素数确定性算法分析

李 彪,左黎明,谢 环
(华东交通大学 基础科学学院,江西 南昌 330013)

摘 要:在计算机技术以及密码学高速发展的阶段,提升生成大素数的效率,构建素数库已成为行业趋向。为了探索素数确定性算法的效率和稳定性,提出用算法执行时间和时间曲线斜率的分析方法。通过改进素数确定性算法,计算10个$n(n \geq 10^6)$以内所有的素数,各类算法所需的执行时间,得出算法执行时间曲线来判断算法的效率和稳定性。在结果中,算法执行时间越短算法的效率越高,时间曲线切点斜率越小算法的稳定性越高,得出筛法的执行效率和稳定性远高于试除法、$6k \pm 1$法优于奇偶过滤法。

关键词:素数;算法;确定性;效率;稳定性
中图分类号:TP301 **文献标识码**:A **文章编号**:1673-629X(2011)08-0026-04

Prime Number Determinacy Algorithm Analysis

LI Biao, ZUO Li-ming, XIE Huan
(School of Basic Science, East China Jiaotong University, Nanchang 330013, China)

Abstract: With rapid development of computer technology and cryptology, improving the efficiency of generating of large prime number and constructing the prime number library have become the trend of computer industry. To explore the efficiency and stability of the algorithm, propose the algorithm execution time method and slope analysis of time curve method. With the improved prime number determinacy algorithm, calculate the time it needs to obtain all the prime numbers within $n(n \geq 10^6)$ and do it ten times, in order to get the algorithm execution time and time curve to prove the efficiency and stability of the algorithm. The results prove that efficiency is higher while the algorithm execution time shorter, at the same time stability is higher while time curve slope smaller. The execution efficiency and stability of the algorithm is much better than trial division, and $6k \pm 1$ method is superior to parity filtering.

Key words: prime number; algorithm; determinacy; efficiency; stability

0 引 言

对$n(n \in N^+)$的素性探测,是一个数学与密码学交叉的热点研究领域。素数是指除了能被1和它本身整除而不能被其他任何数整除的数[1]。目前,素数确定性算法主要分为递归试除法、素数筛法、Miller检验和AKS算法。在给定$n(n \in N^+)$的情况下,通过递归试除法计算可得区间$[2,n]$中的所有素数[2],但算法的效率较低。文献[3]提出任意素数(除2和3)均可表示为$6k \pm 1$($k \in N^+$)的理论,利用此结果可降低算法的遍历次数,提高运行效率。把不超过$n(n \in N^+)$的一切正整数按大小关系排成一串,将不超过$\sqrt{n}$素数的倍数划去,所剩下的数就是n以内所有的素数[4~7]。上述素数筛法虽然运行的效率较高,但对于任意的$k(k < n, k \in N^+, k\%6 = 0)$会重复筛选,影响算法的执行效率。Miller检验[8,9]和AKS算法[10~12]主要用于对数字的素性探测,不便于寻求n以内所有素数。为了提高算法运行效率,通过改进算法、计算算法执行时间和描绘时间曲线来探测算法效率和稳定性的优越程度。

收稿日期:2011-02-09;**修回日期**:2011-05-16
基金项目:国家自然科学基金项目(11061014);江西省教育厅青年科学基金项目(GJJ10129);江西省教育厅科研项目(GJJ10708)
作者简介:李 彪(1988-),男,江西南昌人,硕士研究生,CCF会员,研究方向为信息安全;左黎明,硕士,讲师,研究生导师,CCF会员,研究方向为信息安全、非线性系统。

1 素数的确定性算法

1.1 递归试除法

递归试除法是判别一个整数是否为素数的常用方法。假定任意给定一个自然数X,用Y从2开始来试除它,如果能整除,则这个整数不是素数,否则,用下一个Y值来试除,直到Y大于$\sqrt{X}$为止。利用算术基本定理:任一大于1的整数能表成质数的乘积,即任一大于1的整数

$$a = p_1 p_2 \cdots p_n, p_1 \leq p_2 \leq \cdots \leq p_n \quad (1)$$

其中$p_1, p_2, \cdots, p_n$是质数[1]。

推论 1　假设小于整数 n 最大的素数为 p_n,则 n 必然存在一个质因子 $q < \sqrt{p_n} + 1$。

(1)任意自然数 n 能分解成 $k_1 \times k_2 \times \cdots \times k_j (k_j (j \in N^+)$ 是小于 n 的素数),另设 Max Prime 为区间(2,n) 中已存在的最大素数,则必存在一个因子 $k_j (j \in N^+)$ 在区间$(1, \sqrt{\text{Max Prime}} + 1)$ 中。故 n 模2和区间$(1, \sqrt{\text{Max Prime}} + 1)$ 所有的奇数(除2外所有的偶数都是合数) 的所有结果都不是0,则 n 为素数。算法(递归算法一)描述如下:

