

《计算复杂性理论》 第4讲 分治法

山东师范大学信息科学与工程学院 段会川 2014年9月

目录

- □ 分治法概述
- □ 冯 诺伊曼
- □ 高斯乘法
- □ 乘法分解算法
- □ Karatsuba乘法算法
- □ 递推式的一般解法—主定理及应用
- □ 归并排序算法

第4讲 分治法

分治法—概述

- □ 在计算机科学中,分治法(Divide and Conquer)是建立在多项分支递归的一种很重要的算法范式。字面上的解释是"分而治之",就是把一个复杂的问题分成两个或更多的相同或相似的子问题,直到最后子问题可以简单的直接求解,原问题的解即子问题的解的合并。
- □ 这个技巧是很多高效算法的基础,如排序算法(快速 排序、归并排序)、傅立叶变换(快速傅立叶变换)。
- □ 分治算法通常以数学归纳法来验证。而它的计算成本则多数以解递推关系式来求取。
 - 如果可能,则应用主定理

第4讲 分治法

分治法—三个步骤

- 1. 分解:
 - 将原问题分解为若干个规模较小、相对独立、与原问题形式相同的子问题。
- 2. 解决:
 - 若子问题规模较小且易于解决时则直接解出。
 - 否则递归地解决各子问题。
- 3. 合并:
 - 将各子问题的解合并为原问题的解。

第4讲 分治法

4

分治法—历史

- □ 折半搜索算法(二叉搜索, binary search)
 - 将原来问题连续地拆分成大约一半大小的单一子问题的分 治算法的构想早已在公元前200年的巴比伦尼亚时代就已 经出现。
 - 算法在计算机上的清楚描述出现在1946年约翰·莫齐利 (John Mauchly) 的一篇文章里。
- □ 辗转相除法—欧几里德算法
 - 它是一个通过将问题转化为单一的更小问题从而快速求解的方法,因而也可看成是一种分治算法。
 - 它也是在2000多年前的公元前发现的。

第4讲 分治法

分治法—历史

- □ 专门用于计算机之上而且正确地分析的分治算法最早期的例子,则是约翰 冯 诺伊曼于1945年发明的归并排序算法(merge sort),也称为合并排序算法。
- \square A. A. Karatsuba基于高斯乘法方法于1960年发明的 在 $O(n^{\log_2 3})$ 步骤内将两个n位数相乘的算法是另一个 分治算法的经典例子。
 - 它反证了安德列 柯尔莫哥洛夫(安德列 尼古拉耶维奇 柯尔莫哥洛夫, Andrey Nikolaevich Kolmogorov, Андрей Николаевич Колмогоров, 1903.4.25—1987.10.20)于1956年认为两个n位数相乘需要Ω(n²)步骤的猜想。

第4讲 分治法

分治法 6

目录

- □ 分治法概述
- □ 冯 诺伊曼
- □ 高斯乘法
- □ 乘法分解算法
- □ Karatsuba乘法算法
- □ 递推式的一般解法—主定理及应用
- □ 归并排序算法

第4讲 分治法

约翰 冯 诺伊曼

- □ 出生于匈牙利的美国籍 犹太人数学家,现代计 算机创始人之一
 - 在计算机科学、经济学、 物理学中的量子力学及 几乎所有数学领域都作 过重大贡献



John von Neumann 1903.12.28—1957.2.8

第4讲 分治法

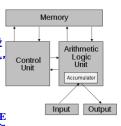
约翰 冯 诺伊曼—计算机之父

- □ 1945年6月,冯 诺伊曼与戈德斯坦、勃克斯等人,联 名发表了一篇长达101页纸的报告,即计算机史上著 名的"101页报告",是现代计算机科学发展里程碑 式的文献。
 - 该报告明确在计算机中用二进制替代十进制运算,并将计算机分成五大组件,这一卓越的思想为电子计算机的逻辑结构设计奠定了基础,已成为计算机设计的基本原则。
 - 由于他在计算机逻辑结构设计上的伟大贡献,他被誉为 "计算机之父"。

第4讲 分治法

冯·诺伊曼体系结构—Von Neumann Architecture

- □ 也称普林斯顿体系结构,是一种将程序指令和数据一起存储的计算机概念结构
 - 现代计算机基本上基于该架构设计,因而也称为存储程序计算机,它是通用图灵机的具体实现
 - IEEE(Institute of Electrical and Electronics Engineers, 国际电气与电子工程师学会,读作I-Triple-E)以他的名字命名了IEEE 冯 诺伊曼奖,奖励计算机科学和技术上具有杰出成就的科学家



