doi:10.16104/j.issn.1673-1891.2019.03.008

基于MATLAB矩阵运算的大整数乘法设计与实现

滕 旭
(云南大学旅游文化学院,云南 丽江 674100)

摘要:大整数运算在信息安全、数学验证、基因工程等领域有着广泛的应用,设计有效的方案提高运算效率成为学者关注的热点。大整数乘法是大整数运算中的核心运算,对如何提高大整数乘法运算效率进行了分析总结,并利用MATLAB矩阵运算结合格子乘法等算法进行了设计与实现。实验表明通过MATLAB矩阵运算进行大整数乘法运算能有效的提高运算效率。

关键词:大整数运算;格子乘法;矩阵运算
中图分类号:TP301.6 **文献标志码**:A **文章编号**:1673-1891(2019)03-0035-04

Design and Implementation of Large Integer Multiplication Based on MATLAB Matrix Operation

TENG Xu
(School of Tourism Culture, Yunnan University, Lijiang, Yunnan 674100, China)

Abstract: Large integer operations have been widely used in the fields of information security, mathematical verification, genetic engineering, etc. It has become a hot topic for scholars to design effective schemes to improve the efficiency of operations. Large integer multiplication is the core operation in large integer operations. In this paper, I analyzed and summarized the approaches to improve the efficiency of large integer multiplication, and employed the algorithm of MATLAB matrix operation in combination with the lattice multiplication to achieve the design and implementation of large integer multiplication. Experiments show that the large integer multiplication can effectively improve the efficiency of large integer operations through MATLAB matrix operation.

Keywords: large integer operation; lattice multiplication; matrix operation

1 大整数乘法的应用背景与研究现状

大整数是指超过计算机整数表达和处理范围的整数。大整数的运算在信息安全,数学验证,科学及工程计算等领域有着广泛的应用,而大整数乘法运算又是其它运算的基础,因此得到了学者的广泛关注,很多运算方案或改进方案被提了出来。

大整数运算的首要问题是计算机表示。设大整数为m,则$m=\sum_{i=0}^{n-1}x_iN^i$,记作:$m=(N,x_0,x_1,\cdots,x_{n-1})$,其中$N$称为数$m$的基,$x_i$称为数$m$的数位。计算机表示大整数的核心问题就是提高运算效率为目标,如何选择基和用什么样的数据结构存储基和数位[1-3]。显然,与不同的运算算法相适应,大整数的计算机表示会略有差别,但基本思想一致。

提高大整数乘法运算效率的方法大体上分为两大类:一类是基于硬件设计,从运算的最低层进行研究,文献[4]基于椭圆曲线密码算法对大整数乘法的特殊性,设计了专用的指令处理器提高运算效率,文献[5-7]则从硬件实现方面改进算法,分别从不同角度设计了专用的乘法器;另一类则基于软件,主要根据大整数的计算机表示方法从多个角度提出了新的算法或对经典算法进行改进,文献[8]利用OpenMP和MPI等并行计算编译原理研究了大整数乘法,并对运算性能进行了实验评估,文献[9-10]主要讨论了基于多项式存储的大整数乘法算法及改进,文献[11-17]则对大整数乘法算法从不同角度进行优化改进,主要分为三种类型:一是对基线乘法进行分析改进;二是对Comba,Toom和Karastsuba三种算法进行改进;三是对快速傅里叶变换乘法进行分析改进。文献[18-23]则基于不同语言环境结合各种乘法算法编程实现了大整数乘

收稿日期:2019-03-29
基金项目:云南省教育厅科学研究基金:基于探究式学习的数学教学研究(2019J0245)。
作者简介:滕旭(1977—),男,河南邓州人,讲师,硕士,研究方向:数学教学方法。

法的运算,部分文献还对不同算法得到的实验数据进行分析,从而验证对算法的理论分析。随着计算机多核处理器的普及,大整数乘法并行计算的研究又出现了一个新方向,文献[24-25]即是基于多核并行计算平台设计并实现了高效的算法。

