常用链接

我的随笔

我的评论

我的参与

最新评论

我的标签

积分与排名

积分 - 82403

排名 - 18654

随笔分类

C++(8)

C++PRIMER(4)

LINUX/UNIX(2)

LOVE/HOPE(16)

SHELL/脚本语言(2)

UNIX环境高级编程第二版(23)

UNIX环境高级编程第三版(10)

UNIX网络编程/网络/网络安全(51)

编程思想/异常处理(2)

编译编译器(4)

并发并行(2)

动态库/OS/正则/IO(2)

工作(1)

算法导论4.3主方法

4.3 主方法

主方法(master method)给出求解如下形式的递归式的“食谱”方法：

$$T(n) = aT(n/b) + f(n) \tag{4.5}$$

其中 $a \geqslant 1$ 和 $b > 1$ 是常数，$f(n)$是一个渐近正的函数。主方法要求记忆三种情况，但这样可很容易确定许多递归式的解，且不需用笔和纸。

递归式(4.5)描述了将规模为 n 的问题划分为 a 个子问题的算法的运行时间，每个子问题规模为 n/b，a 和 b 是正常数。a 个子问题被分别递归地解决，时间各为 $T(n/b)$。划分原问题和合并答案的代价由函数 $f(n)$描述。(即使用 2.3.2 节中的记号，$f(n)=D(n)+C(n)$。)如，MERGE-SORT 过程的递归式中有 $a=2$，$b=2$，$f(n)=\Theta(n)$。

从技术正确性角度看，递归式实际上没有得到很好的定义，因为 n/b 可能不是个整数。但用

货币金融学(2)
深入理解计算机系统(1)
数据结构与算法(2)
算法导论(30)
学习工作方法/规则习惯/工具(45)
证券投资基金(5)
转贴(6)

随笔档案
2016年9月(2)
2016年6月(1)
2016年2月(8)
2016年1月(6)
2015年11月(1)
2015年10月(1)
2015年9月(7)
2015年7月(3)
2015年6月(3)
2015年4月(6)
2015年3月(59)
2015年2月(63)
2015年1月(26)
2011年3月(3)
2011年2月(1)
2011年1月(9)

$T(\lfloor n/b \rfloor)$或 $T(\lceil n/b \rceil)$来代替 a 项 $T(n/b)$并不影响递归式的渐近行为(我们将在下节对此证明)。因而，我们在写分治算法时略去下取整和上取整函数会带来很大的方便。

主定理

主方法依赖于下面的定理：

定理 4.1(主定理) 设 $a \geqslant 1$ 和 $b > 1$ 为常数，设 $f(n)$为一函数，$T(n)$由递归式

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

对非负整数定义，其中 n/b 指 $\lfloor n/b \rfloor$ 或 $\lceil n/b \rceil$。那么 $T(n)$可能有如下的渐近界：

1)若对于某常数 $\varepsilon > 0$，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 $T(n) = \Theta(n^{\log_b a} \lg n)$；

3)若对某常数 $\varepsilon > 0$，有 $f(n) = \Omega(n^{\log_b a + \varepsilon})$，且对常数 $c < 1$ 与所有足够大的 n，有 $af(n/b) \leqslant cf(n)$，则 $T(n) = \Theta(f(n))$。 ■

在运用该定理之前，先来看看它包含哪些内容。在以上三种情况的每一种中，都把函数 $f(n)$与函数 $n^{\log_b a}$进行比较。我们的直觉是解由两个函数中较大的一个决定。例如在第一种情况中，函数 $n^{\log_b a}$更大，则解为 $T(n) = \Theta(n^{\log_b a})$。在第三种情况下，$f(n)$是较大的函数，则解为 $T(n) = \Theta(f(n))$。在第二种情况中，两种函数同样大，乘以对数因子，则解为 $T(n) = \Theta(n^{\log_b a} \lg n)$。

这只是我们的直觉，另外还有一些技术问题要加以理解。在第一种情况中，不仅要有 $f(n)$小于 $n^{\log_b a}$，还必须是多项式地小于，即对某个常量 $\varepsilon > 0$，$f(n)$必须渐近地小于 $n^{\log_b a}$，两者差一个因子 n^{ε}。在第三种情况中，$f(n)$不仅要大于 $n^{\log_b a}$，且要多项式地大于，还要满足“规则性”条件 $af(n/b) \leqslant cf(n)$。后面将碰到的大部分多项式有界的函数都满足这个条件。

要注意三种情况并没有覆盖所有可能的 $f(n)$。当 $f(n)$只是小于 $n^{\log_b a}$但不是多项式地小于时，在第一种情况和第二种情况之间就存在一条“沟”。类似情况下，当 $f(n)$大于 $n^{\log_b a}$，但不是多项式地大，第二种情况和第三种情况之间就会存在一条“沟”。如果 $f(n)$落在任一条“沟”中，或是第三种情况中规则性条件不成立，则主方法就不能用于解递归式。

2010年12月(2)

收藏夹

google

GMail

飞鱼秀

苏铅坤

stackoverflow

ROSI

ChinaUnix Man

snort

tcpip详解笔记

平凡的幸福

动态规划

Leetcode

一亩三分地

北大在线

百度

科学网

结构之法算法之道

刘未鹏

青铜器

孙永杰

园子

新随笔

主方法的应用

在应用此方法时，先决定要选取定理中的哪一种情况（如果有情况可满足的话），然后即可简单地写下答案。

先看第一个例子：

$$T(n) = 9T(n/3) + n$$

在这个递归式中，$a=9$，$b=3$，$f(n)=n$，则 $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$。因为 $f(n) = O(n^{\log_3 9 - \epsilon})$，其中 $\epsilon=1$，这对应于主定理中的第一种情况，答案为 $T(n)=\Theta(n^2)$。

再看一个例子：$T(n)=T(2n/3)+1$

其中 $a=1$，$b=3/2$，$f(n)=1$，$n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$。第二种情况成立，因为 $f(n)=\Theta(n^{\log_b a}) = \Theta(1)$，故递归式的解为 $T(n)=\Theta(\lg n)$。

对递归式 $T(n)=3T(n/4)+n\lg n$，有 $a=3$，$b=4$，$f(n)=n\lg n$，$n^{\log_b a} = n^{\log_4 3} = O(n^{0.793})$。因为 $f(n)=\Omega(n^{\log_4 3+\epsilon})$，其中 $\epsilon\approx 0.2$，如果能证明对 $f(n)$ 第三种情况中的规则性条件成立，则选用定理中的第三种情况。对足够大的 n，$af(n/b)=3(n/4)\lg(n/4) \leqslant (3/4)n\lg n = cf(n)$，$c=3/4$，则递归式的解为 $T(n)=\Theta(n\lg n)$。

对下面的递归式主方法不适用：

$$T(n) = 2T(n/2) + n\lg n$$

这个递归式在形式上是合适的：$a=2$，$b=2$，$f(n)=n\lg n$，$n^{\log_b a} = n$。看上去可选择第三种情况，因为 $f(n)=n\lg n$ 渐近大于 $n^{\log_b a} = n$，但并不是多项式大于。对任意正常数 ϵ，比值 $f(n)/n^{\log_b a} = (n\lg n)/n = \lg n$ 渐近小于 n^{ϵ}。因此，该递归式落在情况二与情况三之间。（练习 4.4-2 给出了解答）