Some primality testing algorithms [*][†]

R.G.E. Pinch
Department of Pure Mathematics and Mathematical Statistics,
University of Cambridge

27 November 1993

Abstract

We describe the primality testing algorithms in use in some popular computer algebra systems, and give some examples where they break down in practice.

1 Introduction

In recent years, fast primality testing algorithms have been a popular subject of research and some of the modern methods are now incorporated in computer algebra systems (CAS) as standard. In this review I give some details of the implementations of these algorithms and a number of examples where the algorithms prove inadequate.

The algebra systems reviewed are Mathematica, Maple V, Axiom and Pari/GP. The versions we were able to use were Mathematica 2.1 for Sparc, copyright dates 1988-1992; Maple V Release 2, copyright dates 1981-1993; Axiom Release 1.2 (version of February 18, 1993); Pari/GP 1.37.3 (Sparc version, dated November 23, 1992). The tests were performed on Sparc workstations.

Primality testing is a large and growing area of research. For further reading and comprehensive bibliographies, the interested reader could consult the works of Bressoud [13], Brillhart et al [14], Knuth [28], Koblitz [29], Ribenboim [44][45] or Riesel [46].

2 Primality tests

The first and most obvious test is *trial division*: that is, given an integer n, try all integers from 2 up to $\sqrt{n}$ to see whether any are factors of n. If one is found, then n is composite; if not, then n is prime.

This test has two drawbacks, the most obvious being that the time taken (even with obvious refinements) is, in the worst case (which will occur when n is prime), of the order of $\sqrt{n}$ and this is not a practical proposition for n of the order likely to occur in practice. This is in itself a sufficient reason for searching for other, more efficient, tests. The second drawback, less obvious, is that the test does not always produce a *certificate* for its answer. When n is composite, then a factor f will be found, and the character of n can then be verified quickly: it is much easier to show that n is divisible by some number f than to find f in the first place. Unfortunately when n is prime, all that emerges from the computation is a bare assertion that no factor was found, and in order to check the calculation (for example, to check whether any error has occurred, or to convince a sceptical onlooker), it is necessary to repeat it all over again.

Most modern algorithms depend in some way on the converse of Fermat's Theorem, that if p is prime then, for $1 \leq a < p$, we have $a^{p-1} \equiv 1 \bmod n$. Given a number n to be tested for primality, we see that if there is an a with $1 < a < n$ and $a^{n-1} \not\equiv 1 \bmod n$, then n must be composite, and a is a certificate for the compositeness of n (although no factor of n need have been found). Since a^{n-1} can be computed modulo n in about $\log n$ multiplications modulo n, this condition is very fast to check. This is the *Fermat test*, and a is the *base*.

[*]AMS Subject classification: Primary 11Y11; Secondary 11A51
[†]An uncorrected version of this paper appeared in *Notices of the American Mathematical Society*, November 1993.

What happens if we find that $a^{n-1} \bmod n$ is 1? We cannot conclude that n is prime from just one test: for example, given $n = 341 = 11 \times 31$ and $a = 2$ we find that $a^{n-1} = 2^{340} \equiv 1$. We describe n as a *Fermat pseudoprime base 2*.

It is not hard to show that there are infinitely many Fermat pseudoprimes to any given base, although it is true that Fermat pseudoprimes are rarer than primes. Put

$$L(X) = \exp\left(-\frac{\log X \log\log\log X}{\log\log X}\right).$$

If $P_{\mathrm{F},a}(X)$ denotes the number of Fermat pseudoprimes base a less than X, then Pomerance has shown [40][41] that

$$\exp\left((\log X)^{5/14}\right) \le P_{\mathrm{F},a}(X) \le XL(X)^{1/2}$$

for sufficiently large X; compare this with the number $\pi(X)$ of primes up to X, which is well-known to be asymptotic to

$$\frac{X}{\log X} = X \exp\left(-\log\log X\right).$$

For $X = 10^{13}$, calculations [37] show that $P_{\mathrm{F},2}(X) = 264239$, compared with $\pi(X) = 37607912018$.

Even worse, if we take $n = 561 = 3 \times 11 \times 17$, then we find that $a^{n-1} \equiv 1 \bmod n$ for every base a which is coprime to n. Such an n is called an *absolute Fermat pseudoprime*, or a *Carmichael number*, and it has recently been proved by Alford, Granville and Pomerance [1] that there are infinitely many Carmichael numbers — see Granville's survey article [23] . Carmichael numbers are of course less numerous than Fermat pseudoprimes to any fixed base: letting $C(X)$ denote the number of Carmichael numbers up to X, we have [1] , [40]

$$X^{2/7} \ll C(X) \ll XL(X)$$

and for $X = 10^{16}$ we have [38] $C(X) = 246683$ compared with $\pi(X) = 279238341033925$.

It seems plausible to conjecture that in fact both $C(X)$ and $P_{\mathrm{F},a}(X)$ exceed $X^{1-\epsilon}$ for sufficiently large values of X.