步 1 输出素数 2,3,5

步 2 for i = 7 to MaxNumber

步 2.1 for j = 3 to SqrtMaxNumber

步 2.1.1 若 $i\%j = 0$,跳转至步 2.3

步 2.1.2 $j = j + 2$

步 2.2 SqrtMaxNumber = $\sqrt{i} + 1$,输出 i 为素数

步 2.3 $i = i + 2$

在上述算法中,除 2 外所有的偶数都是合数,所以将自然数简单划为奇数和偶数两大类,以降低代码的循环次数。但奇数中仍然存在着大量的合数,最好再次降低遍历的次数,以加快代码的执行效率。自然数可表示为:$6k + i (k \in N, i = 0,1,2,3,4,5)$。显然,当 $k \geqslant 0$ 时,$6k, 6k+2, 6k+3, 6k+4$ 都是合数,只有形如 $6k+1$ 和 $6k+5$ 的自然数存在素数的可能。

推论 2　除 2 和 3 之外,素数均可表示成 $6k \pm 1 (k \in N)$ 的形式[3]。

定理 1　若 $6k \pm 1$ 是合数,则存在 $6k \pm 1 = (6k_1 \pm 1)(6k_2 \pm 1) \cdots (6k_n \pm 1)$,其中 $6k_n \pm 1$ 均为素数并且 $6k_1 \pm 1 \leqslant 6k_2 \pm 1 \leqslant \cdots \leqslant 6k_n \pm 1$。

证明:由于 $(6k_1 \pm 1)(6k_2 \pm 1) = 36k_1k_2 \pm 6k_1 \pm 6k_2 \pm 1 = 6(6k_1k_2 \pm k_1 \pm k_2) \pm 1 = 6m \pm 1$,故 $6k \pm 1$ 形式的数的因子形式均为 $6k_n \pm 1$。任意一个合数可表示为多个素因子之积,故定理成立。

(2)根据上述分析,对上述算法进行改进,提高程序的执行效率。算法(递归算法二)描述如下:

步 1 输出素数 2,3,5

步 2 定义 bool 参数 aDecide = false

步 3 for i = 7 to MaxNumber

步 3.1 若 $i\%2 = 0 \,||\, i\%3 = 0 \,||\, i\%5 = 0$,跳转至步 3.4

步 3.2 for j = 7 to SqrtMaxNumber

步 3.2.1 若 $i\%j = 0 \,||\, i\%(j+4) = 0$,跳转至步 3.4

步 3.2.2 $j = j + 6$

步 3.3 SqrtMaxNumber = $\sqrt{i} + 1$,输出 i 为素数

步 3.4 若 aDecide = true, $i = i + 2$;否则, $i = i + 4$

步 3.5 aDecide = ! aDecide

1.2　素数筛法

把不超过 N 的一切自然数按大小关系排成一串,首先划去 1,在 2 的上面画一个圆圈,然后划去 2 的其他倍数;第一个既未画圈又没有被划去的数是 3,将它画圈,再划去 3 的其他倍数;现在既未画圈又没有被划去的第一个数是 5,将它画圈,并划去 5 的其他倍数。依此类推,一直到所有小于或等于 n 的各数都画了圈或划去为止。这时,表中画了圈的以及未划去的那些数正好就是小于 n 的素数[1]。这种方法是公元前三世纪幼拉脱斯展纳(Eratosthenes)提出的,它就像是用筛子从自然数中筛出素数,故称为 Eratosthenes 筛选法。