第4讲 分治法

1

约翰 冯 诺伊曼—博弈论与经济学贡献

- □ 在经济学领域,1944年冯 诺伊曼与摩根施特恩合著的巨作《博弈论与经济行为》(Theory of Games and Economic Behavior)出版,标志着现代系统博弈理论的的初步形成
 - 他因此被称为"博弈论之父"。
 - 博弈论被认为是20世纪经济学最伟大的成果之一
 - INFORMS(Institute for Operations Research and the Management Sciences, 运筹学与管理科学学会)设立了 冯 诺伊曼理论奖,以奖励在运筹学与管理科学领域做出基础性和持续性贡献的科学家

第4讲 分治法

目录

- □ 分治法概述
- □ 冯 诺伊曼
- □ 高斯乘法
- □ 乘法分解算法
- □ Karatsuba乘法算法
- □ 递推式的一般解法—主定理及应用
- □ 归并排序算法

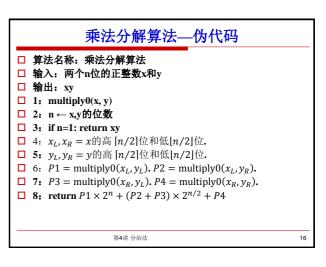
第4讲 分治法

分治法 1

高斯乘法 □ 两个复数相乘 ■ (a+bi)(c+di) = ac - bd + (ad + bc)i ■ 包括4次乘法和两次加法运算 □ 高斯乘法 ■ ad + bc = (a + b)(c + d) - ac - bd ■ 这使两个复数相乘的运算变换为3次乘法和5次加法运算 ■ 虽然初看没有多大改进,但由于乘法运算是0(n²)的复杂度,而加法运算是0(n)的复杂度,如果迭代进行,复杂度的改进还是很可观的

目录	
□ 分治法概述 □ 冯 诺伊曼 □ 高斯乘法 □ 乘法分解算法 □ Karatsuba乘法算法 □ 递推式的一般解法—主定 □ 归并排序算法	理及应用
第4讲 分治法	14

乘法的分解算法 n位的二进制数可以分解为两个 $\frac{n}{2}$ 位的组成部分 $x = \underbrace{x_{k}}_{X_{R}} \underbrace{x_{R}}_{Y_{R}} = 2^{n/2}x_{k} + x_{k}$ $y = \underbrace{y_{k}}_{Y_{R}} \underbrace{y_{k}}_{Y_{R}} = 2^{n/2}y_{k} + y_{k}.$ □ 如: $x = 10110110_{2} = 1011_{2} \times 2^{4} + 0110_{2}$ 即: $x_{L} = 1011, x_{R} = 0110.$ □ 经过上述分解,两个数的积可以表达为 ■ $xy = x_{L}y_{L}2^{n} + (x_{L}y_{R} + y_{L}x_{R})2^{n/2} + x_{R}y_{R}$ ■ 由4次乘法和3次加法组成,而乘法可以递归进行下去 ■ 注意: 乘以 2^{n} 和 $2^{n/2}$ 可以用左移操作在线性时间里实现



乘法分解算法—复杂度□ 乘法分解算法的一次调用中要执行4次乘法和3次加法,以及2次移位操作 ■ 加法和移位操作的复杂度为0(n) ■ 乘法需要递归进行 ■ 当n = 1 时,复杂度为0(1),递归结束 ■ 算法的总复杂度可以用如下的递推式表示 ■ $T(n) = \begin{cases} 0(1) & \text{当} n = 1 \\ 4T(n/2) + 0(n) & \text{当} n > 1 \end{cases}$ ■ 该递推式的解后面将会看到是 $T(n) = 0(n^2)$



Karatsuba乘法算法

- □ A. A. Karatsuba借助于高斯乘法,于1960年提出了第1个快速乘法算法
 - 基本思路是将

$$xy = x_L y_L 2^n + (x_L y_R + y_L x_R) 2^{n/2} + x_R y_R$$

中的 $x_L y_R + y_L x_R$ 表达为

 $(x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R.$

■ 使乘法运算由4次减为3次,代价是增加了3次加法

第4讲 分治法

Karatsuba乘法算法—伪代码

- □ 算法名称: Karatsuba乘法算法
- □ 输入:两个n位的正整数x和y
- □ 输出: xy
- \square 1: multiplyK(x, y)
- □ 2: n ← x,y的位数
- ☐ 3: if n=1: return xy
- □ 4: $x_L, x_R = x$ 的高 [n/2]位和低[n/2]位.
- □ 5: $y_L, y_R = y$ 的高 [n/2]位和低 [n/2]位.
- \square 6: $P1 = \text{multiply}K(x_L, y_L)$. $P2 = \text{multiply}K(x_R, y_R)$.
- $\square \quad 7: \quad P3 = \text{multiply} 0(x_L + x_R, y_L + y_R).$
- **8:** return $P1 \times 2^n + (P3 P1 P2) \times 2^{n/2} + P2$

