Lecture 2 复杂度分析的数学基础

绳伟光

上海交通大学微纳电子学系

2021-09-16

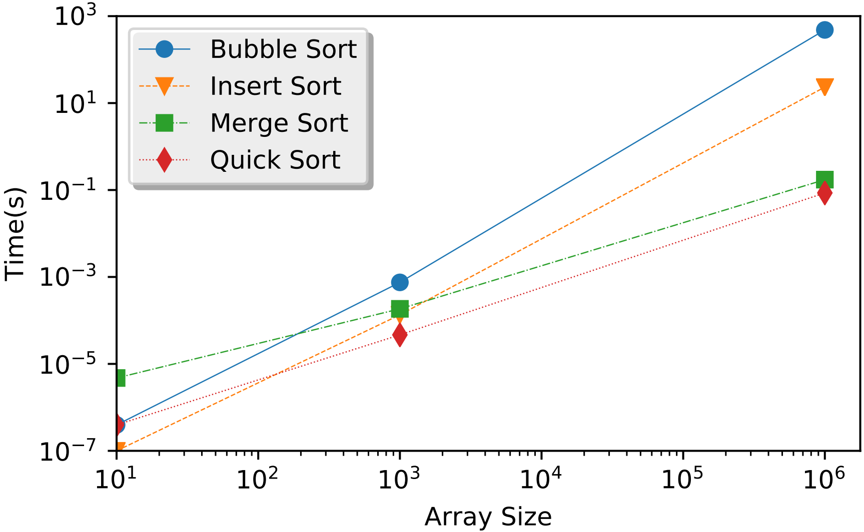
提纲

1. 复杂度函数的阶
2. 标准符号和通用函数
3. 分治算法的递归方程
4. * 一般递归方程求解 (自学)

提纲

1. 复杂度函数的阶
2. 标准符号和通用函数
3. 分治算法的递归方程
4. * 一般递归方程求解 (自学)

典型排序算法的性能比较



不同排序算法在数据量不同时性能差距非常大，该如何比较不同算法的性能?

算法性能比较的难点

算法性能受多种因素影响：

- 机器性能
- 程序编制者的水平
- 编译器的质量
- 算法往往在小数据量和大数据量时表现也不同

可行的解决方案：比较算法的渐进复杂度函数

复杂度函数

复杂度分析将算法的复杂度表达成问题输入规模 n 的函数 $f(n)$，复杂度函数具有如下意义：

- 比较求解同一问题的不同算法的性能
- 有了复杂度函数，可以研究算法性能随问题大小增长的变化，特别是当问题足够大时性能的变化

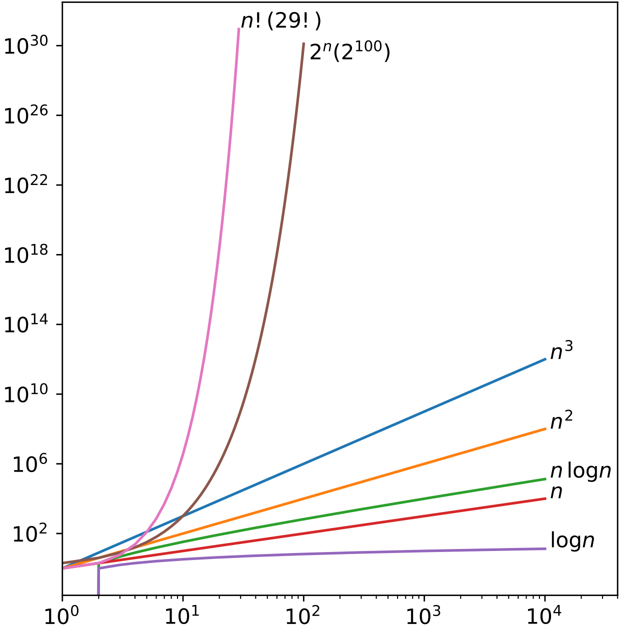
渐进复杂度： $n \to +\infty$ 时算法复杂度函数 $f(n)$ 的增长规律称为算法的渐进复杂度

示例： 比较如下复杂度函数 f(n) 和 g(n)：

$$f(n) = 10n\log n + 5n + 8$$

$$g(n) = 0.001n^2 + n + 10$$

常见函数的渐进趋势



$n!\,(29!)$
$2^n(2^{100})$
n^3
n^2
$n\log n$
n
$\log n$
10^{30}
10^{26}
10^{22}
10^{18}
10^{14}
10^{10}
10^{6}
10^{2}
10^0
10^1
10^2
10^3
10^4

函数渐进增长对比示例

1. **比较 $\log n$ 和 $\sqrt{n}$**

$$\lim_{n\to\infty}\frac{\log n}{\sqrt{n}}=\lim_{n\to\infty}\frac{(\log n)'}{(\sqrt{n})'}=2\lim_{n\to\infty}\frac{(\log e)\frac{1}{n}}{1/\sqrt{n}}=2\log e\lim_{n\to\infty}\frac{1}{\sqrt{n}}=0$$

因此，$\log n$ 的增长慢于 $\sqrt{n}$

2. **比较 $n!$ 和 2^n**

$$\lim_{n\to\infty}\frac{n!}{2^n}=\lim_{n\to\infty}\frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{2^n}=\lim_{n\to\infty}\sqrt{2\pi n}\frac{n^n}{2^n e^n}=\lim_{n\to\infty}\sqrt{2\pi n}\left(\frac{n}{2e}\right)^n$$

上式极限为 ∞，因此，$n!$ 的增长快于 2^n

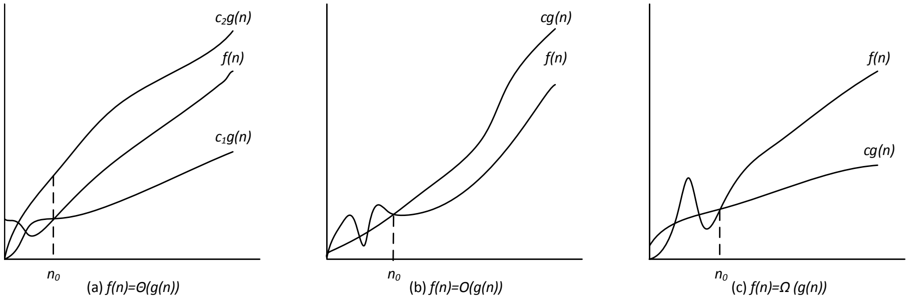
斯特林公式：当 $n\to\infty, n!\approx\sqrt{2\pi n}\left(\frac{n}{e}\right)^n$

复杂度函数

设某一输入大小为 n 的算法其工作量 (效率) 用关于 n 的复杂度函数 $f(n)$ 来表示，则我们将 $f(n)$ 与另一行为已知的函数 $g(n)$ 在 n 趋向无穷大的时候的情况进行对比，有助于我们形成对 $f(n)$ 的复杂度的认识：

- 如果 $\lim_{n\to\infty} f(n)/g(n) = 0$，则称 $f(n)$ 在数量级上严格小于 $g(n)$，记为 $f(n) = o(g(n))$
- 如果 $\lim_{n\to\infty} f(n)/g(n) = \infty$，则称 $f(n)$ 在数量级上严格大于 $g(n)$，记为 $f(n) = \omega(g(n))$
- 如果 $\lim_{n\to\infty} f(n)/g(n) = c$，此处 c 为非 0 常数，则称 $f(n)$ 在数量级上等于 $g(n)$，即 $f(n)$ 和 $g(n)$ 是同一个数量级的函数，记为 $f(n) = \Theta(g(n))$
- 如果 $f(n)$ 在数量级上小于或等于 $g(n)$，则记为 $f(n) = O(g(n))$
- 如果 $f(n)$ 在数量级上大于或等于 $g(n)$，则记为 $f(n) = \Omega(g(n))$

