算法设计与分析

第1章 算法概述（3）

学习要点

- 算法在计算机科学中的地位
- 算法的概念
- 算法分析
- 算法的计算复杂性概念
- 算法渐近复杂性的数学表述

算法的渐近复杂性

- $T(n) \to \infty$， as $n \to \infty$；
- 若存在t(n), 使得$(T(n) - t(n))/ T(n) \to 0$ ，as $n \to \infty$，则称t(n)是T(n)的渐近性态，为算法的渐近复杂性。
- 在数学上， t(n)是T(n)的渐近表达式，它比T(n) 简单, 是T(n)略去低阶项留下的主项
  - $T(n)=3n^2+4n\log n+7$
  - $t(n)=3n^2$ 说：算法运行时间是$O(n^2)$
- 复杂性分析的简化
  - 用渐近复杂性t(n)代替T(n)

算法的渐近复杂性

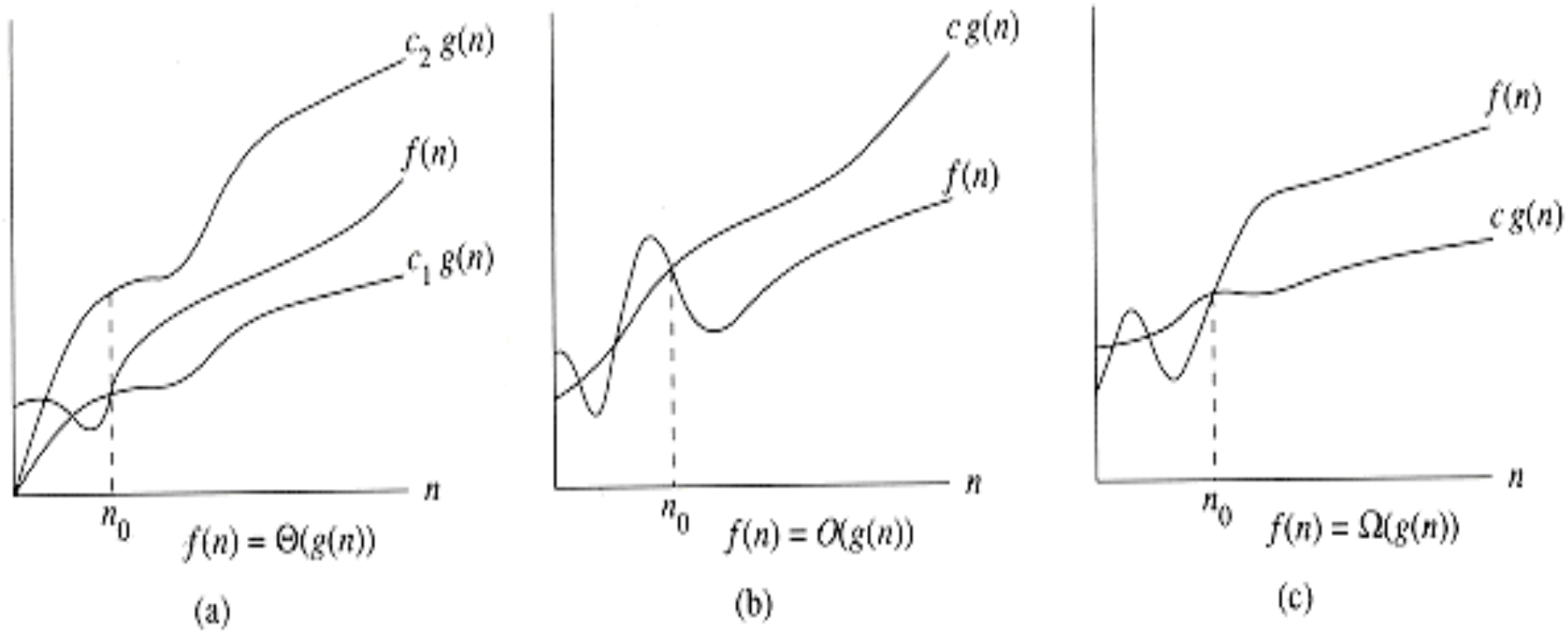
- 渐进复杂性分析的进一步简化
  - 只关心“阶”,不关心常数因子
  - 假设算法中所有不同的基本运算各执行一次所需时间均为一个单位时间
- 结论:
  - 渐近分析的思维：只考虑当问题的规模充分大时,算法复杂性在某些代表性输入的渐进意义下的阶

