算法设计与分析

罗国杰

上节课回顾

- 递归方程的求解
  - 迭代法
    - 直接迭代
    - 换元迭代
    - 差消化简后迭代
  - 递归树
  - 主定理

主定理（Master Theorem）

- Bentley, Haken, Saxe, "A general method for solving divide-and-conquer recurrences," *ACM SIGACT News*, 12 (3): 36–44, 1980.
- Popularized by the CLRS Textbook

主定理（三种情况）

设a≥1, b>1为常数，$f(n)$为函数，$T(n)$为非负整数，且

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

$$T(n) \geq f(n)$$

$$T(n) \geq f(n) + af\left(\frac{n}{b}\right) + a^2 f\left(\frac{n}{b^2}\right) + \cdots = \sum_{j=0}^{\log_b n - 1} a^j f\left(\frac{n}{b^j}\right)$$

$$\ldots\ldots\ T(n) = \Omega(f(n))$$

$$T(n) \geq aT\left(\frac{n}{b}\right) \geq a^2 T\left(\frac{n}{b^2}\right) \geq \cdots \geq a^k T\left(\frac{n}{b^k}\right) \geq a^{\log_b n} T(1) = n^{\log_b a} T(1)$$

$$\ldots\ldots\ T(n) = \Omega(n^{\log_b a})$$

比较$f(n)$和$n^{\log_b a}$两个函数的阶（三种情况：**O Θ Ω**）

主定理

主定理：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$f(n)=\Omega(n^{\log_b a+\varepsilon}), \varepsilon>0$, 且对于某个常数$c<1$和充分大的$n$有 $a f(n/b)\leq cf(n)$，那么 $T(n)=\Theta(f(n))$

串行算法的设计技术

- 分治策略
- 动态规划算法
- 回溯法与分支估界
- 贪心算法
- 概率算法

分治策略（Divide and Conquer）

- 分治策略的基本思想
  - 实例、主要思想、算法描述、注意问题
- 递归算法与递推方程
  - 两类递推方程的求解
- 降低递归算法复杂性的途径
  - 代数变换减少子问题个数
  - 预处理减少递归的操作
- 典型实例分析

分治策略的基本思想

分治策略的实例----二分检索、归并排序

主要思想----划分、求解子问题、综合解

算法描述

Divide-and-Conquer(P)

1. if $|P| \le c$ then $S(P)$.
2. divide P into P_1, P_2, …, P_k.
3. for i = 1 to k
4. y_i = Divide-and-Conquer(P_i)
5. Return Merge(y_1, y_2, …, y_k)

注意问题----连续划分　平衡原则

递归算法与递推方程

- 分治策略的算法分析工具-----递推方程
- 两类递推方程

$$f(n)=\sum_{i=1}^{k}a_i f(n-i)+g(n)$$

$$f(n)=af(\frac{n}{b})+d(n)$$

- 求解方法

  迭代法、递归树、Master定理

典型的递推方程

$$f(n)=af(\frac{n}{b})+d(n)$$

当 $d(n)$为常数 时

$$f(n)=\begin{cases} O(n^{\log_b a}) & a\neq 1 \\ O(\log n) & a=1 \end{cases}$$

当 $d(n) = cn$ 时

$$f(n)=\begin{cases} O(n) & a<b \\ O(n\log n) & a=b \\ O(n^{\log_b a}) & a>b \end{cases}$$

实例

例1 芯片测试

A 报告	*B* 报告	结论
B是好的	***A***是好的	***A*,*B*** 都好或 ***A*,*B*** 都坏
B是好的	***A***是坏的	至少一片是坏的
B是坏的	***A***是好的	至少一片是坏的
B是坏的	***A***是坏的	至少一片是坏的

条件：有n 片芯片，（好芯片至少比坏芯片多**1**片)，

问题：使用最少测试次数，从中挑出1片好芯片

要求：说明测试算法，进行复杂性分析

算法

```
k ← n
while  k > 3  do
  将芯片分成 ⌊k/2⌋ 组
  for  i =1  to  ⌊k/2⌋  do
     if  2片好，则任取1片留下
     else  2片同时丢掉
  k ← 剩下的芯片数
if  k = 3
  then  任取2片芯片测试
     if  至少1坏，取没测的芯片
     else  任取1片被测芯片
if  k =2  or  1  then  任取1片
```

分析

- 说明

  上述算法只是一个概要说明，对于n为奇数的情况需要进一步处理, 处理时间为$O(n)$.