算法复杂度的渐进符号表示图例



上述 3 子图分别对应**同阶函数集合、低阶函数集合、高阶函数集合**

函数阶的概念反映了其数量级的特点，并将复杂度分析时忽略低阶项和常数因子，只关注高阶项这一特点突出出来！

复杂度函数的阶 —— 同阶函数集合

同阶函数集合

$\Theta(g(n)) = \{f(n) | \exists c_1, c_2 > 0$ 以及常数 n_0, 当 $n \geq n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)\}$ 称为与 $g(n)$ 同阶的函数集合

- 如 $f(n) \in \Theta(g(n))$，则称 $f(n)$ 与 $g(n)$ 同阶；$f(n)$ 是 $\Theta(g(n))$ 一员
- 简单起见，将 $f(n) \in \Theta(g(n))$ 记作 $f(n) = \Theta(g(n))$，实际上是用已知函数 $g(n)$ 来刻画 $f(n)$ 的界限
- $\lim_{n\to+\infty} g(n) > 0$，即 n 充分大的时候，$g(n)$ 必须是极限非负的，否则 $\Theta(g(n)) = \emptyset$

同阶函数集合实例

证明 $f(n) = (1/2)n^2 - 3n = \Theta(n^2)$

证　设 $n \geq n_0$ 时有 $c_1 n^2 \leq (1/2)n^2 - 3n \leq c_2 n^2$，两边除以 n^2，有 $c_1 \leq 1/2 - 3/n \leq c_2$，由于 $1/2 - 3/n \leq 1/2$ 对任何 $n \geq 1$ 成立，可取 $c_2 = 1/2$；同样对任何 $n \geq 7$，$1/2 - 3/n \geq 1/14$ 成立，可取 $c_1 = 1/14$，得证。 □

证明 $6n^3 \neq \Theta(n^2)$

证　反证法，如果存在 $c_1, c_2 > 0, n_0$，使得 $n \geq n_0$ 时有 $c_1 n^2 \leq 6n^3 \leq c_2 n^2$，假设 $n_0 = c_2/6$ ，则有 $n > c_2/6$ 的同时 $n \leq c_2/6$，矛盾。 □

同阶函数集合实例

证明 $c = \Theta(1)$

对于任何常数有 $c = \Theta(n^0) = \Theta(1)$，因为 $\exists c_1 = (1/2)c, c_2 = (3/2)c, n_0 = 1$，使得 $c_1 \le c \le c_2$。

$\Theta(1)$ 称为常数复杂度，复杂度为 $\Theta(1)$ 并不意味着无论给算法输入什么它都会花相同的时间，准确的含义是复杂度的界与其输入无关

证明 $f(n) = an^2 + bn + c = \Theta(n^2)$

证 设 $c_1 n^2 \le an^2 + bn + c \le c_2 n^2$，可令 $c_1 = a/4, c_2 = 7a/4$，则 $a/4 \cdot n^2 \le an^2 + bn + c \le 7a/4 \cdot n^2$，令 $n_0 = 2 \cdot max\left(|b|/a, \sqrt{|c|/a}\right)$，则当 $n \ge n_0$ 时 $c_1 n^2 \le an^2 + bn + c \le c_2 n^2$ 成立。 $\square$

多项式复杂度的界

命题

对于任意正整数 d 和任意常数 $a_d > 0$，d 次多项式 $p(n) = \sum_{i=0}^{d} a_i n^i = \Theta(n^d)$

证 令 $c_1 = \frac{1}{d+1}a_d, c_2 = \frac{2d+1}{d+1}a_d, n_0 = \max\left\{ \sqrt[d-i]{(d+1)\frac{|a_i|}{a_d}} \middle| 0 \le i \le d-1 \right\}$

当 $n > n_0$，有 $\sum_{i=0}^{d} a_i n^i - c_1 n^d = \sum_{i=0}^{d-1} \frac{a_d}{d+1} n^i \left(n^{d-i} + (d+1)\frac{a_i}{a_d} \right) \ge 0$

上式成立的关键在 $\frac{d}{d+1}a_d n^d = \sum_0^{d-1} \frac{1}{d+1} a_d n^d$。

同理可得 $c_2 n^d - \sum_{i=0}^{d} a_i n^i = \sum_{i=0}^{d-1} \frac{a_d}{d+1} n^i \left(n^{d-i} - (d+1)\frac{a_i}{a_d} \right) \ge 0$

即 $n > n_0$ 后恒有 $c_1 n^d \le \sum_{i=0}^{d} a_i n^i \le c_2 n^d$。

此处证明不做要求，仅需记住结论

复杂度函数的阶 —— 低阶函数集合

低阶函数集合

$O(g(n)) = \{f(n) | \exists c > 0, n_0,$ 当$n \geq n_0, 0 \leq f(n) \leq cg(n)\}$ 称为比 $g(n)$ 低阶的函数集合

- 如果$f(n) \in O(g(n))$，则称 $g(n)$ 是$f(n)$ 的上界
- $f(n) \in O(g(n))$ 通常记作$f(n) = O(g(n))$
- $f(n) = \Theta(g(n)) \Rightarrow f(n) = O(g(n))$，由此可得 $\Theta(g(n)) \subseteq O(g(n))$

证明 $n = O(n^2)$

令 $c = 1, n_0 = 1$，则当 $n \geq n_0$ 时有 $0 \leq n \leq cn^2$。

复杂度函数的阶 —— 高阶函数集合

高阶函数集合

$\Omega(g(n)) = \{f(n) | \exists c > 0, n_0$, 当$n \geq n_0, 0 \leq cg(n) \leq f(n)\}$ 称为比 $g(n)$ 高阶的函数集合

- 如果$f(n) \in \Omega(g(n))$，则称 $g(n)$ 是$f(n)$ 的下界。
- $f(n) \in \Omega(g(n))$ 通常记作$f(n) = \Omega(g(n))$。

定理（对于任意$f(n)$和$g(n)$，$f(n) = \Theta(g(n))$ iff $f(n) = O(g(n))$且$f(n) = \Omega(g(n))$。）

证 $\Rightarrow$ 如果$f(n) = \Theta(g(n))$，则$\exists c_1, c_2 > 0, n_0 > 0$，当$n \geq n_0$，有$c_1 g(n) \leq f(n) \leq c_2 g(n)$，显然$f(n) = O(g(n))$且$f(n) = \Omega(g(n))$。

$\Leftarrow$ 由$f(n) = O(g(n))$, $\exists c_1, n_1 > 0$，使得当$n \geq n_1$，有$f(n) \leq c_1 g(n)$。

由$f(n) = \Omega(g(n))$, $\exists c_2, n_2 > 0$，使得当$n \geq n_2$，有$0 \leq c_2 g(n) \leq f(n)$。

因此，取$n_0 \geq \max(n_1, n_2)$，当$n \geq n_0$，$c_2 g(n) \leq f(n) \leq c_1 g(n)$，得证。 $\square$

证明 $\log n! = \Theta(n \log n)$

证 首先，$\log n! = \sum_{i=1}^{n} \log i \le n \log n$，可知 $\log n! = O(n \log n)$。

其次，对于 $\log n!$，只考虑后半部分 $[n/2, n]$，可得

$\log n! > \log(n/2)^{n/2}$ 在 $n \ge 4$ 时恒成立，

即 $\log n! > \log(n/2)^{n/2} = \dfrac{n}{2} \log \dfrac{n}{2} \ge \dfrac{1}{4} n \log n$，

