文章编号:1005-0523(2004)01-0117-03

寻找正整数的素数因子的新算法

王 森[1], 王 波[2]

(1.华东交通大学 基础科学学院,江西 南昌 330013;2.齐齐哈尔市第十五中学,161005)

摘要:运用新方法开发了正整数的素数分解的算法,并对其进行了算法分析.

关 键 词:PAR方法;素数分解;算法

中图分类号:TP301.6 **文献标识码**:A

PAR方法是薛锦云教授经过十多年研究的开发快速且正确的算法和程序的方法[2].PAR方法基于分化、递推、扩充的量词变换规则、循环不变式的新技术和软件转换工具,充分利用数据抽象、功能抽象、软件重用、多态、类属、重载等成熟的程序设计技术,可以用统一的方法开发复杂算法.

0 问题介绍

问题:对于给定的一个正整数n,求出它的所有素数因子(或者把n写成素数积的形式,其中包含重复因子).

1 算法开发

第一步:用Radl语言精确地描述求解问题的功能规约.本问题是对于给出的一个正整数 n,求出它的所有的大于1的素数因子,设 n 的素数因子的个数为 m,$0\leqslant s\leqslant m$,所求素数数组:$a[s]=\forall(i,j:2\leqslant j<i\leqslant n\wedge i\bmod j\neq 0\wedge n\bmod i=0:i)$

其中 $a[1],a[2],\cdots a[m]$ 是n的素数因子从大到小排列的,当然其中有重复因子的情况.再设 $n[i]=n/\Pi(j:0\leqslant j\leqslant i:a(j))$,则

后置断言:

$AR=\forall(s:2\leqslant s\leqslant n\wedge a[s]=\forall(i,j:2\leqslant j<i\leqslant n\wedge i\bmod j\neq 0\wedge n\bmod i=0:a[s].$

第二步:把需要求解的问题划成若干和原问题结构相同但规模更小的子问题,分解可一直进行下去直至每个子问题可直接求解.

把问题分划为:$n[i]=F(n(i-1))$,$a[i]=G(a(i-1))$

第三步:构造问题求解序列的递推关系 $S_i=F(S_j)$ 并给出在递推关系中出现的函数和变量的初始化条件,再把初始化条件和所有的递推关系合并成一个算法.

1) 递推关系

$a[s]=\forall(i,j:2\leqslant j<i\leqslant n\wedge i\bmod j\neq 0\wedge n\bmod i=0:i)=\forall(i,j:(2\leqslant j<i\leqslant a[s-1]\vee a[s-1]\leqslant j<i\leqslant n)\wedge i\bmod j\neq 0\bmod i=0:i)=\forall(i,j:(2\leqslant j<i\leqslant a[s-1]\vee i\bmod j\neq 0 n\bmod i=0:i)\vee\forall(i,j:a[s-1]\leqslant j<i\leqslant n\wedge i\bmod j\neq 0\wedge n\bmod i=0:i)=a[s-1]\vee\forall(i,j:a[s-1]\leqslant j<i\leqslant n\wedge i\bmod j\neq 0\wedge n\bmod i=0:i)=$

$$\begin{cases} a[s-1] & 2\leqslant j<i\leqslant a[s-1]=2\leqslant j<i=a[s-1] \\ i & a[s-1]\leqslant j<i\leqslant n\wedge i\bmod j\neq 0\wedge n\bmod i=0 \end{cases}$$

$=G(a[s-1])$.

当 $a[s]=a[s-1]$ 时,a[s]与a[s-1]是n的重复素数因子算法.

收稿日期:2003-05-20

作者简介:王森(1969-),男,齐齐哈尔人,华东交通大学基础科学学院讲师.

$n(i)=n/\Pi(j:0\leqslant j<i:a[j])=n/(\Pi(j:0\leqslant j<i:a[j])\,a[i])=n(i-1)/a[i]$,

令 $k=n$,

$n(i)=\forall(1\leqslant i\leqslant m:n=n/\Pi(j:1\leqslant j\leqslant i:a(i)))$, $n=\Pi(i:1\leqslant i\leqslant m:a(i))$,

事实上, $n(i)=\Pi(j:i<j\leqslant m:a(j))$, m 为 n 的素数个数.

2) 性质推导

(1) $\forall(i:2\leqslant i\leqslant k:n(i)\bmod i=0:i)=\forall(i,j:2\leqslant j<i\leqslant k:n(i)\bmod i=0\wedge(i\bmod j\neq 0\vee i\bmod j=0:i)=\forall(i,j:2\leqslant j<i\leqslant k:n(i)\bmod i=0\wedge i\bmod j=0:j)$

而 $\forall(i,j:2\leqslant j<i\leqslant k:n(i)\bmod i=0\wedge i\bmod j=0:i)=\forall(i,j:2\leqslant j<i\leqslant k:\Pi(p:i<j\leqslant m:a(p))\bmod i=0\wedge i\bmod j=0:i)=\forall(i,j:2\leqslant j<i\leqslant k:false:i)=false$,

于是

$\forall(i,j\leqslant k\leqslant k:n(i)\bmod i=\forall(i,j:2\leqslant j<i\leqslant k:n(i)\bmod i=0\wedge(i\bmod j\neq 0\vee i\bmod j=0:i))=a(i)$,

即保证了算法求出的一定是素数.