推论 3　如果 k 为素数,则 $k \times j (j \in N^+)$ 必为合数。

(1)定义一个 bool 型数组 prime[$n+1$]。根据 $6k \pm 1 (k \in N^+)$ 理论,当 $i = 2$ 或 $i = 3$ 或 $i = 6k+1$ 或 $i = 6k+5 (k \in N^+)$ 时,prime[i] = true;否则 prime[i] = false。寻找第一个 prime[i] = true$(i: 5 \mapsto \sqrt{n})$ 的素数 i,i 的倍数 $j (j \in [0, n/i])$ 均为合数,则 prime[j] = false。最后所有满足 prime[i] = true$(2 \leqslant i \leqslant n)$ 的 i 就是区间[2,n] 的所有素数。算法(素数筛法一)描述如下:

步 1 定义 bool 参数 aDecide = true

步 2 定义 int 参数 SqrtMaxNumber = $\sqrt{\text{MaxNumber}}$

步 3 for i = 1 to MaxNumber

步 3.1 若 $i \neq 1 \&\& (i = 2 \,||\, i = 3 \,||\, i\%6 = 1 \,||\, i\%6 = 5)$,prime[$i$] = true,否则,prime[$i$] = false

步 3.2 $i = i + 1$

步 4 for i = 5 to SqrtMaxNumber

步 4.1 若 prime[i] = true

步 4.2 for $j = i + i$ to MaxNumber

步 4.2.1 prime[j] = false

步 4.2.2 $j = j + i$

步 4.3 若 aDecide = true, $i = i + 2$;否则, $i = i + 4$

步 4.4 aDecide = ! aDecide

步 5 for i = 2 to MaxNumber

步 5.1 若 prime[i] = true,输出素数 i

步 5.2 $i = i + 1$

若用 Eratosthenes 筛选法对素数 k 筛选,去除合数 $k \times j (j \in N^+)$。寻求到第一个素数 $i (i > k)$ 时,其 $i \times k_n$ ($k_n < i$ 且 k_n 为素数)的合数已经被剔除。

推论 4　在 Eratosthenes 筛选法中对寻求到的第一个素数 k,在区间 $(1, k^2)$ 没有被划掉的数均为素数。

(2) 在上述算法中 $j \in [2,n]$ 且 $j\%6 = 0$,则

prime[j] 会被重复遍历。另外,数组赋值时已去除偶数,将内层循环 j 限定为奇数。故当素数为 i ($i \in [2,\sqrt{n}]$) 时,$j = i*i + ik(k \in 2N, j < n)$ 。算法(素数筛法二)描述如下:

步1 定义 bool 参数 aDecide=true

步2 定义 int 参数 SqrtMaxNumber= $\sqrt{\text{MaxNumber}}$

步3 for i = 1 to MaxNumber

步3.1 若 $i \neq 1$&&(i=2 || i=3 || i%6=1 || i%6=5),prime[i]=true,否则,prime[i]=false

步3.2 $i = i + 1$

步4 for i = 5 to SqrtMaxNumber

步4.1 若 prime[i]=true

步4.2 for $j = i * i$ to MaxNumber

步4.2.1 prime[j]=false

步4.2.2 $j = j + i + i$

步4.3 若 aDecide=true, $i = i + 2$;否则, $i = i + 4$

步4.4 aDecide=! aDecide

步5 for i = 2 to MaxNumber

步5.1 若 prime[i]=true,输出素数 i

步5.2 $i = i + 1$

1.3 Miller 检验

Miller 检验是强烈依赖于广义黎曼猜想[Extended Riemann Hypothesis]的算法。猜想[ERH]对于一个素数 p ,第一个模 p 非剩余小于 $2(\ln p)^2$。Miller 检验算法检验自然数是否为素数,只需 $O(n^4 \log n \log n)$ 步。为了描述算法引入一个符号定义 $\#_2(n-1)$,它在算法描述过程中将要被用到。