- 复杂性分析

  设$W(n)$表示n片芯片测试的次数，则

  $$W(n) = W(n/2) + O(n)$$

  $$W(1) = 0$$

  由Master定理，$W(n) = O(n)$

实例

例2　求一个数的幂

问题：计算 a^n，n为自然数

传统算法：$\Theta(n)$

分治法

$$a^n = \begin{cases} a^{n/2} \times a^{n/2} & n \text{ 为偶数} \\ a^{(n-1)/2} \times a^{(n-1)/2} \times a & n \text{ 为奇数} \end{cases}$$

$$T(n) = T(n/2) + \Theta(1) \Rightarrow T(n) = \Theta(\log n) .$$

计算 Fibonacci 数

Fibonacci 数的定义

$$F_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F_{n-1} + F_{n-2} & \text{if } n > 1 \end{cases}$$

0 1 1 2 3 5 8 13 21 …

通常算法：从 F_0, F_1, …, 根据定义陆续相加

时间为$\Theta(n)$

利用数幂乘法的分治算法

定理1 设 $\{F_n\}$为 Fibonacci 数构成的数列，那么

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n$$

证明：对 n 进行归纳

算法：令矩阵 $M = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$，用分治法计算 M^n

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

提高算法效率的途径1

方法一：代数变换 减少子问题个数

例3 位乘问题

设X ,Y 是两个n 位二进制数， $n = 2^k$, 求 XY.

传统算法 $W(n)=O(n^2)$

分治法 令$X = A2^{n/2} +B$, $Y= C2^{n/2} +D$.

$$XY= AC\,2^n + (AD + BC)\,2^{n/2} + BD$$

$$W(n) = 4W(n/2) + cn,$$

$$W(1) = 1$$

解得 $W(n) = O(n^{\log 4}) = O(n^2)$

代数变换

$$AD + BC = (A - B)(D - C) + AC + BD$$

递推方程

$$W(n) = 3\,W(n/2) + cn$$

$$W(1) = 1$$

解

$$W(n) = O(n^{\log 3}) = O(n^{1.59})$$

矩阵乘法

例4 A, B 为两个n 阶矩阵，$n = 2^k$, 计算$C = AB$.

传统算法 $W(n) = O(n^3)$

分治法 将矩阵分块，得

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

其中

$$C_{11} = A_{11}B_{11} + A_{12}B_{21} \qquad C_{12} = A_{11}B_{12} + A_{12}B_{22}$$

$$C_{21} = A_{21}B_{11} + A_{22}B_{21} \qquad C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

递推方程 $W(n) = 8\ W(n/2) + cn^2$

$W(1) = 1$

解 $W(n) = O(n^3)$.

变换方法

$M_1 = A_{11}(B_{12} - B_{22})$
$M_2 = (A_{11} + A_{12})B_{22}$
$M_3 = (A_{21} + A_{22})B_{11}$
$M_4 = A_{22}(B_{21} - B_{11})$
$M_5 = (A_{11} + A_{22})(B_{11} + B_{22})$
$M_6 = (A_{12} - A_{22})(B_{21} + B_{22})$
$M_7 = (A_{11} - A_{21})(B_{11} + B_{12})$

$C_{11} = M_5 + M_4 - M_2 + M_6$
$C_{12} = M_1 + M_2$
$C_{21} = M_3 + M_4$
$C_{22} = M_5 + M_1 - M_3 - M_7$

Strassen矩阵乘法

递推方程是

$$W(n) = 7W(\tfrac{n}{2}) + 18(\tfrac{n}{2})^2$$

$$W(1) = 1$$

由Master定理得

$$W(n) = O(n^{\log_2 7}) = O(n^{2.8075})$$

本课小结

- 分治策略
- 芯片测试
- 求一个数的幂
- 计算 Fibonacci 数
- 提高算法效率的途径1
  - 代数变换 减少子问题个数
  - 举例
    - 位乘问题
    - 矩阵乘法