We describe a number which passes the Fermat test (or one of the derivatives to be described later) as a *(Fermat) probable prime*. We should emphasise at this point that the phrase "probable prime" is to be read as if it were a single word: [1] it has become standard because a number which passes the test is "probably prime" in the intuitive sense that there are "fewer" pseudoprimes than primes. We shall make this qualitative shortly.

To improve the performance of the Fermat test we observe that, since we may assume n is odd, $n - 1$ is even and so a^{n-1} is a square. If n is a prime, 1 has only two square roots, $\pm 1 \bmod n$. If an odd number n passes the Fermat test (faster primality tests are possible for even numbers!) then $a^{n-1} \equiv 1$ is a square, so we require that $a^{(n-1)/2} \equiv \pm 1 \bmod n$. This further requirement we call the *Fermat–Euler test* (although the result was already known to Fermat): Lehmann [30].

Iterating, we arrive at the *strong* or *Miller–Rabin* test [31][32][43]. Write $n - 1 = 2^r s$, where s is odd. For base a, form the Miller–Rabin sequence

$$a^s, a^{2s}, \ldots, a^{2^{r-1}s} \equiv a^{\frac{n-1}{2}}, a^{2^r s} \equiv a^{n-1} \bmod n$$

in which each term is the square root of its successor. Then n passes the test base a if the last term in the sequence is 1 (this is just the requirement of the Fermat test), and the first occurrence of 1 either is the first term or is preceded by -1.

Since the strong test includes the Fermat test, the number of strong pseudoprimes for a fixed base a up to X, $P_{\mathrm{MR},a}(X)$ is bounded above by $P_{\mathrm{F},a}(X)$, but the best upper bound known is no better than that implied by the upper bound for $P_{\mathrm{F},a}(X)$ above. For $X = 10^{13}$ we have [11], [37] $P_{\mathrm{MR},2}(X) = 58897$.

As with the Fermat test, the strong test with a single base does not characterise primes: for example, if $n = 2047 = 23 \times 89$, then $n - 1 = 2^1.1023$ and the Miller–Rabin sequence is $2^{1023} \equiv 1, 2^{2046} \equiv 1$. So 2047 is an example of (indeed, the smallest) *strong pseudoprime base 2*. We have however made an advance: *if n is composite then it passes the strong test to at most $1/4$ of the bases $a \bmod n$.* (Thus composite numbers can be detected in random non-deterministic polynomial time.)

Miller [31][32] observed that a theorem of Ankeney [2] could be applied to turn the strong test into a conditional polynomial-time characterisation of primes: the quantitative version due to Bach

[1] Students of the English legal system may be reminded of the title of Lord Privy Seal, who is neither a Lord, nor . . .

[8] states that *provided a suitable generalisation of the Riemann hypothesis (GRH) holds, a number n is prime iff it passes the strong test to all bases a with* $1 < a \leq 2(\log n)^2$.

If one does not assume the GRH then a result of Burgess [15] implies that testing up to $n^{.134}$ is sufficient.

In the opposite direction, it follows from the result of Alford, Granville and Pomerance [1][23] that there are infinitely many numbers which are strong pseudoprimes with respect to any fixed finite set of bases.

We shall call a primality test *probabilistic* if it employs some random input, so that the precise sequence of operations performed may vary from call to call, even if the input parameters remain the same. A test is *deterministic* otherwise: such a test will repeat exactly the same operations and give exactly the same output, for the same input.

Consider a probabilistic algorithm consisting of t rounds of the strong test with bases chosen uniformly and independently at random modulo n. If the input n is composite, the chance of it passing such a test is at most 4^{-t}. Assuming that there is a probability distribution on the input n, we can say

$$\begin{aligned}
&\text{prob (test gives wrong answer)} \\
&\quad = \sum_n \text{prob (test gives wrong answer for } n\text{)prob (}n \text{ is input)} \\
&\quad = \sum_n \text{prob (}n \text{ passes } t \text{ rounds}|n \text{ is composi te)} \cdot \\
&\qquad\qquad \text{prob (}n \text{ is composite}|n \text{ is input)prob (}n \text{ is input)} \\
&\quad = \leq 4^{-t}.
\end{aligned}$$

Let us note at this point that if the test is deterministic, we can make no such assertion without some knowledge of a probability distribution on the input n. For example, with a deterministic test always using base 2, and a probability distribution on the input concentrated at $n = 2047$, the test is certain to produce the wrong answer.

It is possible to do considerably better given reasonable assumptions on the probability distribution on the input numbers n. If, for example, we assume that n is distributed uniformly over all k-bit odd integers, then it can be shown, using the methods of Kim, Pomerance, Damgård and Landrock [19][20][27] that for $k \geq 100$ and $5 \leq t \leq k/9 + 2$,

$$\text{prob (}n \text{ is composite}|n \text{ passes } t \text{ rounds)} \leq 0.4\,k2^t \left(0.6 \cdot 2^{-2\sqrt{k(t-2)}} + 2^{-t\sqrt{k/2}}\right);$$

and for $t > k/9 + 2$,

$$\text{prob (}n \text{ is composite}|n \text{ passes } t \text{ rounds)} \leq 0.4\,k \left(11.32\sqrt{k}\,2^{-2t-k/3} + 2^{t-t\sqrt{k/2}}\right).$$

For $t = 6$ and $k = 250$ this is less than 2^{-56} and for $t = 10$ and $k = 2000$ the probability of a wrong answer is less than 2^{-228}.