即 $\log n! = \Omega(n \log n)$。

由上页定理可知，$\log n! = \Theta(n \log n)$。 $\square$

复杂度函数的阶 —— 严格低阶函数集合

严格低阶函数集合

$o(g(n)) = \{f(n) | \forall c > 0, \exists n_0 > 0,$ 当 $n \geq n_0, 0 \leq f(n) \leq cg(n)\}$ 称为比 $g(n)$ 严格低阶的函数集合

- 如果 $f(n) \in o(g(n))$，则称 $g(n)$ 是 $f(n)$ 的严格上界。
- $f(n) \in o(g(n))$ 通常记作 $f(n) = o(g(n))$。

证明 $2n = o(n^2)$

证 对 $\forall c > 0$，令 $n_0 = 2/c$，则当 $n \geq n_0$ 时对 $\forall c > 0$ 有 $0 \leq 2n \leq cn^2$。 □

严格低阶函数集合要求对任意常数 c 成立，这是需特别注意的地方！小 o 记号一般很少用！

严格低阶函数集合实例

证明 $2n^2 \neq o(n^2)$

证 令 $c=1$，对 $\forall n_0>0$，当 $n\geq n_0$，$2n^2\leq cn^2$ 都不成立。 □

$$f(n)=o(g(n))\Rightarrow \lim_{n\to\infty}\frac{f(n)}{g(n)}=0$$

证 由于 $f(n)=o(g(n))$，对 $\forall\varepsilon>0$，$\exists n_0>0$，当 $n\geq n_0$ 时有 $0\leq f(n)\leq\varepsilon g(n)$，即 $0\leq\dfrac{f(n)}{g(n)}\leq\varepsilon$，可得 $\lim\limits_{n\to\infty}\dfrac{f(n)}{g(n)}=0$。 □

复杂度函数的阶 —— 严格高阶函数集合

严格高阶函数集合

$\omega(g(n)) = \{f(n) | \forall c > 0, \exists n_0 > 0,$ 当 $n \geq n_0, 0 \leq cg(n) \leq f(n)\}$ 称为比 $g(n)$ 严格高阶的函数集合

定理 ($f(n) \in \omega(g(n))$ *iff* $g(n) \in o(f(n))$)

证 $\Rightarrow$ 对 $\forall c > 0$，有 $1/c > 0$，由 $f(n) \in \omega(g(n))$，对 $1/c > 0$，$\exists n_0 > 0$，当 $n \geq n_0$ 时 $(1/c)g(n) \leq f(n)$，即 $g(n) \leq cf(n)$，于是 $g(n) \in o(f(n))$。

$\Leftarrow$ 对 $\forall c > 0$,有 $1/c > 0$,由 $g(n) \in o(f(n))$,对 $1/c > 0$,$\exists n_0 > 0$,当 $n \geq n_0$ 时 $g(n) \leq (1/c)f(n)$，即 $cg(n) \leq f(n)$，于是 $f(n) \in \omega(g(n))$。 $\square$

严格高阶函数集合 —— 实例

证明 $n^2/2 = \omega(n)$

证 对 $\forall c > 0$，欲证 $cn \le n^2/2$，只需 $n \ge 2c$，则令 $n_0 = 2c + 1$，当 $n \ge n_0$ 时有 $cn \le n^2/2$ 成立。 □

证明 $n^2/2 \ne \omega(n^2)$

证 若 $n^2/2 = \omega(n^2)$，则对 $c = 1/2$，$\exists n_0 > 0$，当 $n \ge n_0$ 时有 $cn^2 \le n^2/2 \Rightarrow c < 1/2$，矛盾。 □

$$f(n) = \omega(g(n)) \Rightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

证 由于 $f(n) = \omega(g(n))$，对 $\forall c > 0$，$\exists n_0 > 0$，当 $n \ge n_0$ 时有 $f(n) > cg(n)$，即 $\dfrac{f(n)}{g(n)} > c$，只要 c 足够大，可得 $\lim_{n\to\infty} \dfrac{f(n)}{g(n)} = \infty$。 □

渐进函数的关系 —— 1

- 传递性：
  - $f(n) = \Theta(g(n)) \wedge g(n) = \Theta(h(n)) \Longrightarrow f(n) = \Theta(h(n))$
  - $f(n) = O(g(n)) \wedge g(n) = O(h(n)) \Longrightarrow f(n) = O(h(n))$
  - $f(n) = \Omega(g(n)) \wedge g(n) = \Omega(h(n)) \Longrightarrow f(n) = \Omega(h(n))$
  - $f(n) = o(g(n)) \wedge g(n) = o(h(n)) \Longrightarrow f(n) = o(h(n))$
  - $f(n) = \omega(g(n)) \wedge g(n) = \omega(h(n)) \Longrightarrow f(n) = \omega(h(n))$
- 自反性：
  - $f(n) = \Theta(f(n))$
  - $f(n) = O(f(n))$
  - $f(n) = \Omega(f(n))$
- 对称性：$f(n) = \Theta(g(n)) \iff g(n) = \Theta(f(n))$

渐进函数的关系 —— 2

- 转置对称性：
  - $f(n) = O(g(n)) \iff g(n) = \Omega(f(n))$
  - $f(n) = o(g(n)) \iff g(n) = \omega(f(n))$
- 与实数关系的对比：
  - $f(n) = \Theta(g(n))$ 类似于 $a = b$
  - $f(n) = O(g(n))$ 类似于 $a \le b$; $f(n) = \Omega(g(n))$ 类似于 $a \ge b$
  - $f(n) = o(g(n))$ 类似于 $a < b$; $f(n) = \omega(g(n))$ 类似于 $a > b$

渐进函数三分性不成立

与实数不同，对两个渐进函数$f(n)$和$g(n)$，可能$f(n) = O(g(n))$和$f(n) = \Omega(g(n))$都不成立，比如对于$f(n) = n$和$g(n) = n^{1+\sin n}$，$g(n)$在n^0和n^2之间摆动，不能用渐进符号来描述。

提纲

Flour 和 Ceiling

Flour 和 Ceiling

$\lfloor x \rfloor$ 表示小于等于 x 的最大整数；$\lceil x \rceil$ 表示大于等于 x 的最小整数。

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

推论 (如果 k 是整数，则 $\lfloor k+x \rfloor = k+\lfloor x \rfloor$，$\lceil k+x \rceil = k+\lceil x \rceil$)

推论 (对任意整数 n，$\lfloor n/2 \rfloor + \lceil n/2 \rceil = n$)

证 若 $n=2k$，则 $\lfloor n/2 \rfloor = k$，$\lceil n/2 \rceil = k$，$\lfloor n \rfloor + \lceil n \rceil = 2k = n$；
若 $n=2k+1$，则 $\lfloor n/2 \rfloor = k$，$\lceil n/2 \rceil = k+1$，$\lfloor n \rfloor + \lceil n \rceil = 2k+1 = n$。 □

Flour 和 Ceiling(续)

推论 (对任意整数a, b, n，$a \neq 0$，$b \neq 0$，则有：$\lceil \lceil n/a \rceil /b \rceil = \lceil n/ab \rceil$，$\lfloor \lfloor n/a \rfloor /b \rfloor = \lfloor n/ab \rfloor$)

证 1) 若$n = kab$，则$\lceil \lceil n/a \rceil /b \rceil = \lceil kb/b \rceil = k = \lceil kab/ab \rceil = \lceil n/ab \rceil$;
若$n = kab + \alpha$，$\alpha \in (0, ab)$，令$\lceil \alpha/a \rceil = m \in (0, b)$，则$\lceil \lceil n/a \rceil /b \rceil = \lceil (kb + m)/b \rceil = k + 1 = \lceil (kab + \alpha)/ab \rceil = \lceil n/ab \rceil$;

