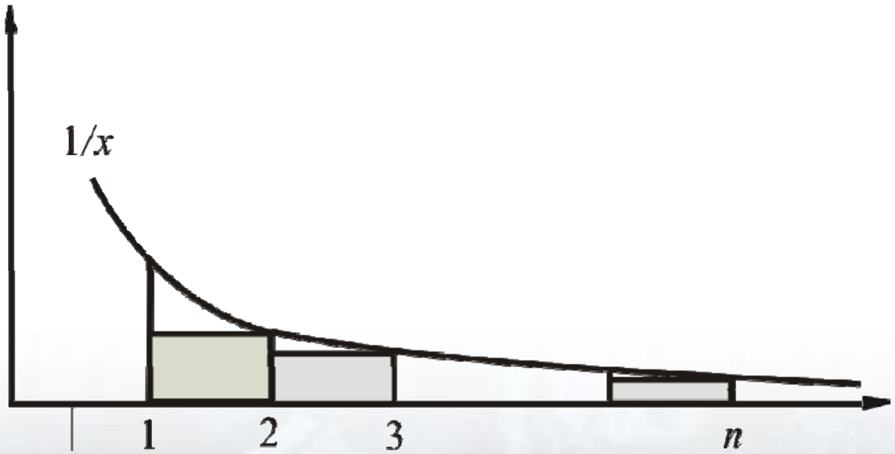
$$= \ln(n+1)$$



$$\sum_{k=1}^{n}\frac{1}{k} = \frac{1}{1} + \sum_{k=2}^{n}\frac{1}{k}$$

$$\leq 1 + \int_{1}^{n}\frac{dx}{x}$$

$$= \ln n + 1$$



递推方程的求解

设序列$a_0, a_1, \ldots, a_n, \ldots$, 简记为$\{a_n\}$, 一个把$a_n$与某些个$a_i(i<n)$联系起来的等式叫做关于序列 $\{a_n\}$ 的递推方程

求解方法:

迭代法

直接迭代：插入排序最坏情况下时间分析

换元迭代：二分归并排序最坏情况下时间分析

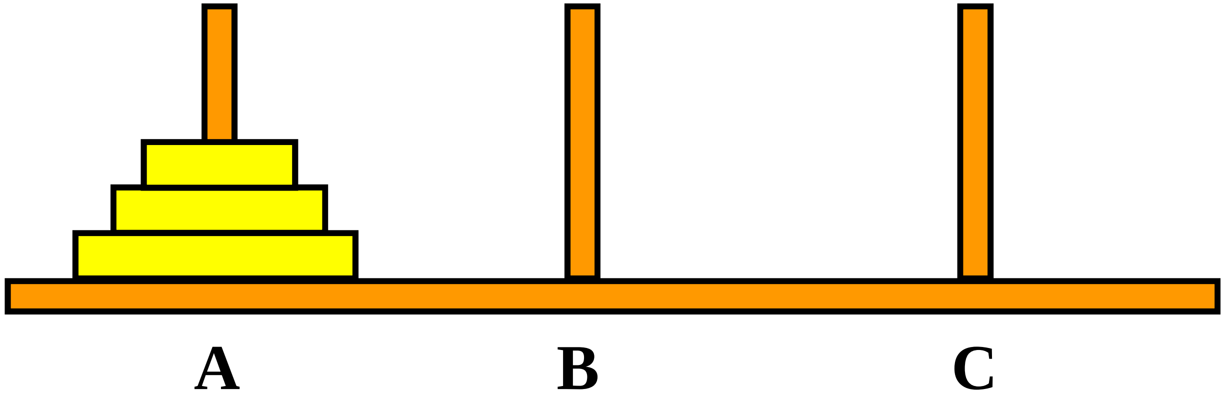
差消迭代：快速排序平均情况下的时间分析

迭代模型：递归树

尝试法：快速排序平均情况下的时间分析

主定理：递归算法的分析

例6：Hanoi塔



算法1.3 Hanoi(A,C,n)　　// 将A的n个盘子按要求移到C

1. if n=1 then move (A,C)　// 将A的1个盘子移到C

2. else Hanoi(A,B,n–1)

3. move(A,C)

