Uncertainty can be Better than Certainty:

Some Algorithms for Primality Testing[*]

Richard P. Brent
Australian National University

[*] Presented at ANU, 3 March 2010.

UMS10

Abstract

For many years mathematicians and computer scientists have searched for a fast and reliable primality test. This is especially relevant nowadays, because the popular RSA public-key cryptosystem requires very large primes in order to generate secure keys. I will describe some efficient randomised algorithms that are useful, but have the defect of occasionally giving the wrong answer, or taking a very long time to give an answer.

In 2002, Agrawal, Kayal and Saxena (AKS) found a deterministic polynomial-time algorithm for primality testing. I will describe the original AKS algorithm and some improvements by Bernstein and Lenstra. As far as theory is concerned, we now know that "PRIMES is in P", and this appears to be the end of the story. However, I will explain why it is preferable to use randomised algorithms in practice.

First, some notation

As usual, we say that

$$f(n) = O(n^k)$$

if, for some c and n_0, for all $n \geq n_0$,

$$|f(n)| \leq cn^k \ .$$

We say that

$$f(n) = \widetilde{O}(n^k)$$

if, for all $\varepsilon > 0$,

$$f(n) = O(n^{k+\varepsilon}) \ .$$

The "$\widetilde{O}$" notation is useful to avoid terms like $\log n$ and $\log\log n$. For example, when referring to the Schönhage-Strassen algorithm for n-bit integer multiplication, it is easier to write

$$\widetilde{O}(n)$$

than the (more precise)

$$O(n \log n \log\log n) \ .$$

Complexity Classes (informal)

P is the class of decision ("yes"/"no") problems that have a *deterministic polynomial time algorithm*, i.e. an algorithm running in time $O(\lambda^k)$ on inputs of *length* λ, for some fixed k.

RP is the class of decision problems that have a randomised algorithm running in time $O(\lambda^k)$, with (one-sided) error probability at most 1/2 (if the *correct* answer is "yes").
co-RP is similar, but permitting an error on the other side (if the correct answer is "no").

BPP is similar but allows errors (with probability at most 1/4) on both sides.

ZPP is the class of decision problems that have an error-free randomised ("Las Vegas") algorithm running in *expected* time $O(\lambda^k)$.

NP is the class of decision problems for which, if the answer is "yes", there is a "certificate" that can be *verified* in time $O(\lambda^k)$.
co-NP is similar (for the answer "no").

Containment Relationships

$$P \subseteq ZPP = RP \cap \text{co-}RP \ ,$$

$$RP \cup \text{co-}RP \subseteq BPP \ ,$$

$$P \subseteq RP \subseteq NP \ ,$$

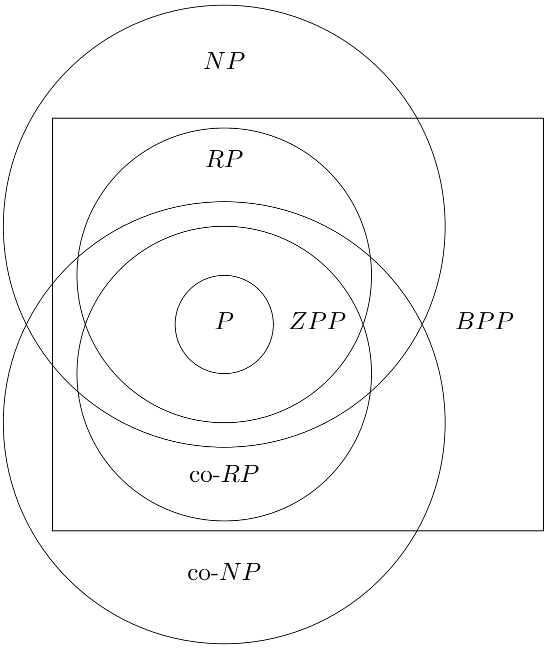
$$P \subseteq \text{co-}RP \subseteq \text{co-}NP \ .$$

It is not known if any of the inclusions is strict, or whether $BPP \subseteq NP$ or $NP \subseteq BPP$.