2) 与 1) 类似。

□

线性和

重新给出多项式和：

推论（对任意多项式 $p(n)=\sum_{i=0}^{d} a_i n^i$，$a_d>0$，$p(n)=\sum_{i=0}^{d} a_i n^i=\Theta(n^d)$。）

证 $p(n)=\sum_{i=0}^{d} a_i n^i \le (d+1)\max(a_i)n^d = O(n^d)$;

同时 $p(n)=\sum_{i=0}^{d} a_i n^i \ge a_d n^d = \Omega(n^d)$,

所以 $p(n)=\sum_{i=0}^{d} a_i n^i = \Theta(n^d)$。 $\square$

如果 $f(n)=O(n^k)$，则称 $f(n)$ 为多项式界限的。

线性和性质

推论 ($\sum_{k=1}^{n}(c\,a_k+b_k)=c\sum_{k=1}^{n}a_k+\sum_{k=1}^{n}b_k$)

推论 ($\sum_{k=1}^{n}\Theta(f(k))=\Theta\left(\sum_{k=1}^{n}f(k)\right)$)

证 用数学归纳法：当 $n=1$ 时，$\Theta(f(1))=\Theta(f(1))$ 显然成立；设 $n\le m$ 时成立；当 $n\le m+1$，

$$\begin{aligned}\sum_{k=1}^{m+1}\Theta(f(k)) &= \sum_{k=1}^{m}\Theta(f(k))+\Theta(f(m+1))=\Theta\left(\sum_{k=1}^{m}f(k)\right)+\Theta(f(m+1))\\ &= \Theta\left(\sum_{k=1}^{m}f(k)+(f(m+1))\right)=\Theta\left(\sum_{k=1}^{m+1}f(k)\right)\end{aligned}$$

注：$\Theta(f(n))+\Theta(g(n))=\Theta(f(n)+g(n))$ $\square$

级数常用公式

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

(2) $$\sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots + x^n = \frac{x^{n+1}-1}{x-1} \qquad (x \neq 1)$$

(3) $$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x} \qquad |x| < 1$$

(4) $$e^x = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \cdots$$

(5) $$1 + x \le e^x \le 1 + x + x^2 \qquad |x| < 1$$

(6) $$\lim_{n\to\infty}\left(1 + \frac{x}{n}\right)^n = e^x$$

(7) $$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots \qquad |x| < 1$$

(8) $$\frac{x}{1+x} \le \ln(1+x) \le x \qquad x > -1$$

$a_{i+1}/a_i < 1$ 时的级数求和

推论 (对于 $\forall i \in N$，若 $a_{i+1}/a_i \le r < 1$，则 $\sum_{i=1}^{+\infty} a_i \le \frac{a_0}{1-r}$)

证 先证明 $a_i \le a_0 r^i$ 对 $\forall i \in N$ 成立，然后再求和。 □

求 $\sum_{i=1}^{+\infty} \frac{i^2}{2^i}$ 的上界

由 $\frac{\frac{(i+1)^2}{2^{i+1}}}{\frac{i^2}{2^i}} = \frac{1}{2}(1+\frac{1}{i})^2 \le \frac{8}{9}$ 在 $i \ge 3$ 时恒成立，利用上述推论可得，

$$\sum_{i=1}^{+\infty} \frac{i^2}{2^i} = \frac{3}{2} + \sum_{i=3}^{+\infty} \frac{i^2}{2^i} \le \frac{3}{2} + \frac{9/8}{1-8/9} = \frac{93}{8}$$

证明 $\sum\limits_{k=0}^{n} 3^k = O(3^n)$

证 令 $c = 3/2$，当 $n = 0$ 时，$\sum\limits_{k=0}^{n} 3^k = 1 \le c = c3^n$。

假设 $n \le m$ 时结论成立，即对于 $c = 3/2$ 和 $n_0 = 1$，当 $n \ge n_0$ 时，$\sum\limits_{k=0}^{m} 3^k \le c3^m$ 成立。令 $n = m + 1$，则 $=1$

$\sum\limits_{k=0}^{m+1} 3^k = \sum\limits_{k=0}^{m} 3^k + 3^{m+1} \le c3^m + 3^{m+1} = c3^{m+1}(1/3 + 1/c) = c3^{m+1}$，得证。 $\square$

在使用数学归纳法时，要避免过早引用归纳假设导致证明中的常数 c 发生变化而引起错误。

数学归纳法错误用法

证明 $\sum_{k=1}^{n} k = O(n)$

证 $n = 1$ 时，$\sum_{k=1}^{1} k = 1 = O(1)$，结论成立。

假设结论对 $n-1$ 成立，则由

$\sum_{k=1}^{n} k = \sum_{k=1}^{n-1} k + n = O(n-1) + O(n) = O(n)$，可知结论对 n 也成立。 □

上述证明的错误在于，证明过程中 $O(n)$ 中隐含的常数 c 发生了变化，并不是真正的常数。欲证 $\sum_{k=1}^{n} k = O(n)$，须证 $\sum_{k=1}^{n} k \leq cn$ 对某个常数 $c > 0$ 恒成立。

缩放法和分裂法

推论 (缩放法)

$$\sum_{k=1}^{n} a_k \leq n \times \max\{a_k\}$$

推论 (分裂法)

如果对于 $\forall 1 \leq i \leq n$ 均有 $a_i \geq 0$，且 $1 \leq k \leq n$，则

$$\sum_{i=1}^{n} a_i = \sum_{i=1}^{k} a_i + \sum_{i=k+1}^{n} a_i \geq \max\{\sum_{i=1}^{k} a_i, \sum_{i=k+1}^{n} a_i\}$$

证明 $\sum_{k=1}^{n} k = \Theta(n^2)$

证 由缩放法，$\sum_{k=1}^{n} k \leq \sum_{k=1}^{n} n = n^2 = O(n^2)$。

由分裂法，则由

$$\sum_{k=1}^{n} k = \sum_{k=1}^{n/2} k + \sum_{k=n/2+1}^{n} k \geq \sum_{k=1}^{n/2} 0 + \sum_{k=n/2+1}^{n} n/2 \geq (n/2)^2 = \Omega(n^2)。$$

□

积分法

推论

(1) 如果 $f(x)$ 单调递增，则 $\int_{m-1}^{n} f(x)\,\mathrm{d}x \le \sum_{k=m}^{n} f(k) \le \int_{m}^{n+1} f(x)\,\mathrm{d}x$

(2) 如果 $f(x)$ 单调递减，则 $\int_{m}^{n+1} f(x)\,\mathrm{d}x \le \sum_{k=m}^{n} f(k) \le \int_{m-1}^{n} f(x)\,\mathrm{d}x$

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

证 令 $f(x) = x^{-1}$，由上述积分公式，$\sum_{k=1}^{n} \frac{1}{k} \ge \int_{1}^{n+1} \frac{\mathrm{d}x}{x} = \ln(n+1) = \ln n + \ln(1+\frac{1}{n}) > \ln n$

$\sum_{k=1}^{n} \frac{1}{k} = 1 + \sum_{k=2}^{n} \frac{1}{k} \le 1 + \int_{1}^{n} \frac{\mathrm{d}x}{x} = 1 + \ln n$ □