定理 1 设 $i_1, i_2, \cdots, i_s$ 分别是已经找到的小于 i 的从小到大素数因子, k 为任意一个大于 i_s 小于 n 的正整数, 如果 $\exists j(1\leqslant j\leqslant s)$, 使得 $k \bmod i_j=0$, 则 $n \bmod k\neq 0$.

由已知 k 是 i_j 倍数, 而此时 $n=m/(i_1 * i_2 * \cdots * i_s)$, 即由循环知 n 中已经没有大于 1 小于 i 的素数因子了, 命题显然成立.

由此知, 此时的 i 需要判断: 如果 i 是 $i_1, i_2, \cdots, i_s$ 的某一个的倍数, 就不是因子, 就没有必要进入循环了.

(2) $\forall(i,j,p:2\leqslant p<i\leqslant n(i):2\leqslant p<j\leqslant n(i):n(i)=i^{*}j\wedge a(i)\bmod p\neq 0:i,j)$

由于 $a(i)\bmod p\neq 0$, m 可以写成 $a(i)=rp+s$, 于是

式(2) $=\forall(i,j,p:2\leqslant p<j\leqslant n(i):2\leqslant p<j\leqslant n(i):(n(i)=i * j=rp+s)\wedge a(p)\bmod p\neq 0:i,j)=\forall(i,j,p:2\leqslant p<j\leqslant n(i):2\leqslant p<j\leqslant n(i):(n(i)=i * j=rp+s)\wedge a(p)\bmod p\neq 0\wedge(j=rp/j+s/j):i,j)=\forall(i,j,p:2\leqslant p<j\leqslant n(i):2\leqslant p<j\leqslant n(i):(n(i)=i * j=rp+s)\wedge a(p)\bmod p\neq 0\wedge(j=rp/j+s)<(r+s)j:i,j)$

同理可得

式(2) $=\forall(i,j,p:2\leqslant p\leqslant i\leqslant n(i),2\leqslant p<j\leqslant n(i):(n(i)=i * j=rp+s)\wedge a(p)\bmod p\neq 0\wedge(i=rp/i+s)<(r+s)j:i,j)$

即

循环变量 i 的上限问题, 实际上 i 并没有必要循环到 m, 因为

定理 2 对于一个正整数 m 和一个小于 m 的素数 p, 设 $m \bmod p\neq 0$, 且 m 可以由两个大于 p 的 i, j 整数分解, 即 $m=i * j$, 则

$\max\{i,j\}<[m/p]+m \bmod p$. ([m/p]为取整运算)

证明: 用反证法

不妨设 $j=\max\{i,j\}$,

设 $r=[m/p]$, $s=m \bmod p$, 由于 $m \bmod p\neq 0$, $m=rp+s$,

假设 $j\geqslant[m/p]+m \bmod p=r+s$,

由已知 $m=i * j\geqslant(r+s)i=ri+si$,

又 $i>p$, 必有 $i>2$,

于是 $m>rp+s=m$, 矛盾.

当然, 如果整数 i, j 是素数因子, 结论肯定成立.

于是

推论 当 i 不是 m 的素数因子时, m 的最大素数因子小于 $[m/p]+m \bmod p$.

定理 3 对于本程序, 如果大于循环变量 i 的素数因子有两个 k, j(可以是相同的), 则必有 $\max\{k,j\}<[m/i]+m \bmod i$.

就是说, 循环变量的次数 i 上限为 $[m/i]+m \bmod i$. 定理 2 的逆否命题为:

定理 4 对于一个正整数 m 和一个小于 m 的素数 p, 设 $m \bmod p\neq 0$, 则大于 $[m/p]+m \bmod p$ 的 m 的整数因子个数小于 2, 即只有一个.

对于本算法可知: 这时的 n 为一个素数.

综上所述, 把这些结论结合起来, 得到该问题的算法程序:

```
Alpgrithm: Primenum
|i,j,s,m:integer; k: real; a: array[0:m-1,integer]|
{AQ∧AR
Begin:I:=2++1;a[i]:=1;j:=0,k:=n;s,m:=0;
Termination:n=1;
Recur:a[i]= G(a[s-1]);
          n(i)=n(i-1)/a[i];
```

第四步: 基于开发循环不变式的新策略, 开发循环不变式.

$I=(i:2\leqslant k\wedge n \bmod i=0:n=n/ia[m]=I)\vee(i:2\leqslant i<k\wedge n \bmod i\neq 0:k=trunc(n/i)\wedge(i=i+$

1)∨∃(s:0≤s<m∧i mod a[s]=0:(i=i+1)∧(s=s+1))∨∃(i:2i≤n∧i=k:a[m]=n)

第五步:基于所得算法和循环不变式,将 *Radl* 算法转换成等价的某一可执行的语言程序.经过 *Apla_ Java* 转换器把该 *Apla* 程序转换为 *Java* 程序,经过测试程序完全正确.

