算法设计与分析

孙永谦 沈舒尹
sunyongqian@nankai.edu.cn
shenshuyin@nankai.edu.cn

课件参考《算法设计与分析》(第2版)，屈婉玲 刘田等，清华大学出版社

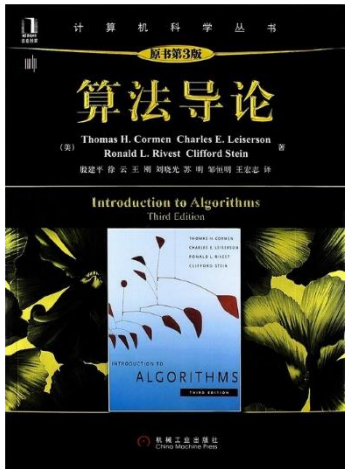
教材

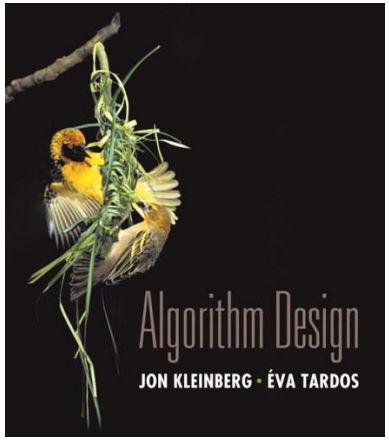
《算法设计与分析》(第2版)

- 屈婉玲 刘田等编著
- 清华大学出版社

推荐书目：

- 《算法导论》(第3版)，Thomas H.Cormen， Charles E.Leiserson，Ronald L.Rivest，Clifford Stein 著，殷建平，徐云，王刚 等 译，机械工业出版社
- 《Algorithm Design》， Jon Kleinberg， Éva Tardos著





课程主要内容

问题处理策略

计算复杂性理论

算法分析方法

算法设计技术

基础知识

第1章　基础知识

1.1 有关算法的基本概念

几个例子

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

1.3.1 函数的渐近的界

1.3.2 求和的方法

1.3.3 递推方程求解方法

1.1 有关算法的基本概念

几个例子

例1.1 调度问题

已知：任务集 $S=\{1, 2, \dots, n\}$，

第 j 项任务加工时间：$t_j \in Z^+$（正整数），$j=1, 2, \dots, n$

一个可行调度方案是$1, 2, \dots, n$ 的一个排列 $i_1, i_2, \dots, i_n$.

求：总等待时间最少的调度方案，即求S的排列$i_1, i_2, \dots, i_n$使得

$$\min\{\sum_{k=1}^{n}(n-k+1)t_{i_k}\}$$

实例：

任务集 $S = \{1, 2, 3, 4, 5\}$，

加工时间：$t_1=3$，$t_2=8$，$t_3=5$，$t_4=10$，$t_5=15$

调度问题的贪心法的解

算法：加工时间：t_1=3, t_2=8, t_3=5, t_4=10, t_5=15. 按加工时间 从小到大安排: (3,5,8,10,15)

解：1, 3, 2, 4, 5

0 3 8 16 26 41

总完成时间：

$t = 3+(3+5)+(3+5+8)+(3+5+8+10)+(3+5+8+10+15)$

$= 3 \times 5 + 5 \times 4 + 8 \times 3 + 10 \times 2 + 15$

$= 94$

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1.2 排序算法的评价

例1.2 排序算法

考察元素比较次数

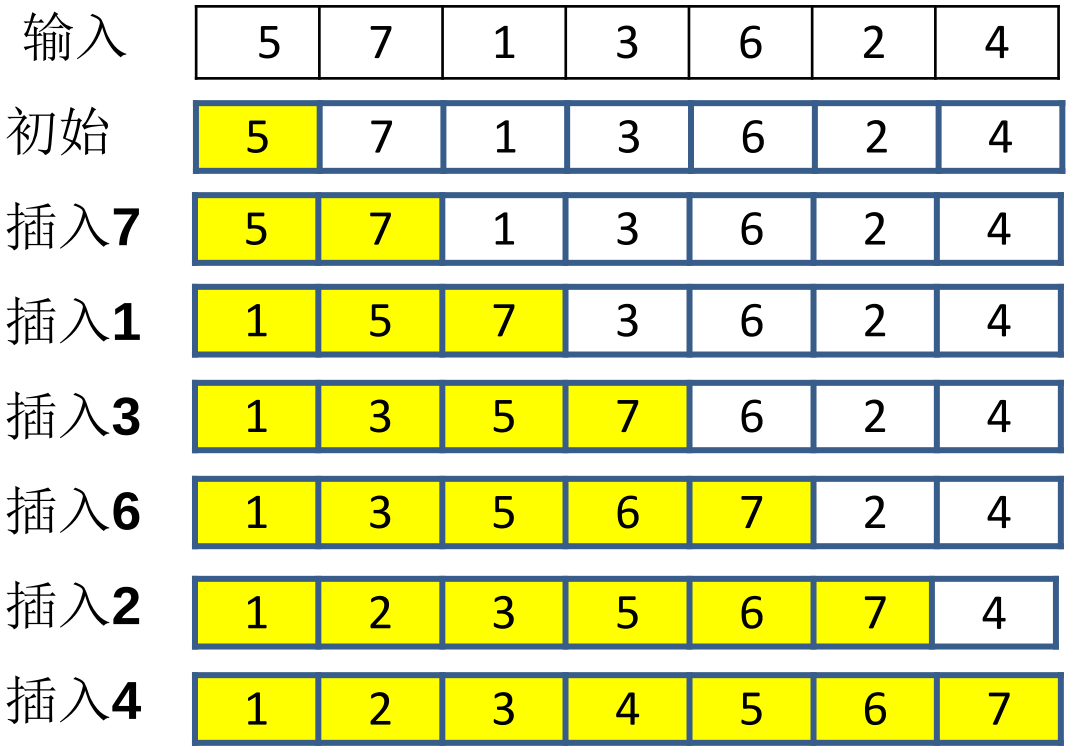
- 插入排序、冒泡排序：最坏和平均状况下都为$O(n^2)$
- 快速排序：最坏状况为$O(n^2)$，平均状况下为$O(n\log n)$
- 堆排序、二分归并排序：最坏和平均状况下都为$O(n\log n)$