4. Hanoi(B,C,n–1)

$T(n) = 2\,T(n-1) + 1$，$T(1) = 1$，　迭代解得　$T(n) = 2^n - 1$

1秒移1个，**64**个盘子要多少时间？(**5000**亿年)

直接迭代：插入排序

算法1.4 InsertSort(A,n) // A为n个数的数组

1. for $j \leftarrow 2$ to n
2. $x \leftarrow A[j]$
3. $i \leftarrow j-1$ // 行3到行7把$A[j]$插入$A[1..j-1]$之中
4. while $i>0$ and $x<A[i]$ do
5. $A[i+1] \leftarrow A[i]$
6. $i \leftarrow i-1$
7. $A[i+1] \leftarrow x$

$$\begin{cases} W(n)=W(n-1)+n-1 \\ W(1)=0 \end{cases}$$

$W(n)=W(n-1)+n-1$

$=[W(n-2)+n-2]+n-1=W(n-2)+(n-2)+(n-1)$

$=[W(n-3)+n-3]+(n-2)+(n-1=\ldots$

$=W(1)+1+2+\ldots+(n-2)+(n-1)$

$=1+2+\ldots+(n-2)+(n-1)=n(n-1)/2$

北京大学

二分归并排序

算法1.5 **MergeSort**(A,p,r)　// 归并排序数组$A[p..r]$

1. if $p<r$
2. then $q \leftarrow \lfloor (p+r)/2 \rfloor$
3. MergeSort(A,p,q)
4. MergeSort($A,q+1,r$)
5. Merge(A,p,q,r)

$$\begin{cases} W(n) = 2W(n/2) + n - 1 \\ W(1) = 0 \end{cases}$$

归并过程

算法1.6 Merge(A,p,q,r) // 将排序数组$A[p..q]$与$A[q+1,r]$合并

1. $x \leftarrow q-p+1,\ y \leftarrow r-q$ // x, y分别为两个子数组的元素数
2. 将$A[p..q]$复制到$B[1..x]$, 将$A[q+1..r]$复制到$C[1..y]$
3. $i \leftarrow 1,\ j \leftarrow 1,\ k \leftarrow p$
4. while $i \le x$ and $j \le y$ do
5. if $B[i] \le C[j]$ // B的首元素不大于C的首元素
6. $A[k] \leftarrow B[i]$ // 将B的首元素放到A中
7. $i \leftarrow i+1$
8. else
9. $A[k] \leftarrow C[j]$
10. $j \leftarrow j+1$
11. $k \leftarrow k+1$
12. if $i > x$ then 将$C[j..y]$复制到$A[k..r]$ // B已经是空数组
13. else 将$B[i..x]$复制到$A[k..r]$ // C已经是空数组

换元迭代

$$\begin{cases} W(n) = 2W(n/2) + n - 1, \quad n = 2^k \\ W(1) = 0 \end{cases}$$

$$W(n) = 2W(2^{k-1}) + 2^k - 1$$

$$= 2[2W(2^{k-2}) + 2^{k-1} - 1] + 2^k - 1$$

$$= 2^2 W(2^{k-2}) + 2^k - 2 + 2^k - 1$$

$$= 2^2[2W(2^{k-3}) + 2^{k-2} - 1] + 2^k - 2 + 2^k - 1$$

$$= \ldots$$

$$= 2^k W(1) + k2^k - (2^{k-1} + 2^{k-2} + \ldots + 2 + 1)$$

$$= k2^k - 2^k + 1$$

$$= n\log n - n + 1$$

使用迭代法，对解可以通过数学归纳法验证

差消化简后迭代

$$\begin{cases} T(n) = \dfrac{2}{n}\sum_{i=1}^{n-1} T(i) + O(n), \quad n \ge 2 \\ T(1) = 0 \end{cases}$$