If we consider a deterministic algorithm using t rounds of the strong test, say with the first t primes as bases, we cannot immediately say that the same estimates apply. It is plausible to suppose that tests with distinct prime bases behave independently, although this will not be so for multiplicatively dependent bases: if n is a strong pseudoprime base a and base b it is likely, although not certain, to be a strong pseudoprime base ab.

Assuming that a deterministic test with a fixed set of t multiplicatively independent bases behaves in the same way as a probabilistic test with random bases, then the test should characterise primes for values of k for which the expected number of pseudoprimes is less than 1. This suggests that k should not exceed $3t$ or, as a rule of thumb, t should be about the number of decimal digits in the input. The result of Bach implies that, if the GRH holds, taking $t > 2k^2$ is sufficient to characterise primes.

The number $341550071728321 = 10670053 \times 32010157$ of 15 digits, 49 bits, is a strong pseudoprime for all bases up to 22, that is, for 8 primes; the number

$$68528663395046912244223605902738356719751082784386681071$$

of 56 digits, 186 bits, is a strong pseudoprime for all bases up to 100, that is, for 25 primes.

3 Quadratic tests

A common feature in the Fermat test and its refinements is the use of a group defined algebraically modulo n which has a predictable number of elements when n in prime: the group in this case being the multiplicative group modulo n with order $n-1$ when n is prime.

We can extend our tests by considering further groups. One important case is taking the multiplicative group of the quadratic extension $\mathbf{Z}/n[\sqrt{d}]$ when d is not a quadratic residue of n. If n is prime, then this quadratic ring is the finite field of n^2 elements, with a multiplicative group of order n^2-1. The elements of $\mathbf{Z}/n[\sqrt{d}]$ may be represented in the form $x+y\sqrt{d}$ with x and y taken modulo n and multiplication defined in the obvious way, with $\sqrt{d}\cdot\sqrt{d}$ defined to be $d \bmod n$. The *norm* of such an element will be x^2-dy^2, and the elements of norm 1 form a subgroup of the multiplicative group of exponent $n+1$ when n is prime. If we let d be the discriminant of the equation $X^2-tX+u=0$, and let α be a root, then if the Jacobi symbol $\left(\frac{d}{n}\right)$ is -1, and n is prime, the map $\alpha \mapsto \alpha^n$ will be the Frobenius automorphism of the finite field GF $\left(n^2\right)$ and so α^n must be equal to α', where $(x+y\sqrt{d})' = x-y\sqrt{d}$. This is the *Lucas test.*

The equivalent of the Fermat test for this group would be to require $\alpha^{n^2-1}=1$ but this is not as strong as the Lucas test, as we shall see. We can make a better parallel with the Fermat test by considering the elements of norm 1, i.e. with $u=1$. The condition $\alpha^n=\alpha'$ is equivalent to $\alpha^{n+1}=\alpha\alpha'=1$. The *norm-one Lucas test* consists of taking the smallest t such that the Jacobi symbol is -1 and then requiring that $\alpha^{n+1}\equiv 1 \bmod n$.

As before, we call n a *Lucas probable prime* if it passes the Lucas test, and a *Lucas pseudoprime* if it is a composite probable prime. A refinement of the norm-one Lucas test proceeds by considering the iterated square roots of α^{n+1}: as in the Miller–Rabin test, let s be the odd part of $n+1$ and then repeatedly square α^s. Let us call this the *strong norm-one Lucas test.*

Letting $P_{\mathrm{L},d}(X)$ denote the number of Lucas pseudoprimes, with respect to d as a quadratic non-residue, up to X, we have

$$\exp\left((\log x)^c\right) \le P_{\mathrm{F},a}(X) \le XL(X)^{1/3}$$

for some constant c.

We should note that finding a quadratic non-residue is not guaranteed to be easy: the best results are those of Bach (on the GRH) and Burgess (unconditionally) mentioned above.

Pomerance et al [39] describe two methods of finding a suitable d and element α. They have issued a challenge (with a total prize now \$620) for an example of a composite number which passes both the strong test base 2 and one of the versions of the Lucas test they propose, or for a proof that no such number exists. At present, the prize is unclaimed: the computations of [35][37] show there is no such number less than 10^{13}.

4 Primality proofs

The probable-prime tests we have described all test for properties which n must have it is prime. Hence the failure of any of these tests proves the compositeness of n, and in all the methods described, the test also furnishes a certificate of the compositeness which may be verified quickly (in time polynomial in $\log n$). We now turn to methods for proving the primality of n. It is well-known that the only odd numbers n for which the multiplicative group modulo n is cyclic are those n which are prime powers. Since it is possible to test whether n is a perfect power quickly, we assume that n is known not to be a perfect power, so that n is prime if and only if the multiplicative group is cyclic. To prove n prime, it suffices to find an element of exact order $n-1$, and indeed, to find elements whose orders have least common multiple $n-1$. A certificate then consists of a list of such elements together with their orders.