…

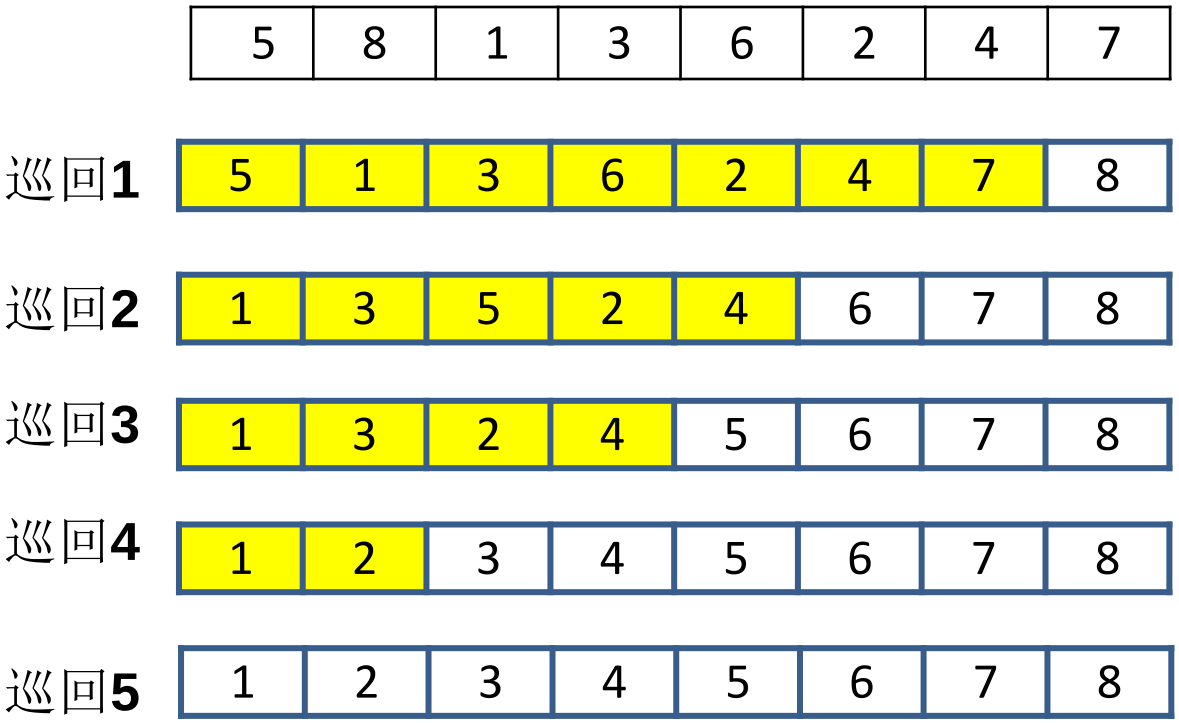
问题

- 哪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？

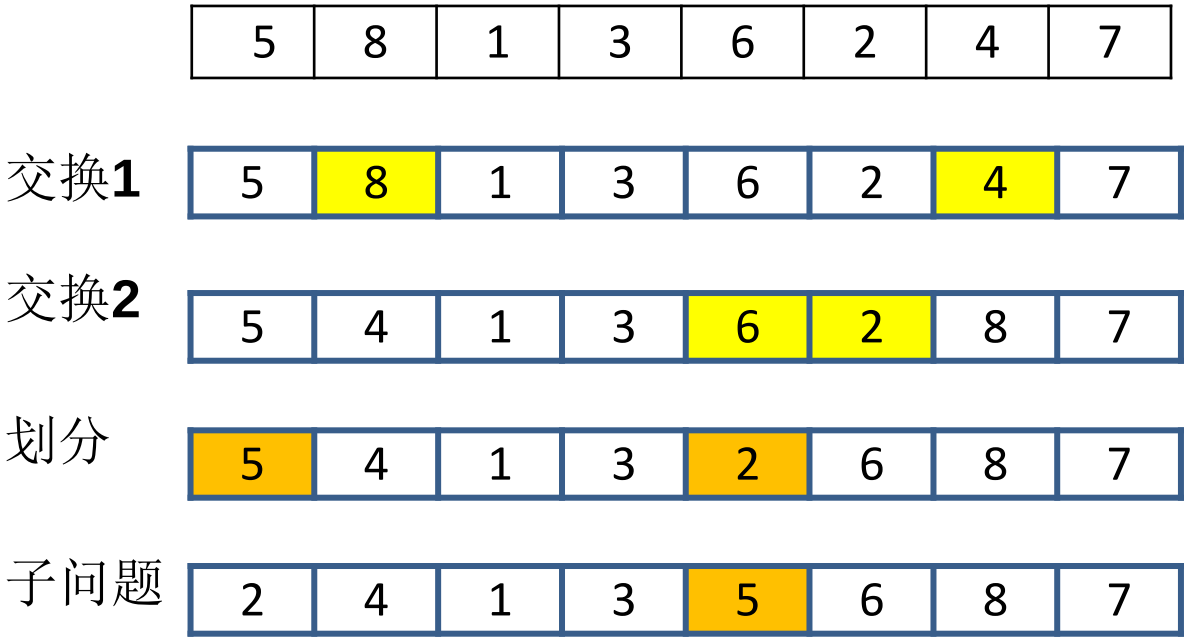
插入排序运行实例


输入
5 7 1 3 6 2 4
初始
5 7 1 3 6 2 4
插入7
5 7 1 3 6 2 4
插入1
1 5 7 3 6 2 4
插入3
1 3 5 7 6 2 4
插入6
1 3 5 6 7 2 4
插入2
1 2 3 5 6 7 4
插入4
1 2 3 4 5 6 7

冒泡排序运行实例


5 8 1 3 6 2 4 7
巡回1
5 1 3 6 2 4 7 8
巡回2
1 3 5 2 4 6 7 8
巡回3
1 3 2 4 5 6 7 8
巡回4
1 2 3 4 5 6 7 8
巡回5
1 2 3 4 5 6 7 8