快速排序平均时间分析

$$nT(n) = 2\sum_{i=1}^{n-1} T(i) + cn^2,$$ c为某个常数

$$(n-1)T(n-1) = 2\sum_{i=1}^{n-2} T(i) + c(n-1)^2$$

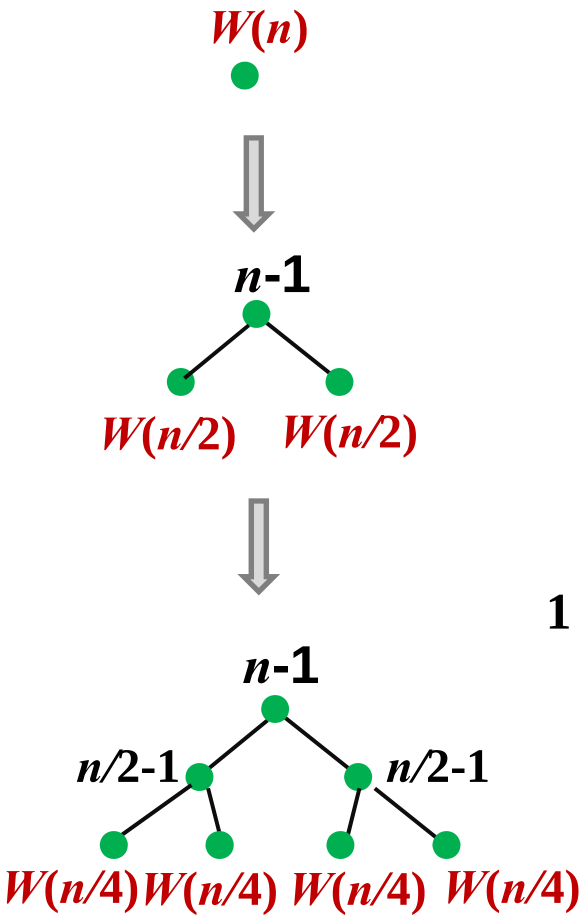
$$nT(n) = (n+1)T(n-1) + O(n)$$

$$\frac{T(n)}{n+1} = \frac{T(n-1)}{n} + \frac{c_1}{n+1} = \cdots = c_1\left[\frac{1}{n+1} + \frac{1}{n} + \ldots + \frac{1}{3} + \frac{T(1)}{2}\right]$$

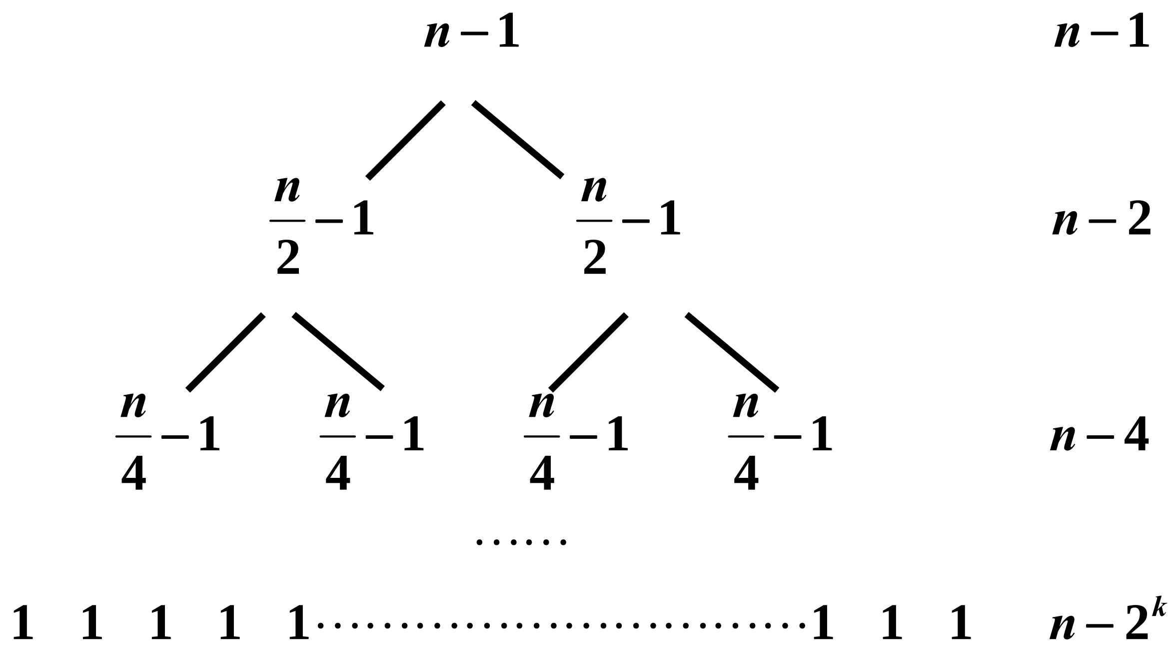
$$= c_1\left[\frac{1}{n+1} + \frac{1}{n} + \ldots + \frac{1}{3}\right] = \Theta(\log n), \quad T(n) = \Theta(n\log n)$$