提纲

递归方程

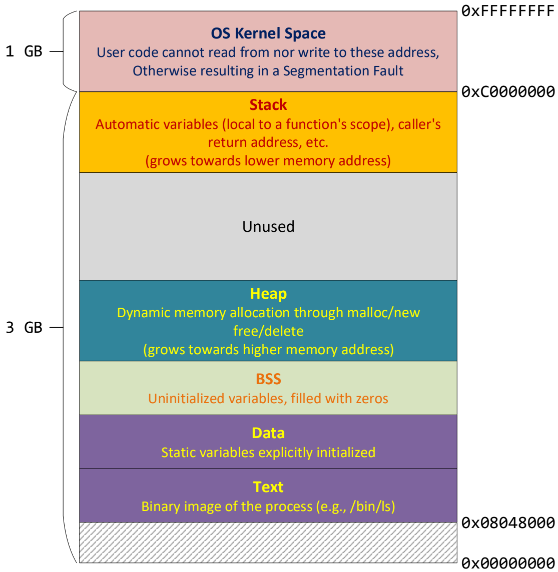
递归方程是用函数在较小自变量上的取值来刻画函数在较大自变量上的取值的函数方程。

递归方程实例

1) $T(n) = a_1 T(n-1) + a_2 T(n-2) + \cdots + a_k T(n-k)$
2) $T(n) = T(n-1) + g(n)$
3) $T(n) = 2T(n/2) + n$

包含递归调用的算法往往可以采用递归方程来描述其时间复杂度，从而简化复杂度分析过程！

函数对递归调用的支持——Stack



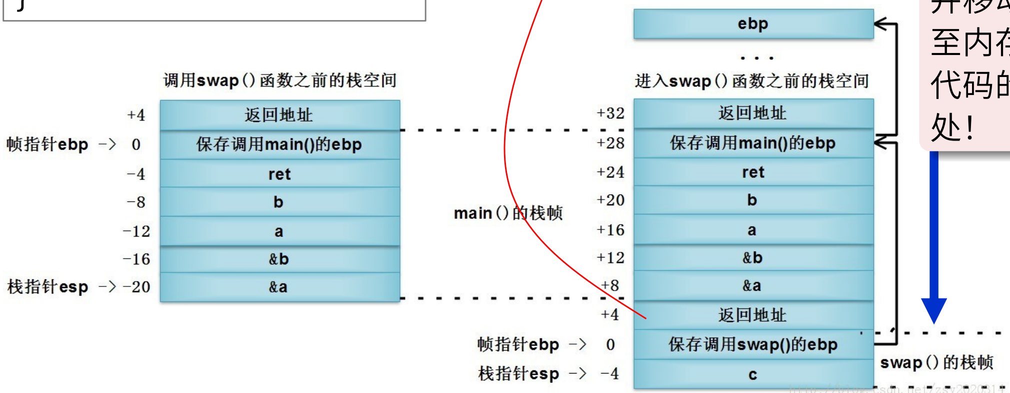
- 对某函数的每一次调用，会在 Stack 中生成一段栈帧
- 函数的不同调用 f(n) 和 f(n-1) 之间是独立的，共享相同的代码，但栈帧不同，从而对不同的参数和数据进行计算

栈帧示例

```
void swap(int *a,int *b)
{
    int c;
    c = *a;
    *a = *b;
    *b = c;
}
```

```
int main(void)
{
    int a=16, b=64, ret=0;
    swap(&a,&b);
    ret = a - b; return ret;
}
```

函数递归调用时，只需要在函数内部调用自身，则会为新的调用生成栈帧，并移动程序指针至内存中本函数代码的起始地址处!



分治算法的递归方程求解方法

分治算法的时间复杂度递归方程通常具有如下形式:

将一个输入规模为n的问题分而治之，变成a个输入规模n/b的问题，和代价f(n),一般来说，2个规模为(n/2)

$$\begin{cases} T(n) = aT(n/b) + f(n) & n > 1 \\ T(n) = 1 & n = 1 \end{cases}$$

三种求解上述递归方程的方法:

- 迭代法/递归树法：将递归方程反复进行迭代，将目标函数表达为一个求和表达式然后求和；或者将递归式画成树，通过累计各层代价来求解；
- 代入法/替换法：猜测递归方程的一个界，然后用数学归纳法证明猜测是正确的；
- **主定理法**：采用主定理直接求解。

递归方程求解时，通常忽略上取整、下取整和边界条件等操作!

迭代法求解递归方程 $T(n) = 2T(n/2) + cn$

解：

$$\begin{aligned} T(n) &= 2T(n/2) + cn \\ &= 2^2T(n/2^2) + 2\cdot(cn/2) + cn \\ &= 2^3T(n/2^3) + 2^2\cdot(cn/2^2) + 2\cdot(cn/2) + cn \\ &= \cdots \\ &= 2^kT(n/2^k) + kcn \\ &= 2^{\log n}T(1) + cn\log n \\ &= \Theta(n\log n) \end{aligned}$$

迭代法求解递归方程 $T(n) = 3T(\lfloor n/4 \rfloor) + n$

解：
$$
\begin{aligned}
T(n) &= 3T(\lfloor n/4 \rfloor) + n \\
&= 3^2 T(\lfloor n/4^2 \rfloor) + 3\lfloor n/4 \rfloor + n \\
&= 3^3 T(\lfloor n/4^3 \rfloor) + 3^2\lfloor n/4^2 \rfloor + 3\lfloor n/4 \rfloor + n = \cdots \\
&= 3^i T(\lfloor n/4^i \rfloor) + 3^{i-1}\lfloor n/4^{i-1} \rfloor + \cdots + 3^2\lfloor n/4^2 \rfloor + 3\lfloor n/4 \rfloor + n
\end{aligned}
$$

令 $n/4^i = 1 \Rightarrow i = \log_4 n$，因此

$$
\begin{aligned}
T(n) &= 3^{\log_4 n} T(\lfloor 1 \rfloor) + \sum_{j=0}^{\log_4 n - 1} 3^j \lfloor \frac{n}{4^j} \rfloor \\
&\leq 3^{\log_3 n \cdot \log_4 3} T(1) + n \sum_{i=0}^{\infty} \left(\frac{3}{4}\right)^i = n^{\log_4 3} T(1) + \Theta(n) = \Theta(n)
\end{aligned}
$$

代入法/替换法

步骤：

- 猜测解的形式；
- 用数学归纳法求出解中的常数，并证明解是正确的。

求解 $T(n) = 2T(n/2) + n$ 的上界并用代入法证明

证 首先猜测 $T(n) = O(n \log n)$；接下来证明对于恰当选择的常数 $c > 0$，有 $T(n) \leq cn \log n$。

假定上界 $T(n) \leq cn \log n$ 对于所有正数 $k < n$ 都成立，则对于 $k = n/2 < n$，有 $T(k) = T(n/2) \leq c(n/2) \log(n/2)$，将其带入递归方程，得到：

$$
\begin{aligned}
T(n) &= 2T(\frac{n}{2}) + n \leq 2c\frac{n}{2} \log \frac{n}{2} + n = cn \log \frac{n}{2} + n \\
&= cn \log n - cn \log 2 + n = cn \log n - cn + n \leq cn \log n
\end{aligned}
$$

只要 $c \geq 1$，上式必成立，从而得证。 $\square$

在代入法中添加低阶项

求解下述递归方程并用代入法证明结论

$$T(n)=\begin{cases} d & n=1 \\ T(\lfloor n/2 \rfloor)+T(\lceil n/2 \rceil)+n & n>1 \end{cases}$$

上题很容易猜测解为 $\Theta(n\log n)$，但直接用归纳法证明 $T(n)\le c_1 n\log n$ 和 $T(n)\ge c_2 n\log n$ 却会遇到困难。此时的一个技巧是在猜测结果中添加低阶项，比如此处分别用归纳法证明 $T(n)\le c_1 n\log n + d_1 n$ 和 $T(n)\ge c_2 n\log n + d_2 n + e$，并在证明结束时确定 c_1、c_2、d_1、d_2、e 等常数的值。