综上所述,对大整数乘法的研究多集中在软件方面,通过将程序设计语言的存储特点与算法的基本原理相结合进行改进而提高运算效率是此类研究的共同特点。采用分治法,提高运算法并行度是提高运算效率的比较多见的策略。MATLAB矩阵运算的效率非常高,本文基于MATLAB矩阵运算,根据格子乘法原理设计并实现了一种大整数乘法算法,为大整数乘法的研究提供了一个新思路。

2 格子乘法的数学原理

现行小学课本中介绍了一种格子乘法,是将基线乘法中每两个数位相乘的结果放在表格中,依据一定的规则得到两个整数乘积的一种方法。相传,这种方法最早记载在1150年印度数学家婆什迦罗的《丽拉娃蒂》一书中,我国明朝数学家夏源泽在《指明算法》一书中也记载了这种方法,把它称为"铺地锦"[26]。

以753乘35为例,计算如图1所示:

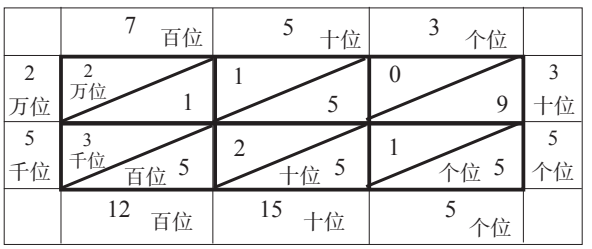

图1 格子乘法示意图

格子乘法的本质是将笔算乘法的数位转换成了表格形式:以对角线为标志分别是乘积个,十,百,千等;上面一行是因数一,以竖线为标志自右向左分别是因数一的个,十,百,千等;右面一列是因数二,以横线为标志自下向上分别是因数二的个,十,百,千等。例如粗线格内第一行第一列的21表示700×30 = 21 000 = 20 000 + 1 000,其中2在万位上表示20 000,1在千位上表示1 000。两个因数的每两个数字相乘则必为两位数,以对角线为分界放置在格子中乘积中相应的数位,将乘积中的各个数位相加分别放置在左边一列和下边一行,若相加结果是两位数则进位就得到乘积为26 355。

由于格子乘法将乘法运算中的中间结果放置在了格子中,与数学中的矩阵正好一致,本文结合矩阵的运算,设计了基于矩阵运算的乘法算法。

3 基于矩阵运算的格子乘法算法步骤

文章根据格子乘法与矩阵运算的特点将格子乘法利用矩阵运算来实现,具体算法步骤如下:

Step1(输入因数一与因数二):将两个因数输入程序以向量形式存储,分别记作:

$a=(a_1,a_2,\cdots,a_m)$, $b=(b_1,b_2,\cdots,b_n)$。

Step2(生成格子矩阵):将两个向量进行矩阵乘法运算得到格子乘法中表格中的数字并记作格子矩阵g如下:

$$g=b'\cdot a=\begin{bmatrix} a_1b_1 & a_2b_1 & \cdots & a_mb_1 \\ a_1b_2 & a_2b_2 & \cdots & a_mb_2 \\ \vdots & \vdots & & \vdots \\ a_1b_n & a_2b_n & \cdots & a_mb_n \end{bmatrix}_{n\times m}$$

取格子矩阵中的数字的个位及十位数字分别记作矩阵g_1,g_2。

Step3(生成乘积数位):分别计算矩阵g_1,g_2的次对角线上的数字之和分别记作:m_1, m_2并将m_1, m_2错位相加得到乘积数位向量m。

Step4(处理进位):将每个超出进制的乘积数位计算出进位向量mj,将进位向量mj与乘积数位向量m相加得到新的乘积数位向量m至无进位为止。

Step5(输出乘积):输出乘积数位向量m得到乘积。

如果计算机内存充足,还可以采用将运算结果预处理后直接进行读取,减少乘法计算及取余操作的次数。

直接读取生成格子矩阵的算法步骤如下:

Step1(生成预处理格子矩阵):将两个因数取作$a=(0,1,\cdots,9)$,$b=(0,1,\cdots,9)$利用前面求格子矩阵的算法生成预处理格子矩阵分别记作矩阵矩阵$g1ready$, $g2ready$。

