数据结构与算法笔记

X

2017 年 11 月 10 日

本文是作者关于数据结构与算法的读书笔记，侧重于记录和总结算法相关的数学方法，主要参考了Mark Allen Weiss的*数据结构与算法分析(C语言描述)*。本文的章节顺序，数学符号等都尽量与该书保持一致，同时也参考了网络资源或者其他书籍，均在对应章节或者习题序号下列出。由于水平所限，文中谬误在所难免，欢迎指正。

目录

1 初等数论基础

[夜深人静写算法：初等数论，http://www.cppblog.com/menjitianya/archive/2015/12/02/212395.html]

1.1 基本概念

1.1.1 整除性

若a, b为整数，a整除b是指b是a的倍数，a是b的约数，记做$a|b$。关于整除的性质有

1. 任意性：若$a|b$，则对于任意非零整数m，都有$am|bm$。
2. 传递性：若$a|b$，$b|c$，则$a|c$。
3. 可消性：若$a|bc$且a, c互素，则$a|b$。
4. 组合性：若$c|a$且$c|b$，则对于任意整数m, n，都有$c|ma+nb$。

Exercise 1.1. *假设x,y,z均为整数，若$11|\left(7x+2y-5z\right)$，求证$11|\left(3x-7y+12z\right)$。*

Solution 1.1.

Proof. 令$3x-7y+12z = m\left(7x+2y-5z\right)+11\left(ax+by+cz\right)$，其中$m, a, b, c$均为整数。

如果等式要成立，则两边x, y, z的系数均要相等，得到

$$\begin{cases} 7m+11a=3 \\ 2m+11b=-7 \\ -5m+11c=12 \end{cases} \tag{1}$$

可知其中的一个解为$m=2, a=-1, b=-1, c=2$。

故可以得到$3x-7y+12z = 2\left(7x+2y-5z\right)+11\left(-1x-1y+2z\right)$。即$(3x-7y+12z)$可以分解为11与$(7x+2y-5z)$的加权之和。

又因为$11|\left(7x+2y-5z\right)$，以及$11|11$，故根据整除性的组合性质，$11|\left(3x-7y+12z\right)$。

$\square$

2 算法分析

[主定理的证明，http://blog.csdn.net/u014627430/article/details/53510696]
[Mark Allen Weiss, 数据结构与算法分析，第十章]

2.1 算法复杂度的数学定义

Definition 2.1. 关于算法的复杂度本文使用如下定义

1. 如果对于所有足够大的n，$T(N)$的上界由$f(N)$的常数倍决定，也就是说，如果存在正常数c和n_0，使得当$N \geq n_0$时，都有$T(N) \leq cf(N)$，则记为$T(N) = \mathcal{O}(f(N))$。
2. 对于所有足够大的n，$T(N)$的下界由$g(N)$的常数倍决定，也就是说，如果存在正常数c和n_0，使得当$N \geq n_0$时，都有$T(N) \geq cg(N)$，则记为$T(N) = \Omega(g(N))$。
3. 如果对于所有足够大的n，$T(N)$的上界和下界由$h(N)$的常数倍决定，也就是说，如果存在正常数c_1, c_2和n_0，使得当$N \geq n_0$时，都有$c_1 g(N) \leq T(N) \leq c_2 g(N)$，则记为$T(N) = \Theta(g(N))$。
4. 如果$T(N) = \mathcal{O}(p(N))$ 且$T(N) \neq \Theta(p(N))$，则$T(N) = o(p(N))$。

2.2 算法复杂度的性质

Theorem 2.1. 如果$T_1(N) = \mathcal{O}(f(N))$且$T_2(N) = \mathcal{O}(g(N))$，那么

- $T_1(N) + T_2(N) = \mathcal{O}\left(\max\{f(N), g(N)\}\right)$ 。
- $T_1(N)T_2(N) = \mathcal{O}\left(f(N)g(N)\right)$。

Proof. 因为$T_1(N) = \mathcal{O}(f(N))$，则存在正常数$c_1$和$n_1$，使得当$N \geq n_1$时，都有$T_1(N) \leq c_1 f(N)$；又因为$T_2(N) = \mathcal{O}(g(N))$，同理存在正常数$c_2$和$n_2$，使得当$N \geq n_2$时，都有$T_2(N) \leq c_2 g(N)$。

令$n = \max\{n_1, n_2\}$，故当$N \geq n$时，都有

$$\begin{aligned} T_1(N) + T_2(N) &\leq c_1 f(N) + c_2 g(N) \\ &\leq \max\{c_1, c_2\}\left(f(N) + g(N)\right) \\ &\leq 2\max\{c_1, c_2\}\max\{f(N), g(N)\} \end{aligned}$$

令$c = 2\max\{c_1, c_2\}$，则说明当$N \geq n$，都有$T_1(N) + T_2(N) \leq c\max\{f(N), g(N)\}$，即$T_1(N) + T_2(N) = \mathcal{O}\left(\max\{f(N), g(N)\}\right)$ 。

对于第二条性质，令$d = c_1 c_2$，则当$N \geq n$，都有$T_1(N)T_2(N) \leq df(N)g(N)$，即$T_1(N)T_2(N) = \mathcal{O}\left(f(N)g(N)\right)$。

□

Theorem 2.2.