阶：次数

- 例如: $17n^2+24n\log_2 n+37n+256$
  - $O(n^2)$
- $5n^2$与$9999n$ ？

渐近分析的记号(1)

时间复杂度



问题规模

渐近分析的记号(2)

- 在下面的讨论中, 对所有n，f(n) ≥ 0，g(n) ≥ 0。

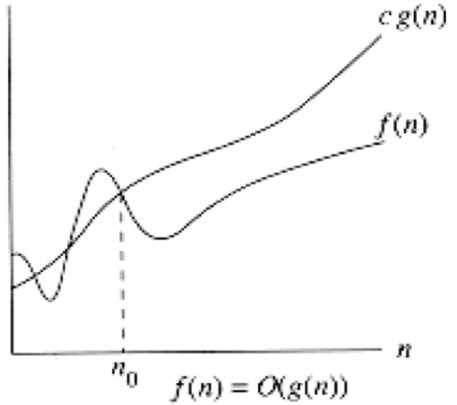
f(n),g(n)是定义在正数集上的正函数

（1）渐近上界记号O

- O(g(n)) = { f(n) |存在正常数c和n_0,使得对所有n≥n_0有：0≤f(n) ≤cg(n) }
  - Eg: 插入排序 最坏情况f(n)=O(n^2)

函数集合

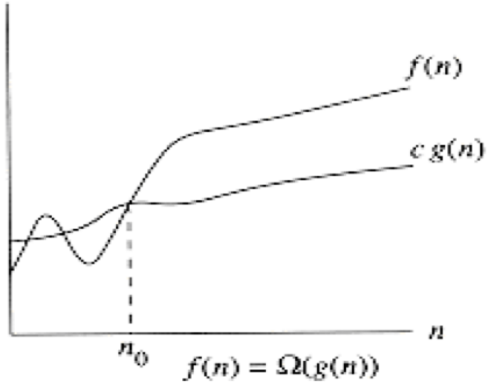
f(n)上有界,g(n)是上界
f(n)的阶不高于g(n)的阶



（2）渐近下界记号Ω

- Ω (g(n)) = { f(n)| 存在正常数c和n_0,使得对所有n≥ n_0有：0≤cg(n) ≤ f(n) }

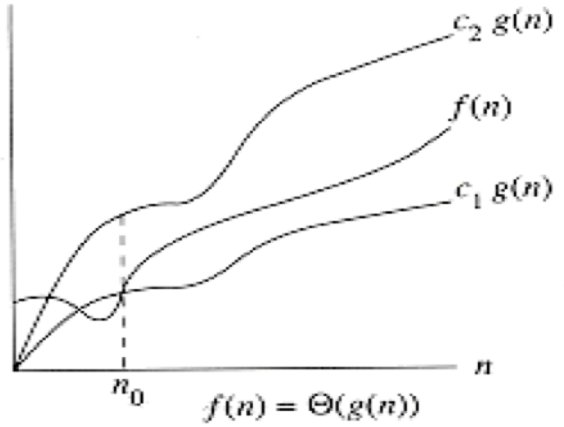
f(n)下有界,g(n)是下界



渐近分析的记号(3)

（3）紧渐近界记号Θ

- Θ (g(n)) = { f(n) | 存在正常数c_1,c_2和n_0,使得对所有n≥n_0,有：$c_1 g(n) \le f(n) \le c_2 g(n)$ }
  - f(n)=O(g(n))且f(n)= Ω(g(n))



- 定理1： Θ (g(n)) = O (g(n)) ∩ Ω (g(n))
  - 实际中，通常根据渐近上界和渐近下界来证明渐近紧界，而不是根据渐近紧界来得到渐近上界和渐近下界。

f(n)和 g(n)相同程度的渐进增长

渐近分析的记号(4)