Unfortunately, to exhibit an element of exact order d, it is necessary to show that the order is not any proper factor of d, and this requires factorisation of d. So we need to be able to factorise $n-1$ in order to use this method.

Assuming for the moment that we can do this, we obtain the factorisation of $n-1$ as a list of primes and their exponents. The factorisation method will undoubtedly use some form of primality test to decide when a prime factorisation has been obtained. To certify that n is prime will require a certificate that the factors of $n-1$ are themselves prime and so the certification will be recursive: Atkin has called this “Downrun”. Verification of the certificate is fast: see Pratt [42].

This proof method works well on numbers of special form, for example, $n-1 = 2^r s$ with $s < 2^r$. Suppose that $a^{2^{r-1}} \equiv -1 \bmod n$. Then if n is composite, take p to be the smallest prime factor of n, so that $p < \sqrt{n}$. In particular, $p < 2^r$. But a is an element of order 2^r modulo p, so $2^r \leq p-1$, a contradiction. The partial factorisation if $n-1$ together with the base a forms a certificate of primality.

5 Elliptic curve tests

The primality proof method just described depends on the factorisation of $n-1$. In cases where this is difficult, one can work in a suitable quadratic extension (as in the Lucas method) and instead try to factorise $n+1$.

Morain [6][7][33][34] suggested replacing these multiplicative groups by the group of points on an elliptic curve modulo n, which can have any order between $n+1 \pm 2\sqrt{n}$ when n is prime. The order of this group is determined by the theory of complex multiplication, and the certificate consists of the order, its factorisation, the points on the curve of orders with least common multiple the order of the group, and (recursively) certificates of the primality of the factors.

6 The tests performed

We used three lists of composite numbers to exercise the primality testing routines of the various systems. The first list, $\mathcal{X}$, was that of the 246683 Carmichael numbers up to 10^{16} described in [36][38]; the second, $\mathcal{Y}$, was that of the 264239 Fermat pseudoprimes base two [35][37], and the third, $\mathcal{Z}$, was a "zoo" of special cases specifically intended to defeat various tests, largely obtained from Arnault [5][3][4], Bleichenbacher [10][11], and Davenport [21][22].

7 The Maple `isprime` function

Maple V provides a function `isprime`, (also invoked as `type/primeint`).

The Maple V language reference manual [16] §1.2, p.7 simply asserts that Maple can test integers for primality. The Maple library reference manual [17] §2.1.164, p.120 and the on-line documentation state

> `isprime (n, iter)`
>
> The function `isprime` is a probabilistic primality testing routine.
>
> It returns `false` if n is shown to be composite within *iter* tests and returns `true` otherwise. If `isprime` returns `true`, n is "very probably" prime – see Knuth volume 2, second edition, section 4.5.4, algorithm P for a reference.

and

> `type (expr, primeint)`
>
> This function returns `true` if *expr* is a prime integer and `false` otherwise.
>
> The function `isprime` is used to check the primality of *expr*, once *expr* has been determined to be an integer.

The algorithm employed by `isprime` tests initially for divisibility by primes up to 1000, and then performs the strong test with the first *iter* primes as base (up to a maximum of 25 tests). The default value of *iter* is 5.

There are 2 numbers in list $\mathcal{Y}$ which pass the strong test bases 2,3,5,7 and 11 and which have no factor under 1000: they are $2152302898747 = 6763 \times 10627 \times 29947$ and $3474749660383 = 1303 \times 16927 \times 157543$, and `isprime` accordingly declares them prime. (Curiously, there are no such pseudoprimes with a factor less than 1000.) We find that 3474749660383 is a strong pseudoprime base 13 as well, and so passes `isprime` with *iter* set to 6, the smallest number to do so.

There are three further numbers from list $\mathcal{X}$ which pass `isprime`: $10710604680091 = 3739 \times 18691 \times 153259$, $4498414682539051 = 46411 \times 232051 \times 417691$ and $6830509209595831 = 21319 \times 106591 \times 3005839$.

Surprisingly, the integer factorisation function `ifactor` gave the correct answer for the composite input 3474749660383, although for the remaining four numbers it returns the number itself.

The `ifactor` function is described in [17] §2.1.151, p.107 and the on-line documentation as

> `ifactor` returns the complete integer factorisation of n.

The first step in this function is to extract prime factors up to 1699. Then, in subprocedure `ifactOth`, a call is made to `isprime` before embarking on any of the more sophisticated algorithms which `factor` can use. This explains the discrepancy between the results of `isprime` and `ifactor` on $1303 \times 16927 \times 157543$.

There is also a discrepancy between the behaviour of `isprime` and the `safeprime` function from the `numtheory` package [17] §4.4.26, p.528.

> The function `safeprime` will compute the smallest safe prime that is greater than n. A safe prime is a number p such that p is prime and $(p-1)/2$ is prime.