对于任意整数k，$\log^k N = \mathcal{O}(N)$。该条定理说明对数增长非常缓慢。

Proof. 当$k < 0$时，$\log^k N < 1 = \mathcal{O}(N)$。

下面用数学归纳法证明$k \geq 0$ 的情况。

- $k = 0$时，$\log^k N = 1 = \mathcal{O}(N)$
- 假设$k = i$时都有$\log^k N = \mathcal{O}(N)$，当$k = i+1$时

$$\begin{aligned}
\lim_{N\to\infty} \frac{\log^{i+1} N}{N} &= \lim_{N\to\infty} (i+1)\frac{\log^i N}{N} \quad \text{(根据洛必达法则)} \\
&= (i+1)\lim_{N\to\infty} \frac{\log^i N}{N} \\
&= (i+1) * 0 \quad \left(\text{根据}\log^k N = \mathcal{O}(N)\text{可知}\lim_{N\to\infty}\frac{\log^i N}{N} = 0\right) \\
&= 0
\end{aligned}$$

□

2.3 复杂度方程的解法

Lemma 2.1. 假设定义在非负整数上的函数$f(n), g(n)$满足关系$f(n) = af(\frac{n}{b}) + g(n)$，其中实数$a \geq 1$，整数$b > 1, n > 0, k > 0$且$n = b^k$，则有

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

Proof. 由$f(n), g(n)$的关系联立得到k个等式

$$\begin{aligned}
f(n) &= af(\frac{n}{b}) + g(n) \\
f(\frac{n}{b}) &= af(\frac{n}{b^2}) + g(\frac{n}{b}) \\
&... \\
f(b) &= af(1) + g(b)
\end{aligned}$$

对这k个等式依次两边分别乘以$1, a, a^1, ...a^{k-1}$，再求和，消去等式两边相等的项，得到

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

□

Theorem 2.3 (Master Theorem)**.**
$T(N)$是定义在非负整数的函数，满足

$$T(N) = aT(\frac{N}{b}) + cN^k$$

其中a, k为实数，b为整数，且$a \geq 1, b > 1$，另外满足$N = b^m, m$为整数，则

$$T(N) = \begin{cases} \mathcal{O}(N^{\log_b a}) & a > b^k \\ \mathcal{O}(N^k \log N) & a = b^k \\ \mathcal{O}(N^k) & a < b^k \end{cases}$$

Proof. 根据引理(2.1)可知

$$\begin{aligned} T(N) &= a^m T(1) + c\sum_{j=0}^{m-1} a^j \left(\frac{N}{b^j}\right)^k \\ &= a^{\log_b N} T(1) + cN^k \sum_{j=0}^{\log_b N-1} \left(\frac{a}{b^k}\right)^j \end{aligned} \tag{2}$$

当$a = b^k$, 公式(2)等价于

$$\begin{aligned} & b^{k\log_b N} T(1) + cN^k \sum_{j=0}^{\log_b N-1} \left(\frac{b^k}{b^k}\right)^j \\ = & \left(b^{\log_b N}\right)^k T(1) + cN^k \log_b N \\ = & N^k T(1) + +cN^k \log_b N \end{aligned}$$

根据定理(2.1)算法的复杂度由具有较大增长次数的部分决定，那么

$$T(N) = \mathcal{O}(N^k \log_b N) = \mathcal{O}(N^k \log N)$$

当$a \neq b^k$时,

$$\begin{aligned} T(N) &= a^{\log_b N} T(1) + c\sum_{j=0}^{\log_b N-1} \left(\frac{a}{b^k}\right)^j N^k \\ &= a^{\log_b N} T(1) + cN^k \frac{\frac{a}{b^k}\left[1 - \left(\frac{a}{b^k}\right)^{\log_b N-1}\right]}{1 - \frac{a}{b^k}} \\ &= a^{\log_b N} T(1) + \frac{cN^k\left[a - \frac{a^{\log_b N}}{b^{k(\log_b N-1)}}\right]}{b^k - a} \\ &= a^{\log_b N} T(1) + \frac{cN^k}{b^k - a}\left(a - \frac{a^{\log_b N} b^k}{b^{k\log_b N}}\right) \\ &= a^{\log_b N} T(1) + \frac{acN^k}{b^k - a} - \frac{cN^k a^{\log_b N} b^k}{(b^k - a)\, b^{k\log_b N}} \end{aligned} \tag{3}$$

根据换底公式可知$a^{\log_b N} = N^{\log_b a}$，故公式(3)第一项等价于$N^{\log_b a}T(1)$。
进一步化简公式(3)第三项可得

$$\frac{cN^k a^{\log_b N} b^k}{(b^k - a)\, b^{k\log_b N}} = \frac{cN^k N^{\log_b a} b^k}{(b^k - a)\, N^k} = \frac{cb^k N^{\log_b a}}{b^k - a}$$

所以公式(3)可化简为

$$\begin{aligned} T(N) &= N^{\log_b a} T(1) + \frac{acN^k}{b^k - a} - \frac{cb^k N^{\log_b a}}{b^k - a} \\ &= N^{\log_b a} \left(T(1) - \frac{cb^k}{b^k - a} \right) + N^k \left(\frac{ac}{b^k - a} \right) \end{aligned}$$

当$a < b^k$，$\log_b a < \log_b b^k = k$

$$T(N) = \mathcal{O}\left(N^k \left(\frac{ac}{b^k - a} \right) \right) = \mathcal{O}(N^k)$$

当$a > b^k$，$\log_b a > \log_b b^k = k$

$$T(N) = \mathcal{O}\left(N^{\log_b a} \left(T(1) - \frac{cb^k}{b^k - a} \right) \right) = \mathcal{O}(N^{\log_b a})$$

□