快速排序一次递归运行



第一个数做key，左右开工，从前往后找比key大的，同时从后往前找比key小的，找到就交换位置。左右相遇后插入key到中间。

二分归并排序运行实例


5 8 1 3 6 2 4 7
划分
5 8 1 3 6 2 4 7
递归排序
1 3 5 8 2 4 6 7
1 3 5 8 2 4 6 7
合并后的输出
1 2 3 4 5 6 7 8

问题的计算复杂度分析

问题：

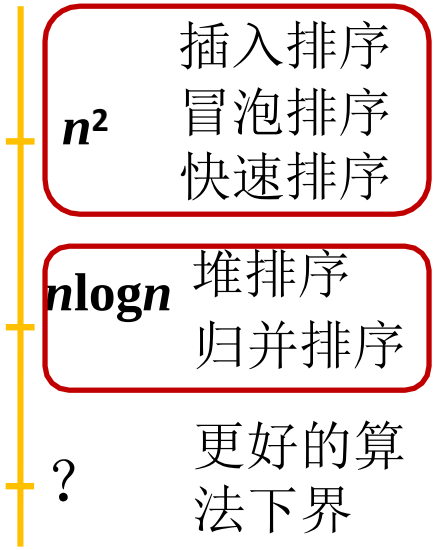
哪个排序算法效率最高？

是否可找到更好的排序算法？

排序问题计算难度如何？

其他问题的计算复杂度

问题计算复杂度估计方法



哪个排序算法效率最高?

排序问题的难度？

例1.3 货郎问题

例1.3 货郎问题：

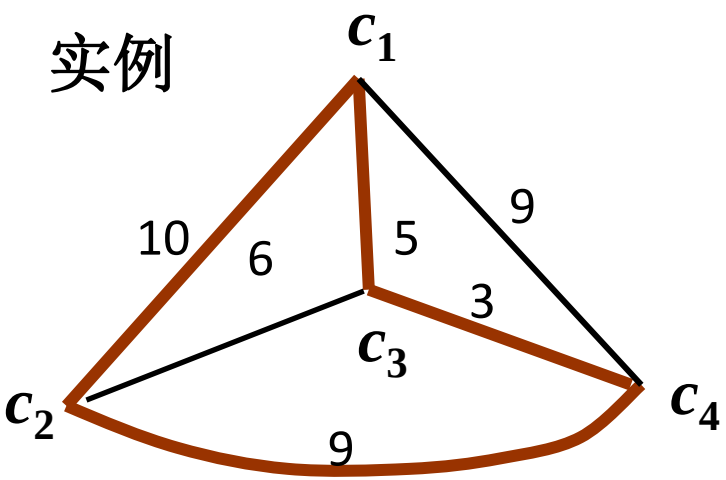
- 有穷个城市的集合$C = \{c_1, c_2, \ldots, c_m\}$, 距离

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

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

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

实例



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？

0-1背包问题

0-1背包问题：有n件物品要装入背包，第i 件物品的重量 w_i, 价值v_i，i=1,2,…,n. 背 包最多允许装入的重量为B, 问如何选择装 入背包的物品，使得总价值达到最大？

实例：n=4,B=6，物品的重量和价值如下

标号	1	2	3	4
重量 w_i	3	4	5	2
价值 v_i	7	9	9	2

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**

  有限条指令的集合，这个指令集确定了解决某个问题的运算或操作的序列.

- **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.

算法的基本概念

- **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. $r \leftarrow n \bmod m$

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

算法复杂度的*O*表示：

- 插入排序、冒泡排序：最坏和平均状况下都为$O(n^2)$
- 堆排序、二分归并排序：最坏和平均状况下都为$O(n\log n)$

规律：

- 实例规模很大时，算法的效率差异更为明显
- 不直接比较$W(n)$或$A(n)$，而是关注n充分大时函数的阶
  - 例如$W(n)=n^2+8n-1$，当n充分大时，后面两项可忽略，只关注n^2

1.3 数学基础： 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 $f(n)= o(g(n))$.

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

大O符号

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

例子：设 $f(n) = n^2 + n$，则

$f(n)=O(n^2)$，取 $c = 2$，$n_0 =1$ 即可

$f(n)=O(n^3)$，取 $c = 1$，$n_0 =2$ 即可

1. $f(n) = O(g(n))$ ，$f(n)$的阶不高于$g(n)$的阶.

2. 可能存在多个正数c，只要指出一个即可.

3. 对前面有限个值可以不满足不等式.

4. 常函数可以写作$O(1)$.

大Ω符号

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

例子：设 $f(n) = n^2 + n$，则

$f(n) = \Omega(n^2)$, 取 $c = 1, n_0 = 1$即可

$f(n) = \Omega(100n)$, 取 $c=1/100$, $n_0 = 1$即可

1. $f(n)=\Omega(g(n))$，$f(n)$的阶不低于$g(n)$的阶.
2. 可能存在多个正数c，指出一个即可.
3. 对前面有限个 n 值可以不满足上述不等式.

小o符号

(3) 若对任意正数 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 $f(n)= o(g(n))$.

例子：$f(n)=n^2+n$，则

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

$c \geq 1$显然成立，因为$n^2+n<cn^3$（$n_0=2$）

任给$1>c>0$, 取 $n_0 > \lceil 2/c \rceil$即可. 因为 $cn \geq cn_0 > 2$ （$n \geq n_0$）

所以，$n^2+n < 2n^2 < cn^3$

1. $f(n) = o(g(n))$, $f(n)$的阶低于$g(n)$的阶
2. 对不同正数c, n_0不一样. c越小n_0越大.
3. 对前面有限个 n 值可以不满足不等式.

小w符号

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

例子：设 $f(n) = n^2 + n$，则

$$f(n) = \omega(n),$$

不能写 $f(n) = \omega(n^2)$，因为取 $c = 2$，不存在n_0 使得对一切 $n \geq n_0$有下式成立

$c\,n^2 = 2n^2 < n^2 + n$ ✗

1. $f(n)=\omega(g(n))$, $f(n)$的阶高于$g(n)$ 的阶.
2. 对不同的正数c, n_0不等，c 越大n_0 越大.
3. 对前面有限个 n 值可以 不满足不等式.