Step2(读取预处理格子矩阵生成格子矩阵):对给定的两个因数

$a=(a_1,a_2,\cdots,a_m)$, $b=(b_1,b_2,\cdots,b_n)$,

读取预处理格子矩阵$g1ready$, $g2ready$得到格子矩阵g_1,g_2如下:

$$g_1=\begin{bmatrix} g1ready(a_1+1,b_1+1) & g1ready(a_2+1,b_1+1) & \cdots & g1ready(a_m+1,b_1+1) \\ g1ready(a_1+1,b_2+1) & g1ready(a_2+1,b_2+1) & \cdots & g1ready(a_m+1,b_2+1) \\ \vdots & \vdots & & \vdots \\ g1ready(a_1+1,b_n+1) & g1ready(a_2+1,b_n+1) & \cdots & g1ready(a_m+1,b_n+1) \end{bmatrix}_{n\times m}$$

$$g_2=\begin{bmatrix} g2ready(a_1+1,b_1+1) & g2ready(a_2+1,b_1+1) & \cdots & g2ready(a_m+1,b_1+1) \\ g2ready(a_1+1,b_2+1) & g2ready(a_2+1,b_2+1) & \cdots & g2ready(a_m+1,b_2+1) \\ \vdots & \vdots & & \vdots \\ g2ready(a_1+1,b_n+1) & g2ready(a_2+1,b_n+1) & \cdots & g2ready(a_m+1,b_n+1) \end{bmatrix}_{n\times m}$$

算法说明:本算法通过矩阵运算大大简化了大整数的运算流程,在两个大整数相乘的过程中,不仅可以逐个数位进行,还可以在计算机操作允许范围内自由的调整每次运行的数位来提高运算效率。

下文通过实验验证了上述改进思想的有效性。

4 基于矩阵运算的改进大整数乘法快速算法步骤

文章参考文献[12]中分治思想,在原有算法的基础上作了相应的改进,基于矩阵运算来实现,具体算法步骤如下:

Step1(输入因数一与因数二):将两个因数输入程序以向量形式存储,比较两个因数的位数,使a的位数大于b的位数,分别记作:

$a=(a_1,a_2,\cdots,a_m)$, $b=(b_1,b_2,\cdots,b_n)$。

Step2(生成乘积临时矩阵):选择向量a分别与2~9相乘再增加$0a$,$1a$得到乘积临时矩阵$temp$如下:

$$(2\ \ 3\ \ \cdots\ \ 9)'\cdot a=\begin{bmatrix}2a_1 & 2a_2 & \cdots & 2a_m\\ 3a_1 & 3a_2 & \cdots & 3a_m\\ \vdots & \vdots & & \vdots\\ 9a_1 & 9a_2 & \cdots & 9a_m\end{bmatrix}_{9\times m}\Rightarrow temp=\begin{bmatrix}0 & 0 & \cdots & 0\\ 1a_1 & 1a_2 & \cdots & 1a_m\\ \vdots & \vdots & & \vdots\\ 9a_1 & 9a_2 & \cdots & 9a_m\end{bmatrix}_{10\times m}$$

Step3(生成乘积数位):因数b中各个数位上数字是0~9中之一,进行笔算时只须每次从乘积临时矩阵$temp$读取相应行向量$temp(b_i+1),1\leqslant i\leqslant n$即可,分析笔算乘法数位求和规律,设计算法对从临时矩阵中调用数据进行求和分别计算乘积中各个数位,得到乘积数位向量$m=(m_1,m_2,\cdots,m_k\cdots,m_{m+n})$,其中$m_k=\sum_{i=1}^{n}a_ia_j(i+j=k,1\leqslant j\leqslant n)$,具体运算如图2所示。