第4讲 分治法

注 20

Karatsuba乘法算法—复杂度

- □ Karatsuba乘法算法的一次调用中要执行3次乘法和6 次加法,以及2次移位操作
 - 加法和移位操作的复杂度为O(n)
 - 乘法需要递归进行
 - $\mathbf{n} = 1$ 时,复杂度为O(1),递归结束
 - 算法的总复杂度可以用如下的递推式表示

■
$$T(n) = \begin{cases} O(1) & \text{\pm n} = 1\\ 3T(n/2) + O(n) & \text{\pm n} > 1 \end{cases}$$

■ 该递推式的解后面将会看到是

$$T(n) = O(n^{\log_2 3}) \approx O(n^{1.59})$$

第4讲 分治法

21

23

Karatsuba乘法算法—复杂度的递归树求解

- □ Karatsuba乘法算法的递归调用过程可以用一棵递归 树表示
 - 其实,所有的分治算法都可以用递归树表示
 - 递归树的高度为[logn]-1
 - 在深度为k = 0(即根)的层上,有1个规模为n的(子)问题
 - 在深度为k = 1的层上,有 $3 = 3^1$ 个规模为 $\frac{n}{2} = \frac{n}{2^1}$ 的子问题
 - 在深度为k = 2的层上,有 $9 = 3^2$ 个规模为 $\frac{n}{4} = \frac{n}{2^2}$ 的子问题
 - 在深度为k的层上,有3^k个规模为ⁿ。的子问题
 - 树的最大深度为[logn] 1

第4讲 分治法

Karatsuba乘法算法—复杂度的递归树求解

- □ Karatsuba乘法算法复杂度的递归树求解
 - 对于深度为k的层上的1个规模为 $\frac{n}{2^k}$ 的子问题,抛去乘法运算(即递归)后的运算是加法和移位,它们合起来的复杂度为 $0\left(\frac{n}{2^k}\right)$
 - 因而,深度为k的层上的全部的 3^k 个规模为 $\frac{n}{2^k}$ 的子问题的 复杂度为 $3^k \times O\left(\frac{n}{2^k}\right) = \left(\frac{3}{2}\right)^k \times O(n)$
 - 整个问题的复杂度为

$$T(n) = \sum_{k=0}^{\lceil \log n \rceil - 1} \left(\frac{3}{2}\right)^k \times O(n)$$

14 公公注

24

Karatsuba乘法算法—复杂度的递归树求解

- □ Karatsuba乘法算法复杂度的递归树求解
 - 等比数列前n项和的求和公式 $\sum_{k=0}^{n} ar^k = \frac{a(1-r^{n+1})}{1-r}$
 - **国而,** $T(n) = \sum_{k=0}^{\lceil \log n \rceil 1} \left(\frac{3}{2}\right)^k \times O(n)$
 - Up, $T(n) = 2\left(\left(\frac{3}{2}\right)^{\lceil \log n \rceil} 1\right) \times O(n) = O\left(n\left(\frac{3}{2}\right)^{\log n}\right)$ = $O\left(n\frac{3^{\log n}}{2^{\log n}}\right) = O(3^{\log n})$.

 - **因此,** $T(n) = O(n^{\log 3}) \approx O(n^{1.59})$

第4讲 分治法

目录

- □ 分治法概述
- □ 冯 诺伊曼
- □ 高斯乘法
- □ 乘法分解算法
- □ Karatsuba乘法算法
- □ 递推式的一般解法—主定理及应用
- □ 归并排序算法

笛4世 公公社

9法 2

递推式的通解—主定理

- □ 主定理(master theorem)
 - 如果对于常数 $a > 0, b > 1, d \ge 0, 有$ $T(n) = aT([n/b]) + O(n^d)$

■ 则有: $T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$

- \square 当分治法每次将问题分解为a个规模为n/b的子问题,而将各子问题的解合并的时间复杂度为 $O(n^d)$ 时,该分治法的运算时间的递推式便是
 - $T(n) = aT([n/b]) + O(n^d)$
 - 因而可以套用主定理求解

第4讲 分治法

27

乘法分解算法复杂度的主定理解

□ 乘法分解算法复杂度的递推式

 $T(n) = \begin{cases} 0(1) & \exists n = 1 \\ 4T(n/2) + 0(n) & \exists n > 1 \end{cases}$

- **显然**, a = 4, b = 2, d = 1
- 此时有, $d = 1 < \log_b a = 2$
- **国** 因而, $T(n) = O(n^{\log_b a}) = O(n^2)$

第4讲 分治法

28

Karatsuba乘法算法复杂度的主定理解

□ Karatsuba乘法算法复杂度的递推式

$$T(n) = \begin{cases} 0(1) & \stackrel{\text{def}}{=} n = 1\\ 3T(n/2) + O(n) & \stackrel{\text{def}}{=} n > 1 \end{cases}$$