The `safeprime` function does not call `isprime` internally, but declares a number to be prime if it has no factor ≤ 113 and passes the Fermat–Euler test for bases 2,3,5,7 and 11. This test is rather weaker than the strong test used by `isprime` and fails, for example, by declaring $p = 1879894019$ to be a safe prime even though $(p-1)/2 = 939947009 = 263 \times 1049 \times 3407$ and is declared composite by `isprime`. This is the smallest counter-example: there are four such p up to 10^{12}. Fortunately if p is declared a safe prime by this method, and $(p-1)/2$ is indeed prime, then it will be true that p is prime as well, since 2 is an element of order at least $(p-1)/2$ modulo p. I was not able to find any examples with both p and $(p-1)/2$ composite.

Gaston Gonnet has informed me that he plans to include a stronger version of `isprime` in a new release.

8 The Mathematica `PrimeQ` and `ProvablePrimeQ` functions

The Mathematica version 2 number theoretic functions are reviewed by Wagon [48] (who discusses version 1 functions in [47] §1.1).

The Mathematica built-in primality test `PrimeQ` is described briefly in [49] (first edition)

> `PrimeQ[`*expr*`]` yields `True` if *expr* is a prime number and yields `False` otherwise.

and less tersely in the on-line documentation

> `PrimeQ[`*expr*`]` yields `True` if *expr* is a prime number, and yields `False` otherwise. In the current version of Mathematica, the algorithm used for large integers is probabilistic, but very reliable (pseudoprime test and Lucas test).

Unfortunately, all these assertions are incorrect. The value `True` is returned if the argument is a probable prime, and there are at least two pseudoprimes which it fails to detect. Finally, it appears from the more extensive description [12] below that the algorithm is in fact deterministic. (Perhaps the documenter confused a probable-prime test with a probable prime-test.)

> In Mathematica 2.0, the built-in function `PrimeQ` uses the Rabin strong pseudoprime test and the Lucas test. This procedure has been proved correct for all $n < 2.5 * 10^{10}$ and for special numbers of the form $a2^b + 1$, where $a < 2^b$. As of April 1991, the procedure has not been proved correct for larger n, nor has a counterexample been found. However, it is a mathematical theorem that when `PrimeQ[`n`]` returns `False`, the number n is genuinely composite. Thus `PrimeQ[`n`]` can only fail if n is composite but `PrimeQ` declares it to be prime. It is important to note that `PrimeQ` is deterministic; no computations based on random numbers are involved.

We note in passing that it is not correct to state that if n is a strong probable prime base 2 and n is of the form $a2^b + 1$ where $a < 2^b$, then n is prime. Consider $n = 4294967297 = 641 \times 6700417 = 2^{2^5} + 1$ (the fifth, and first composite, Fermat number). It is of the special form stated, and a strong pseudoprime base 2: the Miller–Rabin sequence of repeated squares of 2 clearly contains $2^{2^5} \equiv -1 \bmod n$. Clearly any composite Fermat number will have this property. For numbers of the special form stated, as described above, the strong test is capable of proving primality if that the occurrence of -1 in the sequence is sufficiently late.

The second edition of [49] states correctly

> • In *Mathematica* 2.0, the built-in function `PrimeQ` uses the Rabin strong pseudoprime test and the Lucas test. This procedure has been proved correct for all $n < 2.5 \times 10^{10}$. As of 1990, however, the procedure has not been proved correct for larger n and it is conceivable that it could claim that a composite number was prime (though not vice-versa). Nevertheless, as of 1990, no example of such behaviour is known.

On applying the function to lists $\mathcal{X}$ and $\mathcal{Y}$, there were two composite numbers for which the test returns the result `True`: $38200901201 = 89 \times 11551 \times 37159$ and $6646915915638769 = 7309 \times 321553 \times 2828197$.

Examination of the source code shows that the function `PrimeQ` first performs the strong test base 2. Next the smallest $t \geq 2$ for which the Jacobi symbol $(1 - 4t^2|n) = -1$ is found. Let α denote a root of $X^2 - t^{-1}X + 1$ modulo n. Then a variant of the strong Lucas test is performed, except that iterated square roots of the $\left(n^2 - 1\right)$-power of α are used, rather than of the $(n + 1)$-power. If we consider $n = 6646915915638769$, we find that $t = 9$ and $\alpha^n \equiv \alpha \bmod n$. This would cause n to fail the Lucas test, which requires that $\alpha^n \equiv \alpha'$: the variant used by Mathematica is strictly weaker. The other exceptional number also fails the stricter Lucas test. The test performed by Mathematica is therefore not that referred to by Pomerance et al [39] : there appears to have been confusion between the Lucas test and the norm-one Lucas test.

The package `NumberTheory‘PrimeQ‘` already referred to contains `ProvablePrimeQ[`n`]`, described [12] as

> This package implements primality proving. If `ProvablePrimeQ[`n`]` returns `True`, then the number n can be mathematically proven to be prime. In addition, `PrimeQCertificate[`n`]` prints a certificate that can be used to verify that n is prime or composite. In Mathematica Version 2.0, the built-in primality testing function `PrimeQ` does not actually give a proof that a number is prime. However, as of this writing, there are no known examples where `PrimeQ` fails.