北京大学

迭代模型：递归树



生成过程



$$W(n)=2W(n/2)+n-1,\ n=2^k,\quad W(1)=0$$

$$W(n)=n-1+n-2+\ldots+n-2^k$$

$$=kn-(2^k-1)=n\log n-n+1$$

递归树的应用实例

$T(n)=T(n/3)+T(2n/3)+n$

$$n \qquad\qquad n$$

$$\frac{n}{3} \quad \frac{2n}{3} \qquad\qquad n$$

$$\frac{n}{9} \quad \frac{2n}{9} \quad \frac{2n}{9} \quad \frac{2n}{9} \qquad\qquad n$$

……

………………………… 1 $\qquad O(n)$

层数 k: $\quad n(2/3)^k=1 \Rightarrow (3/2)^k=n \Rightarrow k=O(\log_{3/2}n)$

$T(n)=O(n\log n)$

尝试法：快速排序

$$T(n)=\frac{2}{n}\sum_{i=1}^{n-1}T(i)+O(n)$$

(1) $T(n)=C$为常函数，左边$=O(1)$

右边$=\frac{2}{n}C(n-1)+O(n)$

$$=2C-\frac{2C}{n}+O(n)$$

(2) $T(n)=cn$，左边$=cn$

右边$=\frac{2}{n}\sum_{i=1}^{n-1}ci+O(n)$

$$=\frac{2c}{n}\frac{(1+n-1)(n-1)}{2}+O(n)$$

$$=cn-c+O(n)$$

尝试法：快速排序

$$\begin{cases} T(n)=\frac{2}{n}\sum_{i=1}^{n-1}T(i)+O(n), & n\geq 2 \\ T(1)=0 \end{cases}$$

(3) $T(n)=cn^2$，左边$=cn^2$

$$右边=\frac{2}{n}\sum_{i=1}^{n-1}ci^2+O(n)$$

$$=\frac{2}{n}[\frac{cn^3}{3}+O(n^2)]+O(n)=\frac{2c}{3}n^2+O(n)$$

(4) $T(n)=cn\log n$，左边$=cn\log n$

$$右边=\frac{2c}{n}\sum_{i=1}^{n-1}i\log i+O(n)$$

$$=\frac{2c}{n}[\frac{n^2}{2}\log n-\frac{n^2}{4\ln 2}+O(n\log n)]+O(n)$$

$$=cn\log n+O(n)+O(\log n)$$

以积分作为求和的近似

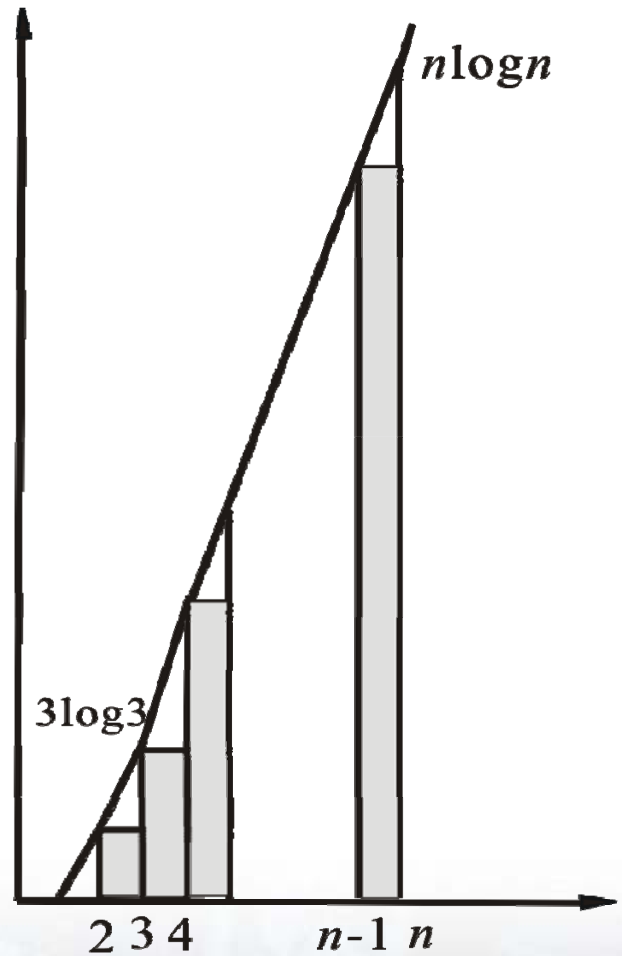
$$\int_2^n x\log x dx = \int_2^n \frac{x}{\ln 2}\ln x dx$$