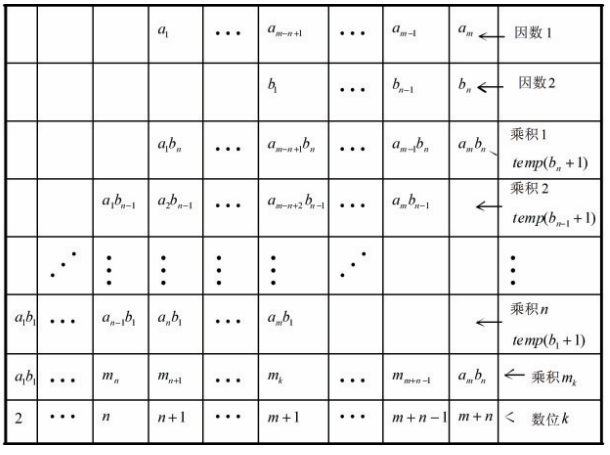

图2　生成乘积数位示意图

Step4(处理进位):同上面基于矩阵运算的格子乘法算法步骤。

Step5(输出乘积):同上面基于矩阵运算的格子乘法算法步骤。

算法说明:本算法通过乘积临时矩阵保存重复计算的数据,大大减少了乘法运算次数,提高了运算效率。与格子乘法相比,乘法算法次数减少的更多,但得到的乘积数位向量中的各个数位上的数字偏大,增加了进位处理的时间。

5 算法实现与分析

为了分析基于MATLAB矩阵的大整数乘法运算性能,文章利用MATALAB-R2010a实现了基于格子乘法的矩阵运算基本算法及其改进算法,和基于矩阵运算的改进大整数乘法快速算法,并对实验结果进行比较分析。

实验计算机配置如下:win7操作系统;CPU为AMD A8-3500M APU with Radeon(tm) HD Graphics 1.5 GHz;内存为6 GB。

基于格子乘法的矩阵运算基本算法记为算法1,2位数字作为整体计算记为算法2,3位数字算法作为整体计算记为算法3,4位数字作为整体计算记为算法4,基于矩阵运算的改进大整数乘法快速算法记为算法5,并且对算法1~3还分别通过相乘直接生成格子矩阵和预处理后通过直接读取矩阵生成格子矩阵两种情形。每种算法分别随机生成20组大整数进行测试,计算程序运行时间进行比较,实验结果如表1所示。

表1　算法运行时间对比　　s

位数	算法1	算法2	算法3	算法4	算法5
20	0.086 01/0.002 40	0.001 13/0.000 98	0.000 72/0.000 610	0.002 01	0.000 60
50	0.091 01/0.007 91	0.003 03/0.002 90	0.001 80/0.001 81	0.007 81	0.000 98
100	0.123 51/0.037 95	0.007 71/0.007 50	0.003 54/0.003 51	0.045 42	0.002 01
200	0.390 11/0.279 15	0.039 15/0.037 12	0.013 21/0.012 91	0.291 31	0.006 18
300	3.209 35/0.117 511	0.107 303/0.040 61	0.040 61/0.039 81	3.288 87	0.017 32
400	8.461 60/8.353 22	0.304 182/0.284 82	0.080 91/0.075 12	8.419 01	0.026 93
500	17.134 6/16.951 2	0.995 321/0.980 11	0.156 12/0.143 09	16.561 5	0.049 81
1 000	303.241/303.102	16.641 1/16.710 1	4.661 21/4.651 12	301.853	0.097 13

注:算法1,2,3中t_1/t_2分别指相乘生成格子矩阵和直接读取生成格子矩阵的运行时间。

比较格子乘法与直接笔算的运行时间,直接笔算随着因数位数增加,运行时间增加缓慢,这是因为格子乘法的空间复杂度是$O(n^2)$,直接笔算的空间复杂度是$O(n)$,直接笔算在保证空间复杂度的基础上由于采用了预处理乘积技术,使乘法的次数大大减少,时间复杂度降低到了$O(n)$,所以可以处理位数很大的整数乘法。

对格子乘法的不同改进而言,当整数位数超过500时运算性能均迅速下降,特别是整数位数1000位时,算法1的两种情形及算法4的运行时间分别达到了303.241秒,303.102秒,301.853秒,完全失去了运算价值,表1数据表明格子乘法只适合低于500位的整数相乘。

格子乘法中直接相乘生成格子矩阵与从读取预处理格子矩阵生成格子矩阵相比较,三位数字作为整体相乘时效果不明显,每个数位逐个相乘和两位数字作为整体相乘时效果显著,这是因为三位数字作为整体相乘时,预处理格子矩阵为1 000×1 000,读取数据的效率会大降低。

格子乘法中将若干个数位作为整体进行运算时,表1数据表明,三位数字作为整体(算法3)相乘,运算效果最好,随着位数的增加会导致取余操作的效率迅速下降,所以用格子乘法进行大整数乘法运算时应取三位数字作为整体进行运算。

6 结语

本文参阅大整数乘法研究的相关文献,对大整数乘法算法提高效率的方法进行了系统的总结,并对其中两种算法进行改进调整,设计并实现了基于矩阵运算的算法,通过不同算法运行时间的统计分析,验证了矩阵运算进行大整数乘法的特性。

参考文献

[1] 李文化,董克家.大整数精确运算的数据结构与基选择[J].计算机工程与应用,2006(32):24-26.