The certificate returned can be complicated, involving several methods of primality proof recursively. It would be of considerable assistance to the user if the methods were more comprehensively documented.

The description in [12] continues:

> As noted above, there is a possibility that `PrimeQ` is incorrect, i.e., it asserts that a number is prime when it is really composite. It is unclear whether `ProvablePrimeQ` always detects this, but if it does, an error message is generated and a counterexample to `PrimeQ` is returned.

Applying the `ProvablePrimeQ` function to 38200901201, the function appears to enter a loop (possibly looking for a primitive root?). Applied to 6646915915638769, the function prints out a number of error messages, finishing with the message `PrimeQCertificate::false: Warning: PrimeQCertificate has detected a counterexample to PrimeQ` and returns the value `True`: this is a bug.

The second edition of [49] states

> • In *Mathematica* version 2.0, the package `NumberTheory‘PrimeQ‘` contains a much slower `PrimeQ` based on a procedure which has been proved correct for all numbers.

This seems to be incorrect: the results from `PrimeQ` do not appear to differ if the package `NumberTheory‘PrimeQ‘` has been preloaded. Perhaps the author means “... a much slower primality test ...”, referring to `ProvablePrimeQ`.

Jerry Keiper has stated on behalf of Wolfram Research Inc. that revised code for `PrimeQ` using a strong form of the norm-one Lucas test, which deals correctly with the two counter-examples to the present version, will be incorporated in version 2.3. “Likewise the anomalies in ProvablePrimeQ are being looked into (the one has been fixed already).”

9 The Axiom `prime?` function

The Axiom package IntegerPrimesPackage includes the function `prime?` described in [26] §9.30.2, p.384, as

> The operation `prime?` returns `true` or `false` depending on whether its argument is a prime.

There is greater detail documented in the source code:

> `prime?(n)` returns `true` if n is prime and `false` if not. The algorithm used is Rabin's probabilistic primality test (reference: Knuth Volume 2 Semi Numerical Algorithms). If `prime? n` returns `false`, n is proven composite. If `prime? n` returns `true`, `prime?` may be in error however, the probability of error is very low and is zero below 25.10^9

(due to a result of Pomerance et al) and below 10^{12} due to a result of Pinch, and below 341550071728321 due to a result of Jaeschke. Specifically, this implementation does at least 10 pseudo prime tests and so the probability of error is $< 4^{-10}$. The running time of this method is cubic in the length of the input n, that is $\mathrm{O}\left((\log n)^3\right)$, for $n < 10^{20}$ beyond that, the algorithm is quartic, $\mathrm{O}\left((\log n)^4\right)$. Two improvements due to Davenport have been incorporated which catches some trivial strong pseudo-primes, such as [Jaeschke, 1991] $1377161253229053 \times 413148375987157$, which the original algorithm regards as prime.

The results referred to are Pomerance et al [39], Pinch [35], Davenport [22] and Jaeschke [24]. The algorithm is deterministic and rather sophisticated: see Davenport [22] for a full description.

The initial stage is a simple check for divisibility by small primes (up to a limit of 313).

For numbers less than 10^{20}, the next stage is to apply the strong test with a suitable set $\mathcal{B}$ of bases and a set $\mathcal{E}$ of exceptional pseudoprimes. For input up to 25.10^9, $\mathcal{B} = \{2,3,5\}$ with $\mathcal{E}$ of order 12; up to 10^{12}, $\mathcal{B} = \{2,3,7,10\}$ with $\mathcal{E}$ of order 7; and up to 10^{20}, $\mathcal{B} = \{2,3,5,7,11,13,17\}$ with $\mathcal{E}$ empty. It is shown in [39], [37] and [25] that the sets $\mathcal{E}$ are indeed the pseudoprimes for this stage of the test — that is, the answer returned is correct for all input up to at least 341550071728321.

For input greater than 10^{20}, the next stage is to perform the strong test with the first ten primes as base. Then a test is made for numbers of the form $3n+1$ or $8n+1$ a perfect square (in which case n has an obvious factorisation). Finally for n of d decimal digits, up to $d/2$ strong tests are made with successive primes as base.

The `prime?` function is correct for numbers up to 10^{20}, except that in the current distribution (updated 9 April 1993) the set of $\mathcal{E}$ of exceptional pseudoprimes up to 25.10^9 has been incorrectly transcribed, and so the test incorrectly reports that $19887974881 = 81421 \times 244261$ is prime.

There were two composite numbers for which the function incorrectly returns `true`:

$$1687908775236769118091924541714 51 = 266420043451 \times 4674035851 \times 135547039651$$

and

$$\begin{aligned} &68528663395046912244223605902738356719751082784386681071 \\ &= 18215745452589259639 \times 4337082250616490391 \times 867416450123298079. \end{aligned}$$

The first is a strong pseudoprime for bases 2 to 82; the second for bases up to 100. These numbers were obtained by Bleichenbacher [10][11].