2　算法分析

2.1　循环分析

主循环变量 i 分两层循环:

一是判断 i 是否为已找到因子的倍数,如果是就 $i=i+1$,不参与主循环.它用 s 次的较小数模运算来代替较大的数 $n(i)$的模运算,它对于输入数较大时是很有意义的.

二是由两部分组成:

(1)当 i 能整除 n 时,可得素数因子,并且可以对重复因子进行运算;

(2)当 i 不能整除 n 时,则 $i=i+1$ 要么是已经求出的因子的倍数,要么是非因子素数或是某些非因子素数的积.这个地方给该算法优化留有了空间.一个最有效的方法是 i 取自素数集合,但这是很困难的.

2.2　计算量分析

1) 本算法用 s 次的较小数模 i mod $a[j](1\leqslant j\leqslant s)$运算来代替较大的数 $n(i)$的模运算 $n(i)$ mod i,它对于输入数 n 较大时是很有有意义的.

同时注意到这里的 $a[j]$是有重复因子的,这也优化的空间.

2) 循环次数

主循环变量 i 的上限 k 是严格递减的:[n/i]+n mod i.

设整数 n 的不同的素数因子从小到大为,则

$n=b_1^{r1}\cdot b_2^{r2}\cdots b_l^{r1}$,且 $r1+r2+\cdots rl=m$.

事实上,当 n 循环到 $i=a[m-2]$时,有两种情况:

(1) $a[m-1]=a[m]$,二者是重复因子,根据算法知:当 $i=a[m]$时,程序就结束了;

(2) $a[m-1]<a[m]$时,$n(i)=a[m]$,由条件 $i<k=[a[m]/i]+a[m]$ mod i,有

当 i 循环到 $a[m-1]$与 $a[m]$之间的某个数 q 时程序就会结束:

$q<k=[a[m]/a[m-1]]+a[m]$ mod $q\leqslant[a[m]/a[m-1]+a[m]$ mod $a[m-1]$;

而,有可能[a[m]/a[m−1]+$a[m]$ mod a[m−1]]<$a[m-1]$;

综合上述有,

定理 5　设此算法主循环变量的循环次数为 q,则

$q<\max\{a[m-1],[a[m]/a[m-1]+a[m]$ mod $a[m-1]]\}$.

对于一般的结论 $q\leqslant\sqrt{n}$结果要好,是一个比较准确的结果.

3　讨　论

目前,计算整数素数分解方法很多,利用穷举法对大数分解是计算量是很大的;如果用穷举法去寻找素数的计算量成指数增长;多段并行穷举法是目前较为普遍的一种,它主要是阶段性的算法,用到了计算机的并行运算来实现.

现在较流行概率算法,如前面提到的 Witness 算法;蒙特卡罗方法:用概率 ε 判别,经过与若干次运算,如果条件小于 ε,则来判定结果偏真,即是素数的可能性很大;遗传算法:对于一个数转化二进制后,如果其字段含有遗传因子,则是素数的可能性偏大;等等.大数素数分解至今仍是一个较难的题目.

本算法是运用 PAR 方法推导出来,保证了算法的正确性,并且有较高的效率,它是对穷举法的改进,但仍有不理想的地方,前面已有陈述.

这个算法稍加改进就可以用来求解从小到大的素数问题.

参考文献:

[1] Xue Jinyun. Two New Strategies for Developing Loop Lnvariants and Its Applications. JOURNAL COMPUTER SCIENCE AND TECHNOLOGIES. No.3,1993,4.

[2] Xue Jinyun. A Unified Approach For Developing Efficient Algorithmic Program. Journal of Computer Science and Technologies, 1997,12(4).

[3] D. Gries, Xue Jinyun, 1988.10. Generating A Random Permutation, Bit28,569−572.

[4] Gries. D. The Science of Programming. Springer Verlag. New York. 1981.

[5] Remmers. J. A Technipue for Developing Loop Invariants Information Processing Letters 18(1984) 137−139.

[6] 华罗庚. 数论导引. 北京:科学出版社,1979.

Developing New Algorithm of Finding Prime Factorization in a Positive Integer by New Method

WANG Sen[1], WANG Bo[2]

(1. East China Jiaotong University Nanchang 330013, China; 2. No. 15 Middle School of Qi-Qi Har City 161005, China)

Abstract: In this paper a the algorithm of finding frime factorization in a positive Integer is developed by a new method and the analysis of this algorithm is given.

Key word: PAR method; prime factrization; algorithm

(上接第 101 页)

Application and Benchmark Test Sotfware of Image Digital Watermarking

WU Zhi-qiang

(School of Mechanical Engineering, East China Jiaotong University, Nanchang, Jiangxi, 330013 China)

Abstract: In recent years, with many arithmetics of image digital watermarking having been proposed, the digital watermarking will be brought advance to, when benchmark testing softwares are developed to aim at the different applications of digital watermarking. To address this issues, the applications and important parameters of digital watermarking system are discussed at frist, and then the typical attacks for it are analysed integratly, At last the existing benchmark stirmarks is probed into, and it 's localizations are pointed out.

Key words: image digital watermarking; Robustness; benchmark testing software