下面仅给出对上界的证明，下界的证明类似！

在代入法中添加低阶项 (证明过程不做要求)

证 利用前述归纳假设，首先证明上界：

$$
\begin{aligned}
T(n) &= T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + n \\
&\le c_1 \lfloor n/2 \rfloor \log \lfloor n/2 \rfloor + d_1 \lfloor n/2 \rfloor + c_1 \lceil n/2 \rceil \log \lceil n/2 \rceil + d_1 \lceil n/2 \rceil + n \\
&= c_1 \lfloor n/2 \rfloor \log \lfloor n/2 \rfloor + c_1 \lceil n/2 \rceil \log \lceil n/2 \rceil + (d_1 + 1)n \\
&\le c_1 \frac{n}{2} \log \frac{n}{2} + c_1 (\frac{n}{2} + 1) \log (\frac{n}{2} + 1) + (d_1 + 1)n = c_1 n \log n \\
&+ \left[d_1 + 1 - c_1 + \frac{1}{2} c_1 \log \left(1 + \frac{2}{n} \right) \right] n - c_1 + c_1 \log \left(1 + \frac{2}{n} \right) \\
&\le c_1 n \log n + (d_1 + 1 - c_1/2) n \quad /* \; n \ge 2 \text{时} \log(1 + 2/n) \le 1 \; */ \\
&\le c_1 n \log n + d_1 n \quad /\text{只要} c_1 \ge 2 \; */
\end{aligned}
$$

最终，取 $c_1 = 2$ 并由 $T(1) = d \le 2 \cdot 0 + d_1$ 推出 $d_1 = d$，则对 $\forall n$ 均有 $T(n) \le c_1 n \log n + d_1 n$。

求解并证明 $T(n)=4T(n/2)+n$ 的上界

证： 猜测 $T(n)=O(n^2)$，则：
$T(n)=4T(n/2)+n\le 4c(n/2)^2+n=cn^2+n$，要求证的是 $T(n)\le cn^2$，证明遇到困难。
技巧：证明一个更难的结论 $T(n)\le c_1n^2-c_2n$，重新利用归纳法，可得：

$$
\begin{aligned}
T(n) &= 4T(n/2)+n\\
&\le 4(c_1(n/2)^2-c_2(n/2))+n\\
&= c_1n^2-2c_2n+n\\
&= c_1n^2-c_2n-(c_2n-n)\le c_1n^2-c_2n
\end{aligned}
$$

代入法中添加低阶项示例

求解下述递归方程上界并用代入法证明结论

$$T(n)=\begin{cases} d & n \le 34 \\ 2T(n/2+17)+n & n>34 \end{cases}$$

证 猜测上界为 $O(n\log n)$，递归方程中包含 $n/2+17$ 的非规则项，为证明带来困难。因此，猜测上界为 $T(n)\le c(n-34)\log n-n$。$n-34$ 项用来在递归方程中消除非规整项。同时，$c(n-34)\log n-n=cn\log n-34c\log n-n$，显然小于 $cn\log n$。我们通过证明一个更严格的界限来说明为 $O(n\log n)$ 的上界成立。假设归纳假设成立，则

$$\begin{aligned} T(n)=2T(n/2+17)+n &\le 2c(n/2+17-34)\log(n/2+17)-2n+n \\ &= c(n-34)\log(n/2+17)-n \\ &\le c(n-34)\log n-n \qquad /*\ n>34, \log(n/2+17)<\log n\ */ \end{aligned}$$

$T(n) = 2T(n/2+17)+n$ 的证法 2

证 递归方程中包含 $n/2+17$ 的非规则项，其常数项很难处理。因此，通过规整化 $n/2+17$ 证明 $O(n\log n)$。

假设 $T(k) \le ck\log k$ 对 $k = n-1$ 成立，则

$$
\begin{aligned}
T(n) &= 2T(n/2+17)+n \\
&\le 2c(n/2+17)\log(n/2+17)+n \\
&= c(n+34)\log(n/2+17)+n \\
&\le c(n+34)\log(n/2+n/3)+n \qquad /*\ n > 51\ */ \\
&= cn\log n + [(c\log(5/6)+1)n] + 34c\log(5n/6) \\
&\le cn\log n + (1-0.26c)n + 34c\log n
\end{aligned}
$$

任取一足够大的 c，比如 $c = 10$，上式后半部分变为 $-1.6n + 340\log n$，该式只要 n 足够大，比如 $n > 4096$，则上式为负，从而 $T(n) < cn\log n$ 成立，得证。

代入法中的变量代换技巧

变量代换求解 $T(n) = 2T(n/2 + 17) + n$

证 令 $n = m + 34$, 代入原方程得: $T(m+34) = 2T(m/2+34) + m + 34$。令 $S(m) = T(m+34)$，则：$S(m) = 2S(m/2) + m + 34$

变换后容易求得 $S(m) = \Theta(m \log m)$。由此，$T(n) = S(n-34) = \Theta((n-34)\log(n-34)) = \Theta(n \log n)$。 □

求解 $T(n) = 2T(\lfloor \sqrt{n} \rfloor) + \log n$

证 令 $m = \log_2 n$，代入原方程得：$T(2^m) = 2T(2^{m/2}) + m$。令 $S(m) = T(2^m)$，则：$S(m) = 2S(m/2) + m$

变换后容易求得 $S(m) = \Theta(m \log m)$。由此，$T(n) = \Theta(\log n \log \log n)$。 □

代入法中的变量代换技巧 (续)

求解递归方程

$$T(n) = \begin{cases} 1 & n \le 4 \\ T(n/2) + T(n/4) & n > 4 \end{cases}$$

证 令 $m = \log_2 n$，代入原方程得：$T(2^m) = T(2^{m-1}) + T(2^{m-2})$。令 $S(m) = T(2^m)$，则：$S(m) = S(m-1) + S(m-2)$，由此：

$$S(m) = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^m - \left(\frac{1-\sqrt{5}}{2}\right)^m\right] \quad \text{(见下节)}$$

$$T(n) = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{\log n} - \left(\frac{1-\sqrt{5}}{2}\right)^{\log n}\right]$$

主定理

主定理适用的问题

递归方程为 $T(n) = aT(n/b) + f(n)$，其中 $a \geq 1$ 和 $b > 1$ 是常数，$f(n)$ 是渐进正函数，即将规模为 n 的问题分解为 a 个子问题，每个子问题的规模为 n/b，其中 a 和 b 都是正常数，每个子问题花费的时间为 $T(n/b)$，函数 $f(n)$ 则包含了分解和合并子问题的代价。

定理 (主定理)

令 $a \geq 1$ 和 $b > 1$ 是常数，$f(n)$ 是渐进正函数，$T(n)$ 是定义在非负整数上的递归方程：$T(n) = aT(n/b) + f(n)$，则 $T(n)$ 的渐进界如下：

1) 若对某个常数 $\varepsilon > 0$ 有 $f(n) = O(n^{(\log_b a)-\varepsilon})$，则 $T(n) = \Theta(n^{\log_b a})$；
2) 若 $f(n) = \Theta(n^{\log_b a})$，则 $T(n) = \Theta(n^{\log_b a} \log n)$；
3) 若对某个常数 $\varepsilon > 0$ 有 $f(n) = \Omega(n^{(\log_b a)+\varepsilon})$，且对某个常数 $c < 1$ 和所有足够大的 n 有 $af(n/b) \leq cf(n)$，则 $T(n) = \Theta(f(n))$。