Θ符号

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

例子：$f(n) = n^2 + n$, $g(n) = 100n^2$，那么有

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

1. $f(n)$ 的阶与 $g(n)$ 的阶相等.
2. 对前面有限个 n 值可以不满足条件.

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 $\lim\limits_{n\to\infty}\dfrac{f(n)}{g(n)}$ 存在且等于某个常数$c>0$, 那么

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

(2) 如果 $\lim\limits_{n\to\infty}\dfrac{f(n)}{g(n)}=0$，那么 $f(n)=o(g(n))$.

(3) 如果 $\lim\limits_{n\to\infty}\dfrac{f(n)}{g(n)}=+\infty$，那么 $f(n)=\omega(g(n))$.

证明定理1.1 (1)

证 （1）根据极限定义，对于给定的正数 $\varepsilon = c/2$, 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 $f(n)= \Theta(g(n))$

有关阶的一些性质

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, 有 $f=O(h)$ 和 $g=O(h)$，那么 $f + g = O(h)$.

该性质可以推广到有限个函数.

算法由有限步骤构成,若每一步的时间复杂度函数的上界都是$h(n)$，那么该算法的时间复杂度函数可以写作 $O(h(n))$.

实例

例4 设 $f(n)=\frac{1}{2}n^2-3n$ ， 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 $f(n)=\Theta(n^2)$.

定理1.5 多项式函数，幂函数的阶低于指数函数

$$n^d=o(r^n)，\ r>1，\ d>0$$

证 不妨设d为正整数，

$$\lim_{n\to\infty}\frac{n^d}{r^n}=\lim_{n\to\infty}\frac{dn^{d-1}}{r^n\ln r}=\lim_{n\to\infty}\frac{d(d-1)n^{d-2}}{r^n(\ln r)^2}$$

$$=\ldots=\lim_{n\to\infty}\frac{d!}{r^n(\ln r)^d}=0$$

分子分母
求导数

几种重要的函数

多项式函数：

$f(n) = a_0+a_1n+a_2n^2+\ldots+a_dn^d$, 其中$a_d \neq 0$，d为常数

易知，

$$f(n) = \Omega(n^d),$$

$$f(n) = O(n^d)$$

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

对数函数

符号：

$\log n = \log_2 n$

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

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

性质：

定理1.4 $\log_b n = o(n^\alpha) \quad \alpha > 0$ $\lim_{n\to\infty} \frac{\log_b n}{n^\alpha} = \lim_{n\to\infty} \frac{\ln n}{n^\alpha \ln b} = \lim_{n\to\infty} \frac{1/n}{\alpha n^{\alpha-1}\ln b} = 0$

$a^{\log_b n} = n^{\log_b a}$ 两边取以b为底的对数，都得$\log_b n * \log_b a$

$\log_k n = \Theta(\log_l n)$ $\lim_{n\to\infty} \frac{\log_k n}{\log_l n} = \lim_{n\to\infty} \frac{\log_l n}{\log_l k * \log_l n} = \frac{1}{\log_l k}$

对数性质：

$\log_a n = \frac{\log_b n}{\log_b a}$

$\log_b n^a = a\log_b n$

阶乘

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

$n! = o(n^n)$

$n! = \omega(2^n)$

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

$$\lim_{n\to+\infty}\frac{\log(n!)}{n\log n} = \lim_{n\to+\infty}\frac{\ln(n!)/\ln 2}{n\ln n/\ln 2} = \lim_{n\to+\infty}\frac{\ln(n!)}{n\ln n}$$

$$= \lim_{n\to+\infty}\frac{\ln\left(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1+\left(\frac{c}{n}\right)\right)\right)}{n\ln n} = \lim_{n\to+\infty}\frac{\ln\sqrt{2\pi n}+n\ln\frac{n}{e}}{n\ln n} = 1$$

上述的c为某个常数

函数的阶

例1.6 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：

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

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

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

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

取整函数

$\lfloor x \rfloor$：表示小于等于 x 的最大的整数，称作底函数

$\lceil x \rceil$：表示大于等于 x 的最小的整数，称作顶函数

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) $\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor = n$　其中n为整数

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

1.3.1小结

- 五种复杂度函数的渐进表示： $O, \Omega, o, \omega, \Theta$
- 相关定理
  - 定理1.1， 利用f/g的极限判断阶的高低
  - 定理1.2，阶的高底具有可传递性
  - 定理1.3，如果 $f=O(h)$ 且 $g=O(h)$，那么 $f+g=O(h)$
- 一些重要函数阶的高低
  - 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： $\log n, \log^2 n, \log\log n, \ldots$

指数函数的阶高幂函数，
幂函数的阶高于对数函数

1.3.2 求和的方法

- 几个有用的结果

$$\sum_{k=1}^{n} a_k = \frac{n(a_1 + a_n)}{2}$$

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

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

- 和的上界 $\sum_{k=1}^{n} a_k \le n a_{\max}$
- 假设存在常数 $r<1$，使得 对一切 $k \ge 0$，$\frac{a_{k+1}}{a_k} \le r$ 成立，则

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

求和实例

例1.7 求和

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

解

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

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

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

$$=\sum_{t=1}^{k}t2^t-\sum_{t=0}^{k-1}t2^t-\sum_{t=0}^{k-1}2^t=k2^k-(2^k-1)=(k-1)2^k+1$$

求和实例

有些求和公式表达式不易直接求得，但仍可以估计上界.