$$= \frac{1}{\ln 2}\left[\frac{x^2}{2}\ln x - \frac{x^2}{4}\right]\Bigg|_2^n$$

$$= \frac{1}{\ln 2}\left(\frac{n^2}{2}\ln n - \frac{n^2}{4}\right) - \frac{1}{\ln 2}\left(\frac{4}{2}\ln 2 - \frac{4}{4}\right)$$

$$\sum_{i=1}^{n-1} i\log i$$

$$= \frac{n^2}{2}\log n - \frac{n^2}{4\ln 2} + O(n\log n)$$



主定理

主定理：设$a\geq 1, b>1$为常数，$f(n)$为函数，$T(n)$为非负整数，且

$$T(n)=aT(n/b)+f(n)$$

则有以下结果：

1. 若$f(n)=O(n^{\log_b a-\varepsilon}),\varepsilon>0$，那么$T(n)=\Theta(n^{\log_b a})$
2. 若$f(n)=\Theta(n^{\log_b a})$，那么$T(n)=\Theta(n^{\log_b a}\log n)$
3. 若$f(n)=\Omega(n^{\log_b a+\varepsilon}),\varepsilon>0$，且对于某个常数$c<1$和充分大的$n$有 $af(n/b)\leq cf(n)$，那么$T(n)=\Theta(f(n))$

北京大学

迭代

设$n=b^k$

$$T(n)=aT(\frac{n}{b})+f(n)$$

$$=a[aT(\frac{n}{b^2})+f(\frac{n}{b})]+f(n)=a^2T(\frac{n}{b^2})+af(\frac{n}{b})+f(n)$$

$$=\ldots$$

$$=a^kT(\frac{n}{b^k})+a^{k-1}f(\frac{n}{b^{k-1}})+\ldots+af(\frac{n}{b})+f(n)$$

$$=a^kT(1)+\sum_{j=0}^{k-1}a^jf(\frac{n}{b^j})$$

$$=c_1n^{\log_b a}+\sum_{j=0}^{k-1}a^jf(\frac{n}{b^j}) \qquad T(1)=c_1$$

北京大学

情况1 $f(n)=O(n^{\log_b a-\varepsilon})$

$$T(n)=c_1 n^{\log_b a}+\sum_{j=0}^{k-1} a^j f(\frac{n}{b^j})$$

$$=c_1 n^{\log_b a}+O(\sum_{j=0}^{\log_b n-1} a^j (\frac{n}{b^j})^{\log_b a-\varepsilon})$$

$$=c_1 n^{\log_b a}+O(n^{\log_b a-\varepsilon}\sum_{j=0}^{\log_b n-1}\frac{a^j}{(b^{\log_b a-\varepsilon})^j})$$

$$=c_1 n^{\log_b a}+O(n^{\log_b a-\varepsilon}\sum_{j=0}^{\log_b n-1}(b^{\varepsilon})^j)$$

$$=c_1 n^{\log_b a}+O(n^{\log_b a-\varepsilon}\frac{b^{\varepsilon\log_b n}-1}{b^{\varepsilon}-1})$$