Note that $P \neq RP$ or $RP \neq NP$ implies

$$P \neq NP \ .$$

This inequality is one of the seven $1,000,000 *Millenium Prize Problems.*



Reducing Probability of Error

Given a problem in RP, co-RP, or BPP, we can reduce the probability of error below any given $\varepsilon > 0$ by iterating $O(\log(1/\varepsilon))$ times with independent random choices at each iteration. In this context an iteration is usually called a "trial".

For example, if an algorithm in RP has (one-sided) error probability at most 1/4, we can perform 10 independent trials to get an algorithm with error probability at most

$$\frac{1}{4^{10}} < \frac{1}{1000000} \ .$$

The Rabin-Miller primality-testing algorithm is an example of such an algorithm. We'll show later that it is actually an RP algorithm for testing *compositeness*, so if PRIMES is the problem of testing primality, we have:

$$\text{PRIMES} \in \text{co-}RP.$$

Factoring

Every positive integer has a unique factorisation into a product of prime powers. That is the main reason why primes are important: they are "building blocks" for the integers.

However, there does not seem to be any simple connection between algorithms for factoring and algorithms for testing primality. We'll see that there are algorithms that can answer the question

Is n prime?

much faster than any known algorithm for finding the prime factors of n (if it is composite).

The popular RSA cryptosystem needs two large primes p and q whose product $n = pq$ is difficult to factor. Primality testing algorithms are useful for generating p and q, but they are not useful for *cracking* RSA by factoring n (if p and q are kept secret).

Factoring Primes ?

Primality testing and factorisation are often confused, even by such luminaries as Bill Gates.

> *The obvious mathematical breakthrough would be development of an easy way to factor large prime numbers.*
>
> – Bill Gates *et al*, 1995[1]

Presumably Gates did not have in mind factorisations such as

$$13 = (2+3i)(2-3i)\ .$$

[1] Bill Gates, Nathan Myhrvold and Peter M. Rinearson, *The Road Ahead*, Viking Press, 1995, p. 265.

Fermat's Little Theorem

If n is prime and a is any integer, then

$$a^n = a \mod n.$$

Thus, if we find a such that $a^n \neq a \mod n$, we can be sure that n is composite. We say that

"a is a *witness* to the compositeness of n".

Note: we can guarantee that n is composite *without knowing the factors of* n.

The converse of Fermat's little theorem is false: if $a^n = a \mod n$ we can not be sure that n is prime. There are infinitely many examples (called *Carmichael numbers*) of composite n for which a^n is always $a \mod n$. The smallest example is

$$n = 561 = 3 \cdot 11 \cdot 17\ ,$$

$$\phi(n) = \mathrm{LCM}(2, 10, 16) = 80 \mid (n-1)\ .$$

The number of Carmichael numbers up to N is at least of order $N^{2/7}$ (Alford, Granville and Pomerance, 1994) so we can't ignore them!

Certificates

In the definition of NP we mentioned certificates. A *certificate* for a property is some information that enables a proof of the property to be completed in polynomial time.

For example, a "witness" a such that $a^n \neq a \mod n$ provides a certificate of the compositeness of n. A nontrivial factor of n would also provide a certificate.

Pratt (1975) showed that every prime p has a certificate of length $O((\log p)^2)$. The idea is to write

$$p - 1 = p_1^{\alpha_1} \cdots p_\nu^{\alpha_\nu}$$

and give a *primitive root* a of p. This can be verified by checking that

$$a^{p-1} = 1 \mod p$$

and

$$a^{(p-1)/p_\beta} \neq 1 \mod p \text{ for } \beta = 1, \ldots, \nu\ .$$

We (recursively) give certificates for $p_1, \ldots, p_\nu$ unless they are sufficiently small (say < 100).