例1.8 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

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

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

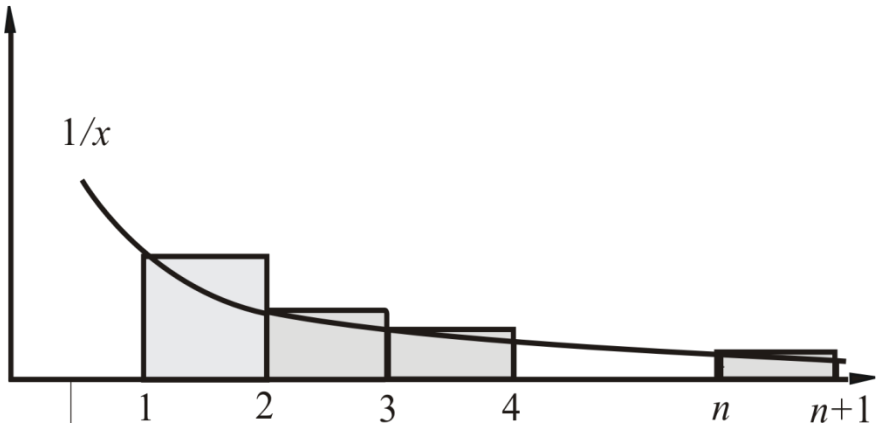
如果常数 $r<1$，使得 $\frac{a_{k+1}}{a_k}\le r$ 成立，则

$$\sum_{k=0}^{n}a_k\le\sum_{k=0}^{\infty}a_0r^k=a_0\sum_{k=0}^{\infty}r^k=\frac{a_0}{1-r}$$

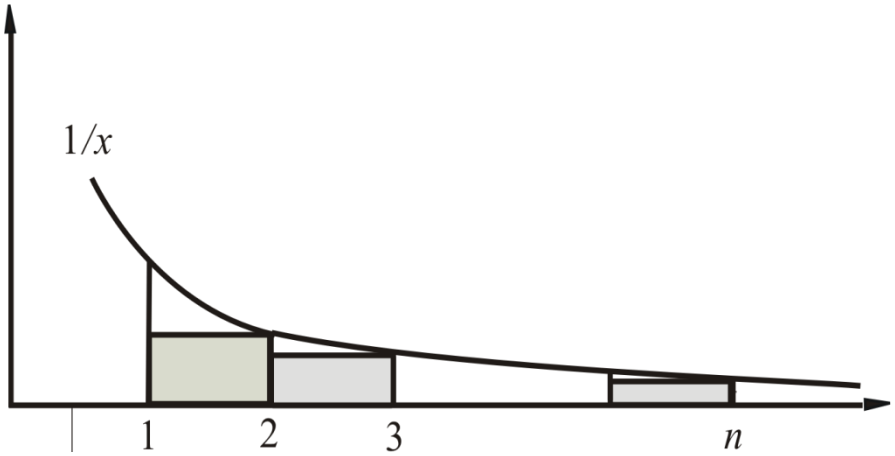
估计和式渐近的界

例1.9 估计 $\sum_{k=1}^{n}\frac{1}{k}$ 的渐近的界.

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



$$\sum_{k=1}^{n}\frac{1}{k}=1+\sum_{k=2}^{n}\frac{1}{k}$$
$$\leq 1+\int_{1}^{n}\frac{dx}{x}$$
$$=\ln n+1$$



1.3.3 递推方程的求解

设序列$a_0, a_1, ..., a_n, ...$, 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