主定理的进一步理解

- 主定理的三种情况靠 $f(n)$ 与 $n^{\log_b a}$ 比较的不同情况来区分：
  - $f(n)$“小于” $n^{\log_b a}$，$n^{\log_b a}$ 在复杂度中起主要作用，此时复杂度为 $\Theta(n^{\log_b a})$
  - $f(n)$ “等于” $n^{\log_b a}$，此时复杂度为 $\Theta(n^{\log_b a}\log n)$
  - $f(n)$ “大于” $n^{\log_b a}$,$f(n)$ 在复杂度中起主要作用,此时复杂度为 $\Theta(f(n))$
- 使用主定理需要注意的地方：
  - 第一种情况所说的小于必须是多项式意义上的小于，必须要相差一个因子 n^{ε}，其中 ε 是一个大于 0 的常数
  - 第三种情况的大于同样是多项式意义上的大于，也相差一个因子 n^{ε}，而且还要满足 “正则” 条件 $af(n/b)\le cf(n)$
  - 从上述要求可知，主定理并未覆盖 $f(n)$ 所有的可能性，情况一和情况二之间，以及情况二和情况三之间，都存在一些缝隙未被覆盖，如果递归方程落入这些缝隙，则不可用主定理求解

主定理应用示例 — 1

求解 $T(n) = 9T(n/3) + n$

解 考虑主定理，$a = 9$，$b = 3$，$f(n) = n$，$n^{\log_b a} = n^2$，所以 $f(n) = O(n^{(\log_3 9)-\varepsilon})$，令 $\varepsilon = 1$，可应用主定理情况一，即 $T(n) = \Theta(n^2)$。 □

求解 $T(n) = T(2n/3) + 1$

解 $a = 1$，$b = 3/2$，$f(n) = 1$，$n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$，所以 $f(n) = \Theta(\log_b a) = \Theta(1)$，适用主定理情况二，即 $T(n) = \Theta(\log n)$。 □

求解 $T(n) = 3T(n/4) + n \log n$

解 $a = 3$，$b = 4$，$f(n) = n \log n$，$n^{\log_b a} = n^{\log_4 3} = n^{0.7925}$，令 $\varepsilon = 0.2$，有 $f(n) = \Omega(n^{(\log_4 3)+\varepsilon})$，可能适用主定理第三种情况。进而，对于 $c = 3/4$，$af(n/b) = 3(n/4)\log(n/4) = (3n/4)\log(n/4) \le (3/4)n \log n$，应用主定理第三种情况得到 $T(n) = \Theta(n \log n)$。 □

主定理应用示例 — 2

求解 $T(n) = 2T(n/2) + n\log n$

解 $a = 2$, $b = 2$, $f(n) = n\log n$，$n^{\log_b a} = n$，虽然对于 $n \geq 2$ 存在 $n\log n \geq n$，但这种大于关系的存在是渐进大于而不是多项式意义上的大于，因为 $f(n)/n^{\log_b a} = (n\log n)/n = \log n$ 对任意正常数 ε 都渐进小于 n^{ε}，因此不能应用主定理求解。 $\square$

求解 $T(n) = 8T(n/2) + \Theta(n^2)$

解 $a = 8$, $b = 2$, $f(n) = n^2$，$n^{\log_b a} = n^{\log_2 8} = n^3$，令 $\varepsilon = 1$，有 $f(n) = O(n^{(\log_2 8)-\varepsilon})$，应用主定理第一种情况得到 $T(n) = \Theta(n^3)$。 $\square$

求解 $T(n) = 7T(n/2) + \Theta(n^2)$

解 $a = 7$, $b = 2$, $f(n) = n^2$, $n^{\log_b a} = n^{\log_2 7} \approx n^{2.80}$, 令 $\varepsilon = 0.8$, 有 $f(n) = O(n^{(\log_2 7)-\varepsilon})$，应用主定理第一种情况得到 $T(n) = \Theta(n^{2.80})$。 $\square$

提纲

常系数线性递归方程

定义 (常系数线性递归方程)

称递归方程

$$T(n)=a_1T(n-1)+a_2T(n-2)+\cdots+a_kT(n-k)+f(n)$$

为 k 次常系数线性递归方程。如果 $f(n)=0$，则该方程为齐次的，否则称为非齐次的。同时，称一元 k 次方程

$$x^k-a_1x^{k-1}-a_2x^{k-2}-\cdots-a_k=0$$

为递归方程的特征方程。

非常系数线性递归方程求解较为复杂，需要先转化为常系数线性递归方程，此处不做要求！

常系数齐次线性递归方程求解

定理 (常系数齐次线性递归方程求解定理)

设常系数线性递归方程的特征方程的根为 $r_1, r_2, \cdots, r_s$，每个根的重数依次为 $m_1, m_2, \cdots, m_s$，则常系数齐次线性递归方程 $T(n) = a_1T(n-1) + a_2T(n-2) + \cdots + a_kT(n-k)$ 的通解为

$$T(n) = \sum_{i=1}^{s} (c_{i1} + c_{i2} \cdot n + \cdots + c_{im_i} \cdot n^{m_i-1}) \cdot r_i^n$$

其中系数 $c_{i1}, c_{i2} \cdots$ 由方程的初始条件 $T(0), T(1), \cdots, T(k-1)$ 唯一确定。

常系数齐次线性递归方程求解示例

求解方程 $T(n) = T(n-1) + T(n-2)$，其中 $T(0) = 1, T(1) = 1$

特征方程为 $x^2 - x - 1 = 0$，其根为 $(1 \pm \sqrt{5})/2$，其重数均为 1。因此，方程的通解为 $T(n) = c_1\left((1+\sqrt{5})/2\right)^n + c_2\left((1-\sqrt{5})/2\right)^n$。将 $T(0) = 1, T(1) = 1$ 代入得 $c_1 = 1/\sqrt{5}$，$c_2 = -1/\sqrt{5}$。因此，

$$T(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

上述递归方程实际上是斐波那契数列的递归方程，其递归解法的复杂度为指数复杂度。如下是一种效率为 $\Theta(\log n)$ 的解法：

$$\begin{bmatrix} F(n-1) & F(n) \\ F(n) & F(n+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n, \quad n \geq 1$$

常系数多项式非齐次线性递归方程求解

定理 (常系数多项式非齐次线性递归方程求解定理)

如果常系数线性递归方程的 $f(n)=\sum_{i=1}^{d} e_i n^i$ 是 n 的 d 次多项式，1 作为特征方程的根的重数为 m，则递归方程的一个特解为

$$T_p(n)=p_0+p_1n+\cdots+p_{d+m}n^{d+m}$$

其中系数 $p_0,p_1,\cdots,p_{d+m}$ 由方程自身决定。该递归方程的解为

$$T(n)=T_h(n)+T_p(n)$$

其中 $T_h(n)$ 是相应齐次方程的通解。

常系数多项式非齐次线性递归方程求解示例

求解递归方程 $T(n) = 2T(n-1) - T(n-2) - 6$

特征方程为 $x^2 - 2x + 1 = 0$，其解为 1 并且是 2 重根，因此递归方程的特解为 $T_p(n) = p_0 + p_1 n + p_2 n^2$，将解代入回原方程得：

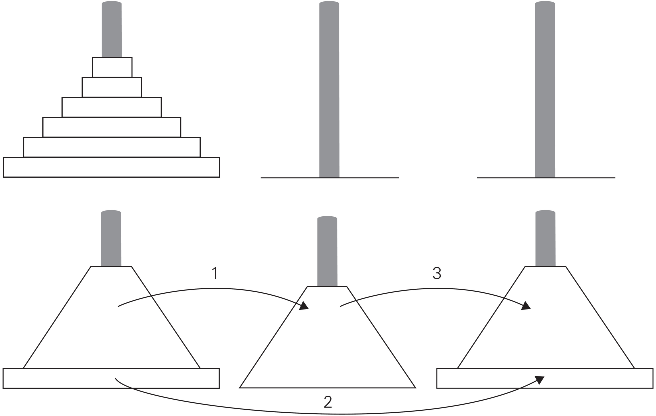
$$\begin{aligned} p_0 + p_1 n + p_2 n^2 &= 2[p_0 + p_1(n-1) + p_2(n-1)^2] \\ &\quad - [p_0 + p_1(n-2) + p_2(n-2)^2] - 6 \\ &= (p_0 - 2p_2 - 6) + p_1 n + p_2 n^2 \end{aligned}$$