We note that up to 10^{13} one can take $\mathcal{B} = \{2,3,5,7,11\}$ with an exceptional set $\mathcal{E}$ of size 2 [37] , or even $\mathcal{B} = \{2,3,5,7,61\}$ with no exceptions, as observed by Bleichenbacher [11].

James Davenport has told me that the misprint mentioned has been corrected, and that a new version of `isprime?` will use an improved test.

10 The Pari/GP `ispsp` and `isprime` functions

The Pari `ispsp` and `isprime` functions are described in the user manual [9] and the on-line documentation:

> `ispsp`(x): true (1) if x is a strong pseudo-prime for a randomly chosen base, false (0) otherwise. `isprime`(x): true (1) if x is a strong pseudo-prime for 10 randomly chosen bases, false (0) otherwise.

Pari declared all the numbers in lists $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ composite. Since the algorithm is probabilistic, the analysis above applies, and we conclude that for k-bit numbers, $k \geq 100$, the probability of Pari returning an incorrect answer is at most $4.1\,k\,2^{-\sqrt{8k}}$. For $k = 100$ this is less than 2^{-19}.

The Pari `factor` function, when applied to integers, calls the library routine `auxdecomp` which declares a number to be prime if it passes 10 initial rounds of the strong test with random bases, followed by a further 5 rounds for every 32 bits. Curiously, then, `factor` is more likely to detect compositeness than `isprime`.

11 Conclusions

All of the tests reviewed fall short, to some extent, of what I would look for. Among the features I regard as desirable are:

- Predictability. If a “random” choice of bases is to be used, there should be an option to reset the random number generator to a consistent initial state.
- Consistency. The same tests should be used in all routines in the package.
- Speed versus certainty. The user should be able to specify the use of a fast test with possibility of error or a slower test with “proof” status.
- Documentation. Whatever the method used, the documentation should make it clear what the algorithm is, what the known classes of exception (if any) are, and an indication of the probability of an incorrect answer. No test which may accept composite numbers should be described as a test for primality. Axiom, Maple and Mathematica all make this claim.
- Primality certificates, where provided, should be described in sufficient detail for the user, at least in principle, to check it independently.
- Nomenclature. I strongly suggest that tests for probable primality should be called by names which reflect their status, such as `IsProbPrime`.
- Power. Routines which use the strong test only should use as many bases as decimal digits in the input.

  On the basis of present knowledge, the best test would appear to be some combination of Miller–Rabin and Lucas tests.

12 Acknowledgements

I am grateful to Daniel Bleichenbacher, James Davenport, Gaston Gonnet, Dan Grayson, Jerry Keiper, Roman Maeder, Carl Pomerance, Stan Wagon and Sam Wagstaff jr for helpful discussions on the topics covered in this review. Thanks are also due to the Department of Pure Mathematics and Mathematical Statistics, the Isaac Newton Institute for Mathematical Sciences and the University Computing Service, Cambridge, and the Mathematical Sciences Research Institute, Berkeley, for use of their computer facilities.

References

[1] W.R. Alford, Andrew Granville, and Carl Pomerance, *There are infinitely many Carmichael numbers*, Annals of Maths **139** (1994), no. 3, 703–722.

[2] N.C. Ankeny, *The least quadratic non-residue*, Ann. of Math. (2) **55** (1952), 65–72.

[3] François Arnault, *Le test de primalité de Rabin–Miller: un nombre composé qui le “passe”*, Tech. report, Université de Poitiers, November 1991.

[4] ———, *Rabin–Miller primality test: composite numbers which pass it*, Preprint, Université de Poitiers, September 1992.

[5] ———, *Carmichaels fortement pseudo-premiers, pseudo-premiers de lucas*, Preprint 73, Université de Poitiers, January 1993, Chapter 6 of [**?**].

[6] A. Oliver L. Atkin and François Morain, *Elliptic curves and primality proving*, Technical report, INRIA, June 1990.

[7] ———, *Elliptic curves and primality proving*, Math. Comp. **61** (1993), 29–68, Lehmer memorial issue.

[8] Eric Bach, *Explicit bounds for primality testing and related problems*, Math. Comp. **55** (1990), no. 191, 355–380.

[9] C. Batut, D. Bernadi, H. Cohen, and M. Olivier, *User's guide to pari-gp*, 1992.

[10] D. Bleichenbacher, *Re: Pseudoprimes too strong for maple*, Usenet *sci.math* posting `1993Apr27.141249.29080 neptune.inf.ethz.ch`, April 1993.

[11] D. Bleichenbacher and Ueli M. Maurer, *Finding all strong pseudoprimes* $\leq x$, Extended abstract, 1993.

[12] P. et al. Boyland, *Guide to standard Mathematica packages*, *Mathematica* technical report, Wolfram Research Inc., Champaign, IL, 1991.

[13] D.M. Bressoud, *Factorization and primality testing*, Springer–Verlag, New York, 1989.

[14] J. Brillhart, D.H. Lehmer, J.L. Selfridge, B. Tuckerman, and S.S. Wagstaff jr, *Factorizations of* $b^n \pm 1$, second ed., Contemporary mathematics, vol. 22, Amer. Math. Soc., Providence RI, 1988.