定义2 $\#_2(n-1) = \min\{k; 2^k \mid n-1\}$

步1 输出素数2,3,5

步2 定义 bool 参数 aDecide=false

步3 for i = 7 to MaxNumber

步3.1 若 $i = a^k (k > 1)$,跳转至步3.4

步3.2 for j = 2 to $f(n) = 2(\ln i)^2$

步3.2.1 若 $i\%j = 0$,跳转至步3.2.4

步3.2.2 若 $a^{n-1}\%n \neq 1$,跳转至步3.2.4

步3.2.3 for k = 1 to $\#_2(n-1)$

步3.2.3.1 若 $((a^{(n-1)/2^k}\%n) - 1, n) \neq 1$,跳转至步3.2.4

步3.2.3.2 $k = k + 1$

步3.2.4 $j = j + 1$

步3.3 输出素数 i

步3.4 若 aDecide=true , $i = i + 2$;否则, $i = i + 4$

步3.5 aDecide=! aDecide

1.4 AKS 算法

2002年8月,印度的 Manindra Agrawal 教授和他的两个学生 Neeraj Kayal、Nitin Saxena 设计了一种被称为 AKS 的素性测试算法。该算法是素数研究领域的一个重大突破,它能够在不基于任何假设的情况下,通过多项式时间表达式内对自然数的素性进行确定性判定。AKS 算法是一个不需要特殊的椭圆曲线数学及类似知识,仅利用了代数数论和有限域中的结论来解决问题。尽管思想相对较简单,但是 AKS 算法将对学术界产生广泛的影响,尤其对整数理论和计算复杂性理论产生巨大的影响。算法描述如下:

步1 输出素数2,3,5

步2 定义 bool 参数 aDecide=false

步3 for i = 7 to MaxNumber

步3.1 若 $i = a^k (k > 1)$,跳转至步3.7

步3.2 找 $r > 0$,满足 $O_r(n) > 4\log^2 i$

步3.3 任意 $0 < j \leqslant r$,计算 (j,i)

步3.3.1 若 $1 < (j,i) < i$,跳转至步3.7

步3.4 若 $i \leqslant r$,输出素数 i,跳转至步3.7

步3.5 for j = 1 to $2\sqrt{\varphi(x)}\log i$

步3.5.1 若 $(x+a)^n - (x^n + a) \neq 0\%(x^r - 1, n)$,跳转至步3.7

步3.5.2 $j = j + 1$

步3.6 输出素数 i

步3.7 若 aDecide=true, $i = i + 2$;否则, $i = i + 4$

步3.8 aDecide=! aDecide

2 素数算法分析

Miller 检验和 AKS 算法适于对自然数的素性探测,并不适于寻求 $n(n \geq 10^6)$ 内中的所有素数。通过实现代码和变化 n 值,本节对递归试除法和素数筛法四个算法进行数据分析。

根据表1和图1、图2进行对比分析:递归试除法

表1 素数确定性算法所需执行时间

算法类型	$k \times 10^6$ 算法执行时间/毫秒(ms)									
	1	2	3	4	5	6	7	8	9	10
递归算法一	1500	3968	7031	10578	14531	18750	23328	28187	33312	38656
递归算法二	984	2640	4625	6937	9500	12281	15265	18437	21750	25265
素数筛法一	78	156	242	335	421	507	593	679	765	859
素数筛法二	62	133	195	273	343	413	492	554	640	703