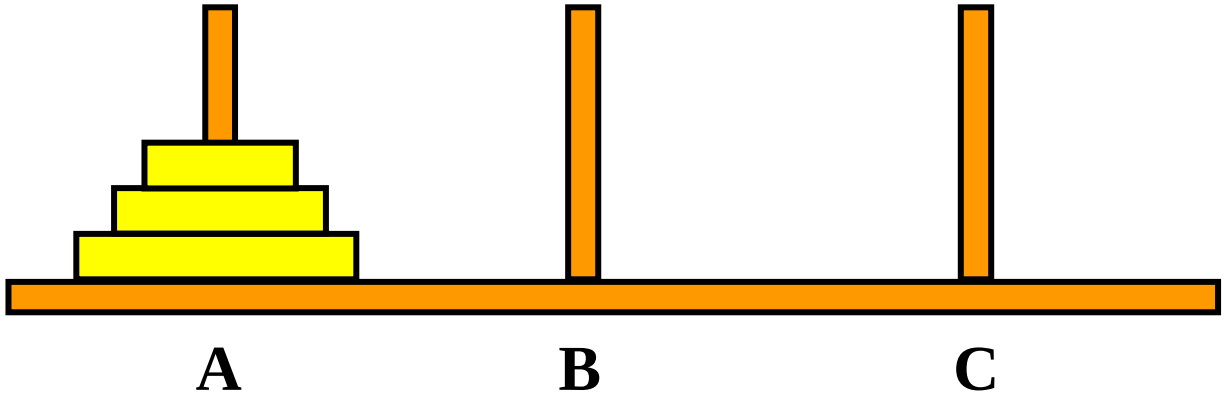
  换元迭代：二分归并排序最坏情况下时间分析

  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1. 10：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？（余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←2 to n do
2.     x←A[j]
3.     i←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]←A[i]
6.         i←i–1
7.     A[i+1]←x
```

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

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

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

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

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

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

二分归并排序

算法1.5 **MergeSort(*A*, *p*, *r*)** // 归并排序数组 ***A*[*p*..*r*], 1≤*p*≤*r*≤*n***

1. if *p*<*r*

2. then *q*←⌊(*p*+*r*)/2⌋

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*←*q*−*p*+1, *y*←*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*←1, *j*←1, *k*←*p*
4. While *i*≤*x* and *j*≤*y* do
5. if *B*[*i*]≤*C*[*j*] // *B*的首元素不大于*C*的首元素
6. then *A*[*k*]←*B*[*i*] // 将*B*的首元素放到*A*中
7. *i*←*i*+1
8. else
9. *A*[*k*]←*C*[*j*]
10. *j*←*j*+1
11. *k*←*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

例1.12(2) 换元迭代

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

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

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

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

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

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

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

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

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

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

快速排序平均时间分析

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

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

$$nT(n)=(n+1)T(n-1)+2cn-c$$

例1.7(1)：

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

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

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

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

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

迭代模型：递归树

递归树思想

- 递归树是求解迭代计算的模型.
- 递归树的生成过程与迭代过程一致.
- 递归树上所有项恰好是迭代展开后和式中的项, 因此对递归树上的项求和就是迭代后方程的解.

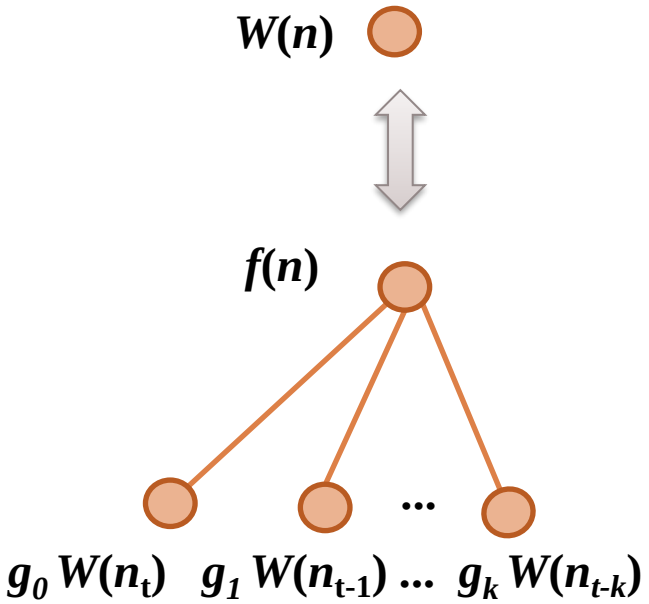
$$W(n) = g_0(n)W(n_t)+...+ g_k(n)\ W(n_{t-k})+ f(n),$$

$n_t < n,\ k < t$

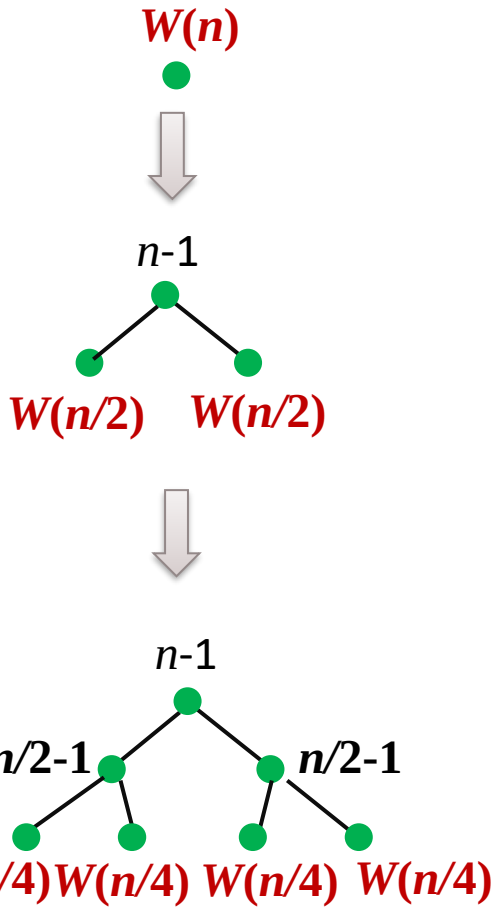
函数项

递归树生成规则

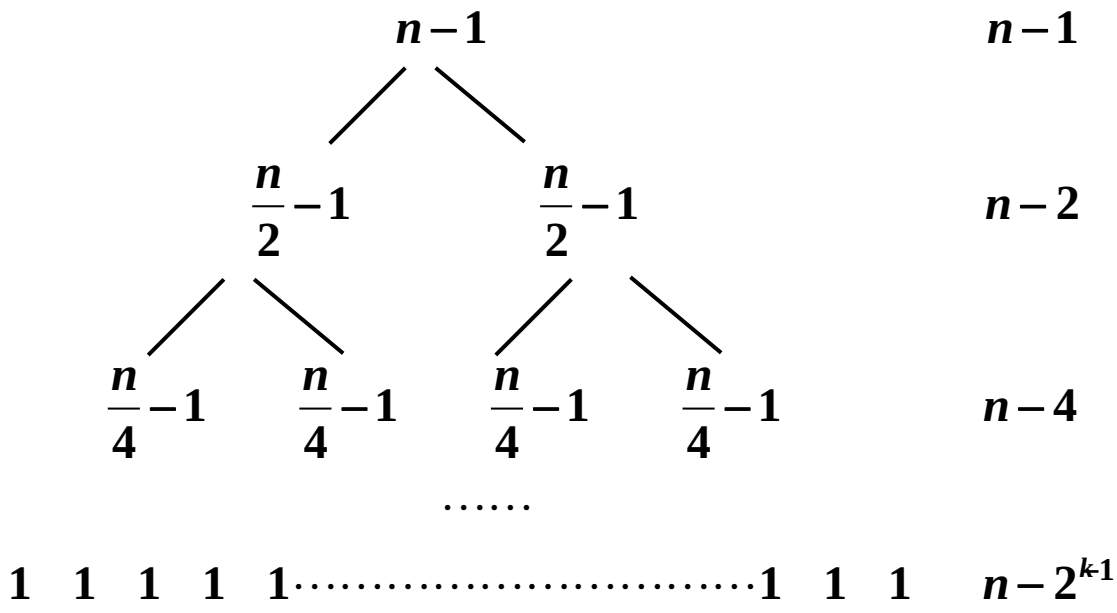
- 初始，递归树只有根结点，$W(n)$.
- 循环以下步骤直到树中没有函数项结点：将函数项叶结点的迭代式$W(m)$表示成二层子树（非函数项为根，函数项非叶结点）.



递归树解二分归并排序



生成过程



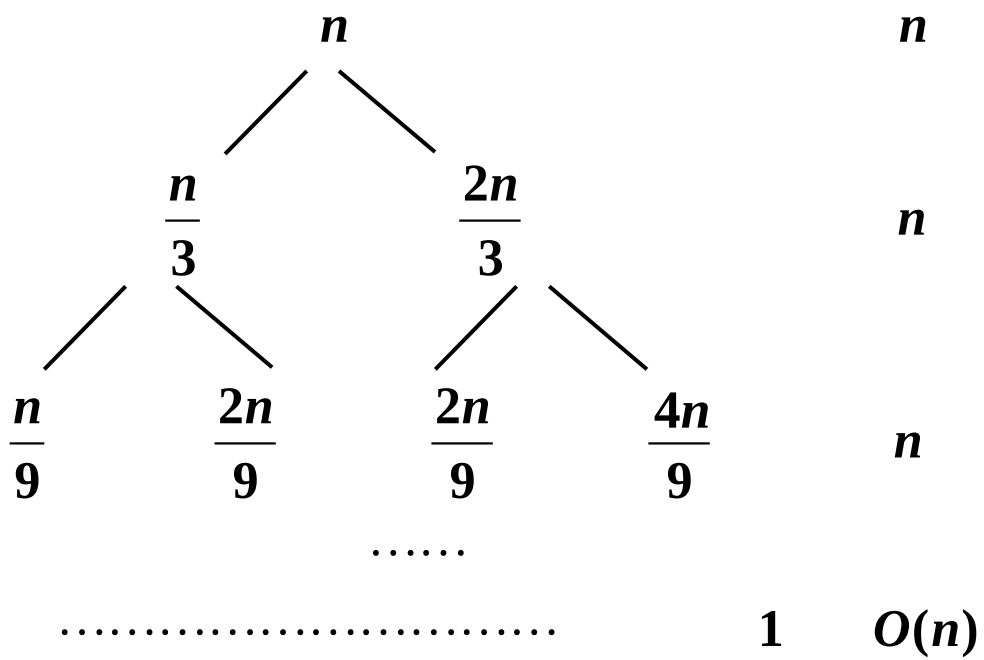
$W(n) = 2W(n/2) + n-1$，$n=2^k$，$W(2)=1$.

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

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

递归树的应用实例

求解：$T(n)=T(n/3)+T(2n/3)+n$



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

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

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$=cn$