解得 $p_2 = -3$，故方程的特解为 $p_0 + p_1 n - 3n^2$。

该递归方程的齐次方程通解为 $T_h(n) = c_0 1^n + c_1 n 1^n$，因此方程的解为 $T(n) = c_0 + c_1 n + p_0 + p_1 n - 3n^2$，合并同类项，$T(n) = c_0' + c_1' - 3n^2$，其系数可根据初值 $T(0)$ 和 $T(1)$ 确定。

汉诺塔问题递归解法的复杂度分析

1) 为把 n 个盘子从木桩 1 移到 3, 首先将 $n-1$ 个盘子从木桩 1 移到 2
2) 将底部最大盘子从木桩 1 移到 3
3) 将木桩 2 上的 $n-1$ 个盘子从木桩 2 移到 3, 完成移动



汉诺塔问题的递归方程及其求解

汉诺塔问题的递归方程可直接列出：$T(n) = T(n-1) + 1 + T(n-1)$，等号右边的三项分别对应前述步骤 1–3 的代价，即

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

首先采用迭代法求解：

$$\begin{aligned}
T(n) &= 2T(n-1) + 1 \\
&= 2[2T(n-2) + 1] + 1 = 2^2T(n-2) + 2 + 1 \\
&= 2^2[T(n-3) + 1] + 2 + 1 = 2^3T(n-3) + 2^2 + 2 + 1 \\
&\cdots \\
&= 2^{n-1}T(n-(n-1)) + 2^{n-2} + \cdots + 2 + 1 \\
&= 2^n - 1
\end{aligned}$$

不要被递归算法的简洁形式迷惑，可能复杂度很高！

汉诺塔问题递归方程的公式求解

解汉诺塔递归方程 $T(n) = 2T(n-1) + 1$

1) 特征方程为 $x - 2 = 0$，其解不包含 1(重数为 0)，因此递归方程的特解为 $T_p(n) = p_0$，将解代入回原方程得：$p_0 = 2p_0 + 1$，因此 $p_0 = -1$，即方程的特解为-1
2) 该递归方程的齐次方程通解为 $T_h(n) = c_0 2^n$
3) 因此方程的解为 $T(n) = c_0 2^n - 1$，由 $T(1) = 1$ 可得 $c_0 = 1$，因此 $T(n) = 2^n - 1$

常系数指数型非齐次线性递归方程求解

定理 (常系数多项式非齐次线性递归方程求解定理)

如果常系数线性递归方程的 $f(n) = ca^n$，且 a 作为特征方程的根的重数为 m，则递归方程的一个特解为

$$T_p(n) = (p_0 + p_1 n + \cdots + p_m n^m)a^n$$

其中系数 $p_0, p_1, \cdots, p_m$ 由方程自身决定。该递归方程的解为

$$T(n) = T_h(n) + T_p(n)$$

其中 $T_h(n)$ 是相应齐次方程的通解。

常系数指数型非齐次线性递归方程求解示例

求解递归方程 $T(n)=5T(n-1)-6T(n-2)+4\cdot 3^n$

特征方程为 $x^2-5x+6=0$，其解为 2 和 3，并且 3 是 1 重根，因此递归方程的特解为 $T_p(n)=p_0 3^n+p_1 n 3^n$，将解代入回原方程得：

$$
\begin{aligned}
p_0 3^n+p_1 n 3^n &= 5[p_0 3^{n-1}+p_1(n-1)3^{n-1}] \\
&= -6[p_0 3^{n-2}+p_1(n-2)3^{n-2}]+4\cdot 3^n \\
&= [(9p_0-3p_1+36)+9p_1 n]3n^{n-2}
\end{aligned}
$$

解得 $p_0 3^n+p_1 n 3^n=(9p_0+9p_1 n)3n^{n-2}\Rightarrow p_1=12$，故方程的特解为 $T_p(n)=p_0 3^n+12n3^n$。该递归方程的齐次方程通解为 $T_h(n)=c_1 2^n+c_2 3^n$，因此方程的解为 $T(n)=c_1 2^n+c_2 3^n+p_0 3^n+12n3^n$，合并同类项，$T(n)=c_1 2^n+c_2' 3^n+12n3^n$，其系数可根据方程初值 $T(0)$ 和 $T(1)$ 确定。

生成函数法求解递归方程

求解递归方程

$$\begin{cases} T(n)=1 & n=1 \\ T(n)=T(1)T(n-1)+T(2)T(n-2)+\cdots+T(n-1)T(1) & n>1 \end{cases}$$

解：令递归方程的生成函数为如下形式的级数：
$G(z)=T(1)z+T(2)z^2+\cdots+T(n)z^n+\cdots$。由此，

$$\begin{aligned} G^2(z)=&T^2(1)z^2+[T(1)T(2)+T(2)T(1)]z^3+\cdots \\ &+[T(1)T(n-1)+T(2)T(n-2)+\cdots+T(n-1)T(1)]z^n+\cdots \end{aligned}$$

由递归方程：

$$\begin{aligned} &T(2)=T(1)T(1) \\ &T(3)=T(1)T(2)+T(2)T(1) \\ &\quad\cdots \\ &T(n)=T(1)T(n-1)+T(2)T(n-2)+\cdots+T(n-1)T(1) \end{aligned}$$

生成函数法求解递归方程 (续)

对比可得 $G(z)=z+G^2(z)$，求解该方程得到 $G(z)$ 为：$G(z)=\dfrac{1\pm\sqrt{1-4z}}{2}$

由于 $G(z)$ 中各项系数均为正整数，故 $G(z)=\dfrac{1+\sqrt{1-4z}}{2}$

将 $G(z)$ 在 0 点泰勒展开，其 n 次项的系数即为 $T(n)$，所以 $T(n)$

$$=\frac{1}{2}\left[\frac{\frac{1}{2}(-\frac{1}{2})(-\frac{3}{2})\cdots(-\frac{2n-3}{2})}{n!}(-4)^n\right]=\frac{1}{2}\left[\frac{\frac{1}{2}(\frac{1}{2})(\frac{3}{2})\cdots(\frac{2n-3}{2})(n-1)!}{n!(n-1)!}(2)^{2n}\right]$$

$$=\frac{1}{2}\left[\frac{\frac{1}{2}(\frac{1}{2})\cdot 1\cdot(\frac{3}{2})\cdot 2\cdots\cdot(\frac{2n-3}{2})(n-1)}{n!(n-1)!}2^{2n}\right]=\frac{1}{2}\left[\frac{\frac{1}{2}\cdot 1\cdot 2\cdot 3\cdot 4\cdot\cdots\cdot(2n-3)(2n-2)}{n!(n-1)!}2^2\right]$$

$$=\frac{1\cdot 2\cdot 3\cdot 4\cdot\cdots\cdot(2n-3)(2n-2)}{n!(n-1)!}=\frac{1}{n}\binom{2n-2}{n-1}$$

小结

- 理解复杂度函数的阶并掌握其证明方法 (重点)
- 掌握常见标准符号和通用函数
- 掌握分治算法的递归方程 (重点)
- 了解一般递归方程的求解方法 (自学，不作为考试内容)