[15] D.A. Burgess, *The distribution of quadratic residues and non-residues*, Mathematika **4** (1957), 106–112.

[16] B.W. Char, K.O. Geddes, G.H. Gonnet, B.L. Leong, M.B. Monagan, and S.M. Watt, *Maple v language reference manual*, Springer–Verlag, 1991.

[17] ———, *Maple v library reference manual*, Springer–Verlag, 1991.

[18] I.B. Damgård (ed.), *Advances in cryptology — EUROCRYPT '90*, Lecture notes in Computer Science, vol. 473, Berlin, Springer–Verlag, 1991.

[19] Ivan Damgård and Peter Landrock, *Improved bounds for the Rabin primality test*, Cryptography and coding III (M. Ganley, ed.), IMA conference series (n.s.), vol. 45, Institute of Mathematics and its Applications, Oxford University Press, 1993, Proceedings, 3rd IMA conference on cryptography and coding, Cirencester, December 1991., pp. 117–128.

[20] Ivan Damgård, Peter Landrock, and Carl Pomerance, *Average case error estimates for the strong probable prime test*, Math. Comp. **61** (1993), 177–194, Lehmer memorial issue.

[21] James H. Davenport, *Primality testing revisited*, Proc. ISSAC 1992 (New York) (P.S. Wang, ed.), Assoc. for Computing Machinery, 1992, pp. 123–129.

[22] ———, *Primality testing revisited*, Technical report TR2/93 (ATR/6) (NP2556), Numerical Algorithms Group, Oxford, May 1993.

[23] Andrew Granville, *Primality testing and Carmichael numbers*, Notices Amer. Math. Soc. **39** (1992), no. 7, 696–700.

[24] G. Jaeschke, Unpublished.

[25] ———, *On strong pseudoprimes to several bases*, Math. Comp. **61** (1993), 915–926.

[26] R.S. Jenks and R.S. Sutor, *Axiom: the scientific computation system*, Springer–Verlag, New York, 1992.

[27] Su Hee Kim and Carl Pomerance, *The probability that a random probable prime is composite*, Math. Comp. **53** (1989), no. 188, 721–741.

[28] Donald E. Knuth, *The art of computer programming: Seminumerical algorithms*, second ed., vol. 2, Addison–Wesley, Reading, Mass., 1981.

[29] Neal Koblitz, *A course in number theory and cryptography*, Graduate Texts in Mathematics, vol. 114, Springer–Verlag, New York, 1987.

[30] D.J. Lehmann, *On primality tests*, SIAM J. Comput. **11** (1982), 374–375.

[31] G.L. Miller, *Riemann's hypothesis and tests for primality*, Conference Record of Seventh Annual ACM Symposium on Theory of Computation (Albuquerque, New Mexico), ACM, 5–7 May 1975, pp. 234–239.

[32] ———, *Riemann's hypothesis and tests for primality*, J. Comp. System Sci. **13** (1976), 300–317.

[33] François Morain, *Distributed primality proving and the primality of* $\left(2^{3539}+1\right)/3$, Research report 1152, INRIA, December 1989.

[34] ———, *Distributed primality proving and the primality of* $\left(2^{3539}+1\right)/3$, in Damgård [18], pp. 110–123.

[35] Richard G.E. Pinch, *The pseudoprimes up to* 10^{12}, Unpublished, 1992.

[36] ———, *The Carmichael numbers up to* 10^{15}, Math. Comp. **61** (1993), 381–391, Lehmer memorial issue.

[37] ———, *The pseudoprimes up to* 10^{13}, Preprint, 1994.

[38] ———, *The Carmichael numbers up to* 10^{16}, March 1998, `arXiv:math.NT/9803082`.

[39] C. Pomerance, J.L. Selfridge, and S.S. Wagstaff jr, *The pseudoprimes up to* 25.10^9, Math. Comp. **35** (1980), no. 151, 1003–1026.

[40] Carl Pomerance, *On the distribution of pseudoprimes*, Math. Comp. **37** (1981), 587–593.

[41] ———, *A new lower bound for the pseudoprime counting function*, Illinois J. Maths **26** (1982), 4–9.

[42] V.R. Pratt, *Every prime has a succinct certificate*, SIAM J. Comput. **4** (1975), 214–220.

[43] Michael O. Rabin, *Probabilistic algorithm for testing primality*, J. Number Theory **12** (1980), no. 1, 128–138.

[44] Paulo Ribenboim, *The book of prime number records*, Springer–Verlag, New York, 1988.

[45] ———, *The little book of big primes*, Springer–Verlag, New York, 1991.

[46] Hans Riesel, *Prime numbers and computer methods for factorization*, Progress in mathematics, vol. 57, Birkhauser, Boston, 1985.

[47] S. Wagon, *Mathematica in action*, W.H. Freeman, San Francisco, 1991.

[48] ———, *Number theory*, Mathematica Journal **2** (1992), 24–26.

[49] S. Wolfram, *Mathematica: a system for doing mathematics by computer*, Addison–Wesley, Redwood City, Ca, 1988.