设$O(n)$为c_0n, c_0不为0,则$c=c+c_0$不可能成立, 且左边<右边

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

(3) $T(n)=cn^2$，左边$=cn^2$

$c>2c/3$, 左边>右边

右边$=\frac{2}{n}\sum_{i=1}^{n-1}ci^2+O(n)=\frac{2}{n}[\frac{cn^3}{3}+O(n^2)]+O(n)=\frac{2c}{3}n^2+O(n)$

$$\sum_{i=1}^{n}i^2=\frac{1}{6}n(n+1)(2n+1)$$

(4) $T(n)=cn\log n$，左边$=cn\log n$

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

解：$T(n)=\Theta(n\log n)$

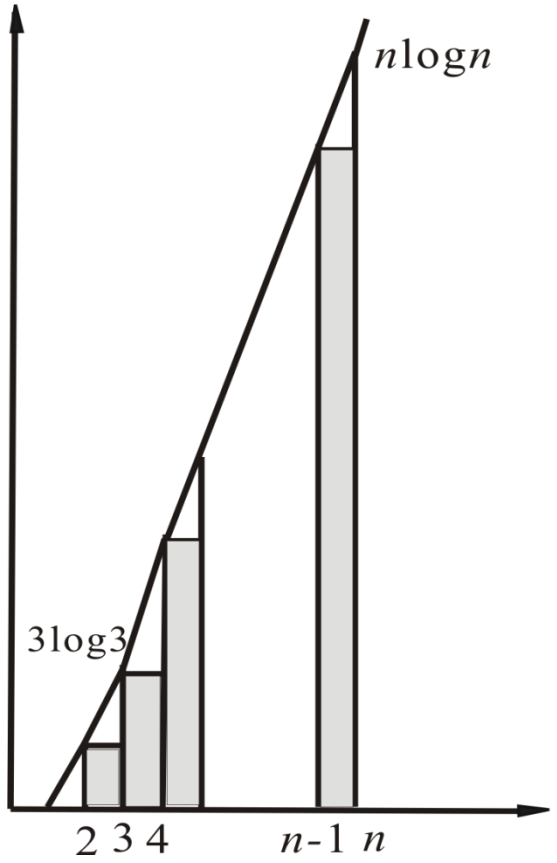
以积分作为求和的近似

$$\int_1^{n-1} x\log x dx \le \sum_{i=1}^{n-1} i\log i \le \int_2^n x\log x dx$$

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

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

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



主定理应用背景

适用于下面一类递推方程：

求解递推方程　$T(n) = a\,T(n/b) + f(n)$

a：　　归约后的子问题个数

n/b：归约后子问题的规模

$f(n)$：归约过程及组合子问题的解的工作量

例：

二分检索：　　$T(n) = T(n/2)+1$

二分归并排序：　$T(n) =2T(n/2)+n-1$

主定理

定理1.6 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$f(n)=\Omega(n^{\log_b a+\varepsilon}),\varepsilon>0$，且对某个常数$c<1$和所有充分大的$n$有 $af(n/b)\leq cf(n)$，那么

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

主定理的证明

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

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

不妨设$n=b^k$

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

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

$$= \ldots$$

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

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

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

情况1

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

1. 若$f(n)=O(n^{\log_b a-\varepsilon}),\varepsilon>0$，那么$T(n)=\Theta(n^{\log_b a})$

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

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

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

等比数列求和

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

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

$b^{\varepsilon \log_b n}=n^{\varepsilon}$

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

$$=c_1 n^{log_b a}+O\left(n^{log_b a}\right)=\Theta(n^{\log_b a})$$

情况2

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

2. 若$f(n)=\Theta(n^{\log_b a})$，那么$T(n)=\Theta(n^{\log_b a}\log n)$

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

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

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

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

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

提取与j无关的系数$n^{\log_b a}$，且$b^{j\log_b a}=a^{\log_b b^j}=a^j$

情况3

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

3. 若$f(n)=\Omega\left(n^{\log_b a+\varepsilon}\right),\varepsilon>0$,且对某个常数 c <1和所有充分大的 n 有 $a\,f(n/b)\le c\,f(n)$，那么$T(n)=\Theta(f(n))$

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

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

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

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

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