$$=c_1 n^{\log_b a}+O(n^{\log_b a-\varepsilon}n^{\varepsilon})=O(n^{\log_b a})$$

情况2 $f(n)=\Theta(n^{\log_b a})$

$$T(n)=c_1 n^{\log_b a}+\sum_{j=0}^{k-1} a^j f(\frac{n}{b^j})$$

$$=c_1 n^{\log_b a}+\Theta(\sum_{j=0}^{\log_b n-1} a^j (\frac{n}{b^j})^{\log_b a})$$

$$=c_1 n^{\log_b a}+\Theta(n^{\log_b a}\sum_{j=0}^{\log_b n-1}\frac{a^j}{a^j})$$

$$=c_1 n^{\log_b a}+\Theta(n^{\log_b a}\log n)$$

$$=\Theta(n^{\log_b a}\log n)$$

情况3 $f(n)=\Omega(n^{\log_b a+\varepsilon})$

$$T(n)=c_1 n^{\log_b a}+\sum_{j=0}^{k-1}a^j f(\frac{n}{b^j})$$

$$\le c_1 n^{\log_b a}+\sum_{j=0}^{\log_b n-1}c^j f(n) \qquad (af(\frac{n}{b})\le cf(n))$$

$$=c_1 n^{\log_b a}+f(n)\frac{c^{\log_b n}-1}{c-1}$$

$$=c_1 n^{\log_b a}+\Theta(f(n)) \qquad （c<1）$$

$$=\Theta(f(n)) \qquad (f(n)=\Omega(n^{\log_b a+\varepsilon}))$$

北京大学

主定理的应用

例7 求解递推方程 $T(n)=9T(n/3)+n$

解 上述递推方程中的 $a=9, b=3, f(n)=n$，那么

$$n^{\log_3 9}=n^2, \quad f(n)=O(n^{\log_3 9-1}),$$

相当于主定理的第一种情况，其中ε=1. 根据定理得到

$$T(n)=\Theta(n^2)$$

例8 求解递推方程 $T(n)=T(2n/3)+1$

解 上述递推方程中的 a=1, b=3/2, $f(n)$=1，那么

$$n^{\log_{3/2} 1}=n^0=1, \quad f(n)=1$$

相当于主定理的第二种情况. 根据定理得到 .

$$T(n)=\Theta(n^0 \log n)=\Theta(\log n)$$

实例

例9 求解递推方程

$$T(n) = 3T(n/4) + n\log n$$

解 上述递推方程中的a=3, b=4, $f(n)$=$n$log$n$，那么

$$n\log n = \Omega(n^{\log_4 3+\varepsilon}) = \Omega(n^{0.793+\varepsilon})，\ \varepsilon\approx 0.2$$

此外，要使 $a\,f(n/b) \le cf(n)$成立，代入 $f(n)$=$n$log$n$，得到

$$\frac{3n}{4}\log\frac{n}{4} \le cn\log n$$

显然只要 $c \ge 3/4$，上述不等式就可以对充分大的n成立.
相当于主定理的第三种情况. 因此有

$$T(n) = \Theta(f(n)) = \Theta(n\log n)$$

$n^{\log_2 2} = n$

不能使用主定理的例子

例10 求解 $T(n) = 2T(n/2) + n\log n$

$a=b=2$, $n^{\log_2 2} = n$ ， $f(n)=n\log n$

不存在$\varepsilon > 0$ 使 $n\log n = \Omega(n^{1+\varepsilon})$， 不存在$c<1$使$2(n/2)f(n) \le cf(n)$

$$n\log n \qquad\qquad n\log n$$

$$\frac{n(\log n-1)}{2} \quad \frac{n(\log n-1)}{2} \qquad\qquad n(\log n-1)$$

$$\frac{n(\log n-2)}{4} \quad \frac{n(\log n-2)}{4} \quad \frac{n(\log n-2)}{4} \quad \frac{n(\log n-2)}{4} \qquad n(\log n-2)$$

....

$$T(n) = n\log n + n(\log n - 1) + n(\log n - 2) + ... + n(\log n - k + 1)$$

$$= (n\log n)\log n - n(1+2+...+k-1)$$

$$= n\log^2 n - nk(k-1)/2 = O(n\log^2 n)$$

北京大学