（4）非紧上界记号o

- o(g(n)) = { f(n) | 对于任何正常数c>0，存在正整数n_0 >0使得对所有n≥ n_0有：0 ≤ f(n)<cg(n) }
- 等价于 f(n) / g(n) →0 ，as n→∞。
  - Eg: $2n=o(n^2)$, 但是$2n^2 \neq o(n^2)$

（5）非紧下界记号ω

- ω (g(n)) = { f(n) | 对于任何正常数c>0，存在正整数n_0 >0使得对所有n≥ n_0有：0 ≤ cg(n) < f(n) }
- 等价于 f(n) / g(n) →∞ ，as n→∞。
- f(n) ∈ ω (g(n)) ⇔ g(n) ∈ o (f(n))

渐近分析记号在等式和不等式中的意义

- $f(n)= \Theta(g(n))$的确切意义是：$f(n) \in \Theta(g(n))$

等号表示集合的成员关系

- 一般情况下，等式和不等式中的渐近记号$\Theta(g(n))$表示$\Theta(g(n))$中的某个函数
  - 例如：$2n^2 + 3n + 1 = 2n^2 + \Theta(n)$ 表示

    $2n^2 + 3n + 1 = 2n^2 + f(n)$，其中$f(n)$ 是$\Theta(n)$中某个函数

越规使用

- 等式和不等式中渐近记号O,o, Ω和ω的意义是类似的
- 工程做法：舍去低阶项,忽略后续的常量
  - 将函数中所有的加法项常数都去掉
  - 在修改后的函数中,只保留最高阶项
  - 去除高阶项前面的系数
  - Example: $3n^3 + 90n^2 - 5n + 6046 = \Theta(n^3)$

渐近分析的记号

- O、 Ω和 Θ分别表示算法的上界、下界和渐进界，可以用来描述算法在最坏、最好以及平均情况下的行为。

- 如何得到一个函数的上界、下界？
  - 寻找参数

渐近分析记号- Big Oh 例子

- 例1. $2n^2+11n-10 = O(n^2)$
  - 证：因为$2n^2+11n-10 \le 3n^2$（当$n \ge 10$）
    存在$c=3$和$n \ge 10$ 使得$f(n) \le 3n^2$
- 例2. $100n+5 = O(n^2)$
  - 证：因为 $100n+5 \le 100n+n$(当$n \ge 5$)$=101n$
    $\le 101n^2$
    存在$c=101$和$n \ge 5$时 $f(n) \le 101n^2$
- 例3. $3n^2-100n+6 \neq O(n)$
  - $3n^2-100n+6 < cn$?
  - 找反例：因为当 $n>c$ 时 $c*n<n^2<3n^2$

渐近分析记号-Big Omega例子

- 例4. 证明：$n^3 = \Omega(n^2)$

渐近分析记号-Big Theta例子

- 例5. 证明$\frac{1}{2}n^2 - 3n = \Theta(n^2)$
  - 分析：因为O and Ω
  - 证：首先要确定常数c_1, c_2 和n_0 ，使得对于所有$n \geq n_0$, 有：

$$c_1 n^2 \leq \frac{1}{2}n^2 - 3n \leq c_2 n^2 \text{成立}$$

$$c_1 \leq \frac{1}{2} - \frac{3}{\mathrm{n}} \leq c_2$$

  - 上界： $c_2 \geq \frac{1}{2}$时，右边不等式对所有n ≥1成立
  - 下界： $c_1 \leq \frac{1}{14}$时，左边不等式对所有n ≥ 7成立
  - 通过选择$c_1 = \frac{1}{14}$， $c_2 = \frac{1}{2}$以及$n_0 = 7$时，证得 $\frac{1}{2}n^2 - 3n = \Theta(n^2)$

渐近分析中函数比较

- 两个函数的渐近比较和两个实数的比较之间的类比关系:
- $f(n) = O(g(n)) \approx a \leq b$;
- $f(n) = \Omega(g(n)) \approx a \geq b$;
- $f(n) = \Theta(g(n)) \approx a = b$;
- $f(n) = o(g(n)) \approx a < b$;
- $f(n) = \omega(g(n)) \approx a > b$.