[2] 英昌盛,周喜龙.大整数乘法的数据结构及算法选择探究[J].长春工业大学学报(自然科学版),2008,29(2):204-207.

[3] 刘觉夫,周娟.大整数运算的基选择[J].华东交通大学学报,2008,24(2):100-102.

[4] 夏辉,于佳,秦尧,等.嵌入式领域ECC专用指令处理器的研究[J].计算机学报,2017,40(5):1092-1108.

[5] 严忻恺,吴东,邬贵明,等.高性能多精度乘法器设计[J].计算机工程与科学,2013,35(11):146-152.

[6] 许亮,王震.基于CUDA的快速大整数乘法[J].计算机工程与应用,2013,49(16):221-224.

[7] 原巍,许琪,沈绪榜.大整数乘法器设计[J].微电子学与计算机,2003(S).

[8] TEMBHURNE JV,SATHE SR.performance evaluation of long integer multiplication using OpenMP and MPI on shared memory architecture[J].In:Proc.of the 2014 7th Int'l Conf.on Contemporary Computing(IC3).IEEE,2014:283-288.

[9] 贾晓静,汤伟,范园利.基于多项式的大整数相乘算法[J].计算机工程与设计.2009,30(11):2622-2625.

[10] 尹绪昆.中国剩余定理在多项式乘法计算中的应用[J].河北省科学院学报.2012,29(1):5-9.

[11] 罗水龙,余国林,周利华.一个基于分治法的快速多精度乘法[J].吉林化工学院学报,2003,20(2):76-78.

[12] 周健,李顺东,薛丹.改进的大整数相乘快速算法[J].计算机工程,2012,38(16):121-123.

[13] 张力,张引兵,刘海.一种新的大整数乘法算法[J].计算机安全,2011(1):11-13.

[14] 薛方芳,范明芳,张蓓.对大整数乘法求解问题改进算法的思考[J].福建电脑.2009(5),67+77.

[15] 龚雪慧,王成杰.基于C++三维数组实现大整数相乘的算法[J].电脑与信息技术,2016,24(4):16-18.

[16] 实对称双线性函数与多精度整数的快速乘法[J].计算机科学,2007,34(6):92-97.

[17] 史庆霞,张桂芸,吴美云.基于万进制数组的大整数乘法的算法设计[J].哈尔滨师范大学自然科学学报,2012,28(1):55-57.

[18] 李文化.基于.NET的大整数类设计与实现[J].海南大学学报自然科学版,2010,28(2):153-158.

[19] 杜青.基于类的大整数乘法运算的实现[J].微型机与应用,2017,36(2),8-9+13.

[20] 杨灿,桑波.大整数乘法运算的实现及优化[J].计算机工程与科学,2013,35(3):183-190.

[21] 罗洋.大整数乘法的计算机处理[J].辽宁师专学报,2005,7(1):38-40.

[22] 杨剑.字符化大整数运算系统的构造与实现[J].扬州大学学报(自然科学版),2000(11):52-55.

[23] 沈建涛.用字符串进行大数运算的算法设计与实现[J].南通职业大学报,2011,25(4):93-97.

[24] 蒋丽娟,刘芳芳,赵玉文,等.大整数Comba和Karatsuba乘法的多核并行化研究[J].计算机系统应用,2016,25(11):232-236.

[25] 赵玉文,刘芳芳,蒋丽娟,等.大整数乘法Schonhage-Strassen算法的多核并行化研究[J].软件学报,2018,29(12):3604-3613.

[26] 潘红丽,潘有发."铺地锦"史话[J].珠珠算与珠心算,2010(1):53-56.

(责任编辑:曲继鹏)