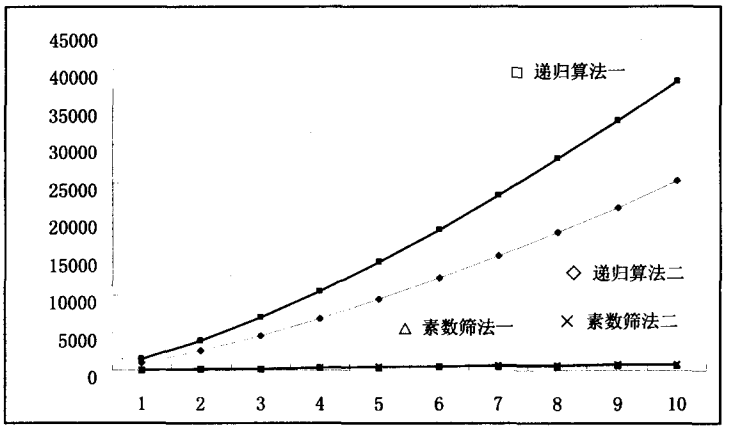

图 1　素数确定性算法所需执行时间

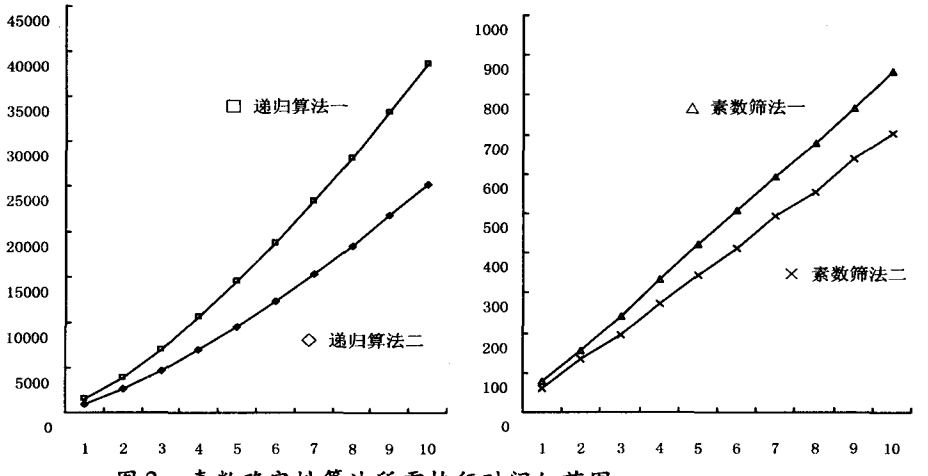

图 2　素数确定性算法所需执行时间细节图

随 n 值逐渐增大,曲线切点斜率逐渐增大,说明算法执行时间指数倍增长;素数筛法随 n 值逐渐增大,曲线切点斜率保持平稳,说明算法执行时间直线增长。据此分析,素数筛法的效率比递归试除法相对较高且较稳定,$6k \pm 1$ ($k \in N$) 理论的效率比奇偶数判断相对较高且较稳定。

3　结束语

文中结合素数可表示为 $6k \pm 1$ ($k \in N$) 的数学理论,对素数的确定性算法进行改进。利用图表的方式对算法执行时间和时间曲线进行分析,得出筛法的执行效率和稳定性远高于试除法、$6k \pm 1$ 法优于奇偶数方法的结论。但当寻求 $n(n > 10^{10})$ 内中的所有素数,算法仍需改进。

参考文献:

[1] 裴定一,祝跃飞. 算法数论[M]. 北京:科学出版社,2002.

[2] Cohen H. A Course in Computational Algebraic Number Theory[M]. 北京:世界图书出版公司,1996.

[3] 王　娅, 杨晨曦. 大于 3 的素数的一个性质[J]. 云南民族学院学报(自然科学版),2000,9(4):217-218.

[4] Quesada A R. On the K-th extension of the Sieve of eratosthenes[J]. International Journal of Mathematics and Mathematical Sciences, 1995, 18(3): 539-544.

[5] Miller G L. Riemann's Hypotheses and Tests for Primality[J]. J Comput Sys Sci ,1976(13) :300-317.

[6] Agrawal M, Kayal N, Saxena N. Primes is in P, Preprint[M]. Kanpur: IIT Kanpur,2002.

[7] 张　琦,许　勇. Matlab 环境下素数筛选算法的分析及比较[J]. 计算机技术与发展,2009,19(3):95-98.

[8] 陈晓文 ,郑建德. 新型大素数快速并行搜索策略[J]. 厦门大学学报:自然科学版,2008,47(2):207-210.

[9] 张　宏,刘晓霞,张　若. RSA 公钥密码体制中安全大素数的生成[J]. 计算机技术与发展,2008,18(9):131-133.

[10] 张四兰,夏静波,余荣威. 可信赖的高效素数生成和检验算法[J]. 计算机工程与应用,2005,41(30):31-34.

[11] 姜丽华,马永光. AKS 算法对现代密码学的影响[J]. 微机发展(现更名:计算机技术与发展), 2004,14(4):104-106.

[12] 金正平,温巧燕. AKS 素性测定算法的一个改进版本在 PC 上的实现[J]. 四川大学学报(工程科学版),2009,49(1):147-152.

(上接第 25 页)

明,耿　岳,译. 北京:中国电力出版社,2005.

[8] 毛德操,胡希明. Linux 内核源代码情景分析[M]. 杭州:浙江大学出版社,2001.

[9] 夏　鸿,程克非. 基于零拷贝技术的千兆网络性能优化研究[J]. 网络与通信,2008,24:155-157.

[10] Liu Tianhua, Zhu Hongfeng, Chang Guiran, et al. The design and implementation of zero-copy for linux[C]// Eighth International Conference on Intelligent Systems Design and Applications. [s. l.]:[s. n.],2008:121-123.

[11] 徐　力. 一种基于实时环境的零拷贝通信技术的研究[D]. 武汉:华中科技大学,2006.

[12] Benvenuti C. Understanding LINUX network internals[M]. California: O'Reilly Meida, 2006.