渐近分析记号的若干性质(1)

（1）**传递性**：

- $f(n)= \Theta(g(n))$，$g(n)= \Theta(h(n)) \Rightarrow f(n)= \Theta(h(n))$
- $f(n)= O(g(n))$，$g(n)= O(h(n)) \Rightarrow f(n)= O(h(n))$
- $f(n)= \Omega(g(n))$，$g(n)= \Omega(h(n)) \Rightarrow f(n)= \Omega(h(n))$
- $f(n)= o(g(n))$，$g(n)= o(h(n)) \Rightarrow f(n)= o(h(n))$
- $f(n)= \omega(g(n))$，$g(n)= \omega(h(n)) \Rightarrow f(n)= \omega(h(n))$

渐近分析记号的若干性质(2)

（2）反身性：

- $f(n)= \Theta(f(n))$
- $f(n)= O(f(n))$
- $f(n)= \Omega(f(n))$

（3）对称性：

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

（4）互对称性：

- $f(n)= O(g(n)) \Leftrightarrow g(n)= \Omega(f(n))$
- $f(n)= o(g(n)) \Leftrightarrow g(n)= \omega(f(n))$

渐近分析记号的若干性质(3)

（5）算术运算：

- $O(f(n))+O(g(n)) = O(\max\{f(n),g(n)\})$ 只取高阶项
- $O(f(n))+O(g(n)) = O(f(n)+g(n))$
- $O(f(n))*O(g(n)) = O(f(n)*g(n))$
- $O(cf(n)) = O(f(n))$ 去掉系数
- $g(n)= O(f(n)) \Rightarrow O(f(n))+O(g(n)) = O(f(n))$

渐近分析记号的若干性质(4)

- $O(f(n))+O(g(n)) = O(\max\{f(n),g(n)\})$

证明: $t_1(n)=O(f(n))$, $t_2(n)=O(g(n))$, 有

存在常量c_1和n_1, 使得: 对于所有的$n \geq n_1$, $t_1(n) \leq c_1 f(n)$

存在常量c_2和n_2, 使得: 对于所有的$n \geq n_2$, $t_2(n) \leq c_2 g(n)$

设$c_3=\max\{c_1,c_2\}$,且$n \geq \max\{n_1,n_2\}$, 有

$$t_1(n)+t_2(n) \leq c_1 f(n)+c_2 g(n)$$
$$\leq c_3 f(n)+c_3 g(n)=c_3[f(n)+g(n)]$$
$$\leq c_3\ 2\max\{f(n),g(n)\}$$

因此有:

$$t_1(n)+t_2(n) \in O(\max\{f(n),g(n)\}),$$

满足符号O定义的c和n_0的值分别为

$$2c_3=2\max\{c_1,c_2\},\quad n_0=\max\{n_1,n_2\}$$

对于4个任意实数a1,a2,b1,b2, 如果a1 ≤ b1且a2 ≤ b2, 则a1+a2 ≤ 2max{b1,b2}

算法渐近复杂性分析中常用函数

（1）单调函数

- 单调递增：$m \le n \Rightarrow f(m) \le f(n)$;
- 单调递减：$m \le n \Rightarrow f(m) \ge f(n)$;
- 严格单调递增：$m < n \Rightarrow f(m) < f(n)$;
- 严格单调递减：$m < n \Rightarrow f(m) > f(n)$.

（2）取整函数

- $\lfloor x \rfloor$：不大于x的最大整数；
- $\lceil x \rceil$：不小于x的最小整数。

算法渐近复杂性分析中常用函数

（3）多项式函数

n的d次多项式是具有如下形式的函数

- $p(n)= a_0+a_1n+a_2n^2+\ldots+a_dn^d$ ； $a_d>0$;
- $p(n) = \Theta(n^d)$;
- $f(n) = O(n^k) \Leftrightarrow$ f(n)有多项式界；
- $f(n) = O(1) \Leftrightarrow f(n) \le c$;
- $k \ge d \Rightarrow p(n) = O(n^k)$;
- $k \le d \Rightarrow p(n) = \Omega(n^k)$;
- $k > d \Rightarrow p(n) = o(n^k)$;
- $k < d \Rightarrow p(n) = \omega(n^k)$.