Consequence of Pratt's Theorem

From Pratt's result,

$$\mathrm{PRIMES} \in NP.$$

Note that we don't claim that Pratt's certificate of primality of n is easy to find. Finding it is as difficult as factoring, since it requires the factorisation of $n-1$. The mere *existence* of the certificate is sufficient.
If n is composite, a nontrivial factor f of n (satisfying $1 < f < n$) provides a certificate of compositeness. Again, we don't claim that this certificate is easy to find! Its existence is enough to show that

$$\mathrm{PRIMES} \in \text{co-}NP.$$

Thus, we know that

$$\mathrm{PRIMES} \in NP \cap \text{co-}NP.$$

$NP \cap$ co-NP is believed to be smaller than NP, but a proof of this would imply that $P \neq NP$, so is likely to be difficult.

First Extension of Fermat

A slight extension of Fermat's little Theorem is useful for primality testing. The idea is simple. Suppose we are testing the primality of n, and we find x such that

$$x^2 = 1 \mod n \ .$$

If n is prime, then $x = \pm 1 \mod n$.

However, if n is composite, then it is possible (and quite likely) that $x \neq \pm 1 \mod n$. For example, consider $n = 21$ and $x = 8$.
More generally, if $n = p_1 p_2$, take
$x = -1 \mod p_1$, $x = +1 \mod p_2$.

When applying the Fermat test to n, we compute $a^{n-1} \mod n$ for some choice of a.
The Fermat test is passed if $a^{n-1} = 1 \mod n$.
If n is odd, then the exponent $n-1$ is even, and we can take x to be a suitable power of a. This gives the following primality test.

First Extension of Fermat cont.

If $n = 2^k q + 1$ is an odd prime, and $0 < a < n$, then either $a^q = 1 \mod n$, or the sequence

$$S = (a^q, a^{2q}, a^{4q}, \ldots, a^{2^k q}) \mod n$$

ends with 1, and the value just preceding the first appearance of 1 must be $-1 \mod n$.

That is, the sequence S looks like

$$\begin{array}{ll} (1, 1, \ldots 1) & \text{if } a^q = 1 \mod n, \\ (?, \ldots, ?, -1, 1, \ldots, 1) & \text{otherwise.} \end{array}$$

Proof: If $x^2 = 1 \mod n$ then $n|(x-1)(x+1)$.
Since n is prime, $n|(x-1)$ or $n|(x+1)$.
Thus $x = \pm 1 \mod n$. □

This fact has been known for a long time. Relevant names are Dubois, Selfridge, Artjuhov, and Miller. Rabin was the first to prove its usefulness in a randomised algorithm, usually called the *Rabin-Miller* algorithm.

The Rabin-Miller Algorithm

The extension of Fermat's little Theorem gives a *necessary* (but not sufficient) condition for primality of n. The Rabin-Miller algorithm checks if this condition is satisfied for a random choice of a, and returns "yes" if it is, "no" otherwise.

If n is prime, the answer is always "yes" (correct).

If n is composite, the answer could be "yes" (wrong) or "no" (correct), but it is "no" with probability greater than 3/4, i.e. the probability of error is $< 1/4$ (this is a theorem of Rabin).

Thus we have an RP-algorithm for testing compositeness. We can say that compositeness is in RP, or (equivalently)

$$\text{PRIMES} \in \text{co-}RP.$$

The Rabin-Miller algorithm gives a certificate for compositeness, but not a certificate for primality.

Popular Primality-Testing Algorithms

Popular algorithms include:

- The Rabin-Miller algorithm (1976).
- The *Jacobi Sums* algorithm of Adleman, Pomerance and Rumely (1983).
- The *Elliptic Curve Primality Proving* algorithm ECPP of Atkin and Morain (1993), based on a proposal by Goldwasser and Kilian.

These algorithms all have their good (and bad) points. We'll look at each more closely before discussing

- The AKS algorithm of Agrawal, Kayal and Saxena (2002).

It will be convenient to define

$$\lambda = \log n \ ,$$

where n is the number being tested for primality.

Jacobi Sums

The *Jacobi Sums* algorithm runs in time

$$\lambda^{O(\log\log\lambda)} \,.$$

This is *almost* polynomial time[2].