- **显然**, a = 3, b = 2, d = 1
- 此时有, $d = 1 < \log_b a = \log_2 3$
- **国而**, $T(n) = O(n^{\log_b a}) = O(n^{\log 3}) \approx O(n^{1.59})$

第4讲 分治法

‡ 分治法 29

二分搜索算法(binary search, 折半查找)

- □ 算法名称:二分搜索算法(BinarySearch)
- □ 输入:元素由低到高排序的数组a和待搜索的数据x
- □ 输出: x在a中的位置, -1表示未找到
- ☐ 1: BinarySearch(a, low, high, x)
- \square 2: if low>high: return -1
- \square 3: mid = [(low+high)/2]
- □ 4: if x=a[mid] then return mid
 □ 5: if x<a[mid] then BinarySearch(a, low, mid-1, x)
- ☐ 6: else BinarySearch(a, mid+1, high, x)

第4讲 分治法

治法 30

二分搜索算法—复杂度分析 x^n 的计算算法 □ 算法名称: x^n 的计算算法(power) □ 最好情况复杂度 ■ 在第1次调用时,中间的元素就是要找的元素 □ 输入: x,n ■ 因而,复杂度为0(1) □ 输出: xⁿ □ 最坏情况复杂度 \square 1: power(x, n) ■ 当low=high时才找到,或没有找到 \square 2: if n=1 then return x T(n) = T(n/2) + O(1) \square 3: p = power(x, n/2)■ 运用主定理 \Box $a = 1, b = 2, d = 0 = \log_b a = 0$ □ 4: p = p*p□ 因而, $T(n) = O(n^d \log n) = O(\log n)$ \Box 5: if n is odd then p = p*x 31

xⁿ计算算法—复杂度分析 □ 显然, xⁿ计算算法与二分搜索算法相似 ■ T(n) = T(n/2) + O(2) ■ 运用主定理 □ a = 1, b = 2, d = 0 = log_b a = 0 □ 因而, T(n) = O(n^dlogn) = O(logn)



32

归并排序(merge sort, 合并排序)—描述 □ 归并排序由冯 诺伊曼于1945年提出,是一个典型的分治算法 ■ 分解:将排序数组分解为两个等大的子数组 ■ 解决:无 ■ 合并:将两个已经有序的数组合并为一个有序的数组□ 大小分别为m和n的有序数组可以在线性时间0(m+n)内合并

归并排序—算法伪代码	
□ 算法名称: 归并排序(MergeSort) □ 输入: n个元素的数组a □ 输出: 排序的a □ 1: MergeSort(a, low, high, b) □ 2: if high-low<=1 then return □ 3: mid = (low+high)/2 □ 4: MergeSort(a, low, mid, b) □ 5: MergeSort(a, mid+1, high, b) □ 6: Merge(a, low, mid, high, b) □ 7: copy(b, low, high, a) P59	
第4讲 分治法	36

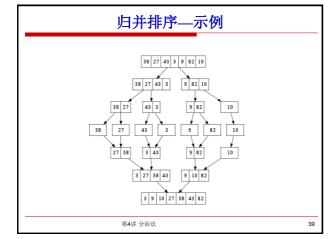
归并排序—归并算法描述

- 申请空间,使其大小为两个已经排序序列之和,该 空间用来存放合并后的序列
- 设定两个指针,最初位置分别为两个已经排序序列 的起始位置
- 3. 比较两个指针所指向的元素,选择相对小的元素放入到合并空间,并移动指针到下一位置
- 4. 重复步骤3直到某一指针到达序列尾
- 将另一序列剩下的所有元素直接复制到合并序列末 尾

井 分治法

归并排序—归并算法伪代码

- □ 算法名称: 合并两个有序数组(Merge)
- □ 输入:数组a,low到mid及mid+1到high间已排序
- □ 输出:数组a,从low到high是排序的
- □ 1: i0 = low, i1 = high
- \square 2: for j = low to high
- \square 3: if i0 < mid and (i1 >= high or A[i0] <= A[i1]))
- $\square \ \mathbf{4:} \qquad \mathbf{B[j]} = \mathbf{A[i0]}$
- □ 5: i0 = i0 + 1
- □ 6: else
- \square 7: B[j] = A[i1]
- \square 8: i1 = i1 + 1
- 9: end if 第4讲 分治法





归并排序—复杂度分析 □ 显然归并排序计算复杂度的递推式为 ■ T(n) = 2T(n/2) + O(n)■ 运用主定理 $a = 2, b = 2, d = 1 = \log_b a = 1.$ 因而, $T(n) = O(n^d \log n) = O(n \log n)$ ■ 归并排序的合并阶段需要一个规模为n的数组,因而空间复杂度为O(n)■ 归并排序是基于比较的排序算法中的一个最优算法,因为可以证明基于比较的排序算法最少需要 $\Omega(n \log n)$ 的复杂度