情况3

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

3. 若$f(n)=\Omega\left(n^{\log_b a+\varepsilon}\right),\varepsilon>0$,且对某个常数 c <1和所有充分大的 n 有 $af(n/b)\le cf(n)$，那么$T(n)=\Theta(f(n))$

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

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

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

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

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

又$T(n)=aT(n/b)+f(n)\ge f(n)=\Omega(f(n))$

所以$T(n)=\Theta(f(n))$

主定理的应用

设$a\geq 1$, $b>1$为常数，$f(n)$为函数，$T(n)$为非负整数，且
$T(n)=aT(n/b)+f(n)$

例1.16 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

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

1. 若$f(n)=O\left(n^{\log_b a-\varepsilon}\right), \varepsilon>0$，那么$T(n)=\Theta(n^{\log_b a})$

2. 若$f(n)=\Theta\left(n^{\log_b a}\right)$，那么 $T(n)=\Theta(n^{\log_b a}\log n)$

例1.17 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 .

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

应用实例

设$a\geq 1$, $b>1$为常数，$f(n)$为函数，$T(n)$为非负整数，且$T(n)=aT(n/b)+f(n)$

3. 若$f(n)=\Omega\left(n^{\log_b a+\varepsilon}\right),\varepsilon>0$, 且对某个常数 $c<1$和所有充分大的 n 有 $af(n/b)\leq cf(n)$，那么$T(n)=\Theta(f(n))$

例1.18 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 $af(n/b)\leq cf(n)$成立，代入 $f(n)=n\log n$，得到

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

显然只要 $c\geq 3/4$，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

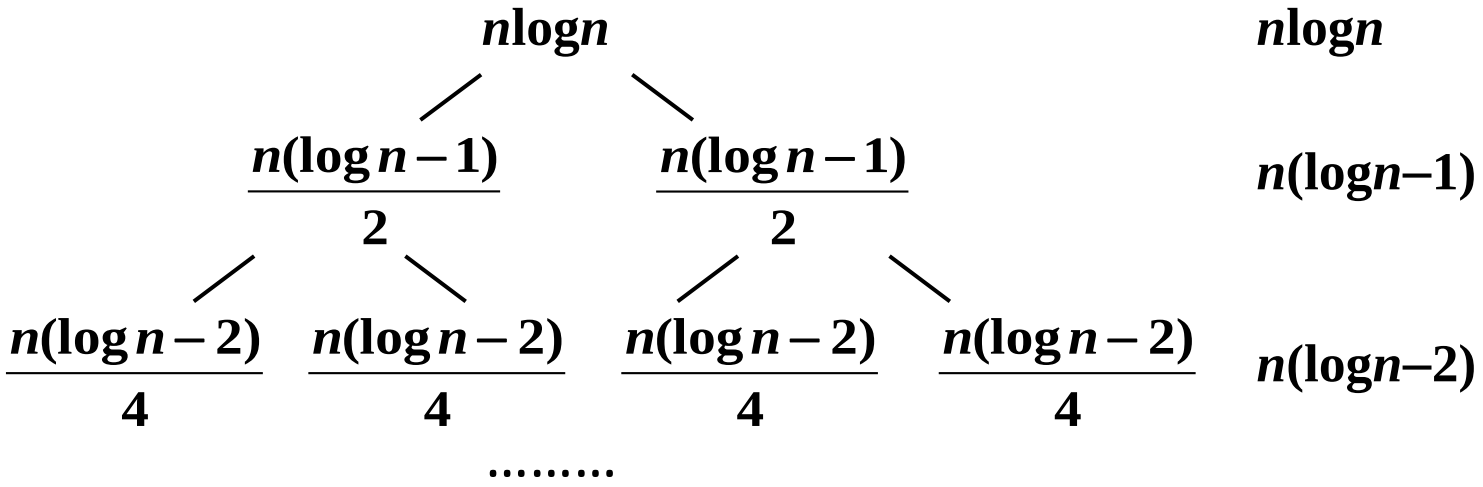
$$T(n)=\Theta(f(n))=\Theta(n\log n)$$

不能使用主定理的例子

例1.19 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$\varepsilon>0$ 使得 $n\log n=\Omega(n^{1+\varepsilon})$ 成立，



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

1.3.2&1.3.3小结

- 求解求和形式的T(n)表达式或估计其渐进的界
  - 等差级数、等比级数、调和级数等直接套公式
  - 一些结论：$\sum_{k=1}^{n-1}\frac{1}{k(k+1)}=1-\frac{1}{\mathrm{n}}$　$\sum_{t=1}^{k}t2^{t-1}=(k-1)2^{k}+1$
  - 放缩直接求级数渐进的界：$\sum_{k=1}^{n}a_k\le na_{\max}$　$\sum_{k=0}^{n}a_k\le\sum_{k=0}^{\infty}a_0r^k=a_0\sum_{k=0}^{\infty}r^k=\frac{a_0}{1-r}$
  - 利用对应函数的积分放缩
- 求解递推形式的T(n)表达式或估计其渐进的界
  - 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

第1章　知识回顾

1.1 有关算法的基本概念

几个例子

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

1.3.1 函数的渐近的界

1.3.2 求和的方法

1.3.3 递推方程求解方法

1.3.1

- **五种复杂度函数的渐进表示：** $O, \Omega, o, \omega, \Theta$
- **相关定理**
  - 定理1.1， 利用f/g的极限判断阶的高低
  - 定理1.2，阶的高底具有可传递性
  - 定理1.3，如果 $f=O(h)$ 且 $g=O(h)$，那么 $f + g = O(h)$
- **一些重要函数阶的高低**
  - 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： $\log n, \log^2 n, \log\log n, \ldots$

1.3.2&1.3.3

- 求解求和形式的T(n)表达式或估计其渐进的界
  - 等差级数、等比级数、调和级数等直接套公式
  - 一些结论：$\sum_{k=1}^{n-1}\frac{1}{k(k+1)}=1-\frac{1}{n}$ $\sum_{t=1}^{k}t2^{t-1}=(k-1)2^{k}+1$
  - 放缩直接求级数渐进的界：$\sum_{k=1}^{n}a_k\le na_{\max}$ $\sum_{k=0}^{n}a_k\le\sum_{k=0}^{\infty}a_0r^k=a_0\sum_{k=0}^{\infty}r^k=\frac{a_0}{1-r}$
  - 利用对应函数的积分放缩
- 求解递推形式的$T(n)$表达式或估计其渐进的界
  - 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
    - 比较$n^{\log_b a}$和$f(n)$的阶，谁的阶高（必须高出n^{ε}），$T(n)$与谁同阶
    - 若他们同阶，$T(n)$与其×$\log n$同阶