算法渐近复杂性分析中常用函数

（4）指数函数

- 对于正整数m,n和实数a>0:
- $a^0=1$;
- $a^1=a$;
- $a^{-1}=1/a$;
- $(a^m)^n = a^{mn}$;
- $(a^m)^n = (a^n)^m$;
- $a^m a^n = a^{m+n}$;
- a>1 $\Rightarrow$ a^n为单调递增函数;
- a>1 $\Rightarrow$ $\lim_{n\to\infty} \frac{n^b}{a^n} = 0$ $\Rightarrow$ $n^b = o(a^n)$

任何底大于1的指数函数比任何多项式函数增长得更快

算法渐近复杂性分析中常用函数

（5）对数函数

- log n
- lg n = $\log_2 n$
- ln N = $\log_e n$;
- $\log^k n = (\log n)^k$
- log log n = log(log n);
- for a>0,b>0,c>0 和n
  - $\log^b n = O(n^a)$

$$\log_b n = \Theta(\log_a n)$$

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

$$\log_b a = \frac{\log_c a}{\log_c b}$$

$$\lim_{n\to\infty} \frac{\log^b n}{(2^a)^{\log n}} = \lim_{n\to\infty} \frac{\log^b n}{n^a} = 0$$

多项式函数比多项式对数函数增长得更快

算法渐近复杂性分析中常用函数

（6）阶乘函数

$$n!=\begin{cases}1 & n=0\\ n(n-1)! & n>0\end{cases}$$

- Stirling’ s approximation

$$n!=1\cdot 2\cdot 3\cdots n$$

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

$$n!=o(n^n) \qquad n!=\omega(2^n) \qquad \log(n!)=\Theta(n\log n)$$

算法分析中常用的复杂性函数

FUNCTION	NAME
c	Constant
$\log N$	Logarithmic
$\log^2 N$	Log-squared
N	Linear
$N \log N$	N log N
N^2	Quadratic
N^3	Cubic
2^N	Exponential
$N!$	*阶乘*
N^N	

效率最高

扫描规模为N的列表

分治算法

三重嵌套循环

求n个元素集合的所有子集

求n个元素集合的完全排列

算法分析中常见的复杂性函数

- 按照时间复杂度排序复杂性函数?

$$O(1) < O(logn) < O(n) < O(nlogn) < O(n^2) < O(n^3) < O(2^n) < O(n!) < O(n^n)$$

- 简化经验规则
  - 常系数可以省略
  - 当a>b时，n^a支配n^b
  - 任何指数项支配任何多项式项： 3^n支配n^5
  - 任何多项式项支配对数项：n支配$(logn)^3$

复杂性函数排序的启示

- 陈国良院士的大数据计算理论研究
  - 线性< 多项式对数 <多项式

最优算法

- 任何一个求解问题a的算法的计算时间下界为Ω(f(n))，则计算时间复杂性在O(f(n))时间内求解问题的任何算法称为问题a的**最优算法**。
- 例如：排序问题的计算时间下界为Ω(nlogn)， 计算时间复杂性为O(nlogn)的排序算法是最优算法。
  - 堆排序算法是最优算法。

小结

- 主要内容
  - 渐进复杂性的思想
    - 问题的规模充分大时,算法复杂性在某些代表性输入的渐进意义下的阶
  - 算法渐近复杂性的数学表述
    - 渐进分析记号的定义及其性质
    - 渐进复杂性分析常用函数

- 重点和难点
  - 渐近复杂性的数学表述方法
  - 渐进表达式的写法

本章小结

- 算法在计算机科学中的地位
- 算法的概念和特征
- 算法分析的数学工具和方法
- 算法的计算复杂性概念
- 算法渐近复杂性的数学表述

创新实践能力

- 利用算法分析的结果改进算法
  - $\boldsymbol{O(1) < O(logn) < O(n) < O(nlogn) < O(n^2) < O(n^3) < O(2^n) < O(n!) < O(n^n)}$

扩展题

- 作业：总结对各种排序算法（插入排序，冒泡排序，合并排序，快速排序）的特点，能够从额外空间消耗、平均时间复杂度和最坏时间复杂度等方面进行比较，分析优缺点。

- 问题：请实现一个排序算法，要求时间效率为O(n).
  - 明确：排序应用的环境是什么，有哪些约束条件，在此基础上选择合适的排序算法