We can be more precise: Odlyzko and Pomerance have shown that, for all large n, the running time is in

$$[\lambda^{A\log\log\lambda},\ \lambda^{B\log\log\lambda}] \,,$$

where A, B are positive constants. The lower bound shows that the Jacobi Sums algorithms is definitely not polynomial-time (in theory anyway).

The Jacobi sums algorithm is deterministic and practical: it has been used for numbers of at least 3395 decimal digits (Mihailescu: 6.5 days on a 500 Mhz Dec Alpha).

[2] Recall that $\lambda = \log n$ so $\log\log\lambda = \log\log\log n$. While it has been *proved* that $\log\log\log n \to +\infty$ with n, it has never been *observed* doing so [Pomerance].

ECPP

The *Elliptic Curve Primality Proving* algorithm ECPP runs in *expected* polynomial time under some plausible assumptions, but the time bound has not been proved rigorously. With an improvement suggested by Shallit, the (conjectured) expected time is $\widetilde{O}(\lambda^4)$.

ECPP is a Las Vegas algorithm: the running time is random but the result is error-free.

ECPP is practical and has been used to prove primality of a number of 20562 decimal digits (Morain, 2006). It took the equivalent of 2219 days (about 6 years) on an AMS Opteron (2.39GHz), but actually only 9 months since the workload was distributed.

In practice ECPP is comparable to the Jacobi Sums algorithm, but ECPP has the advantage of producing an easily-checked certificate of primality. In fact, ECPP produces a certificate of size $O(\lambda^2)$ that can be checked in deterministic polynomial time $\widetilde{O}(\lambda^3)$.

Rabin-Miller

Rabin-Miller is a Monte Carlo algorithm: there is a nonzero probability of error. In practice the probability of error is negligible (less than 10^{-6}) if we take at least ten independent trials.

The algorithm is fast: one trial takes time

$$\widetilde{O}(\lambda^2)$$

(or $O(\lambda^3)$ with classical $O(\lambda^2)$ multiplication).

Rabin-Miller is feasible for numbers of 10^6 decimal digits. It produces a certificate of compositeness, but not a certificate of primality.

Combination Algorithm

Recall that ECPP produces a certificate of primality. Thus, using a combination of Rabin-Miller and ECPP, we can get a randomized algorithm that produces a certificate to prove that its result (whether "prime" or "composite") is correct.

All we have to do is run the Rabin-Miller and ECPP algorithms in "parallel" until one of them produces a certificate[3]. The expected running time is believed to be $\widetilde{O}(\lambda^4)$, although we can't *prove* this.

[3] That is, run both algorithms simultaneously using time-sharing.

Extension of Fermat to Polynomials

If n is prime and a is fixed, then

$$(x+a)^n = x^n + a \mod n \tag{1}$$

holds, where each side of the equality is a polynomial in x. Formally, we are working in the ring $(\mathbb{Z}/n\mathbb{Z})[x]$ of polynomials whose coefficients are in the ring $\mathbb{Z}/n\mathbb{Z}$ of integers mod n.

Agrawal and Biswas (1999) noticed that, provided $a \neq 0 \bmod n$, condition (1) is both necessary and sufficient for the primality of n.

In general, we can not compute $(x+a)^n \mod n$ in time polynomial in λ because it is a polynomial with $n+1$ terms.

The AKS Algorithm

In August 2002, Agrawal, Kayal and Saxena announced a *deterministic* polynomial-time primality test based on (1). Thus,

$$\text{PRIMES} \in P.$$

The idea is to compute $(x+a)^n \mod (x^r - 1, n)$ for $1 \le a \le s$ and sufficiently large r, s. The not-so-obvious fact is that it is sufficient to choose

$$r = O(\lambda^6), \;\; s = O(\lambda^4) \;.$$

Thus, we can do everything in time $\widetilde{O}(\lambda^k)$.

The precise value of the exponent k depends on details of the implementation. A revision of the AKS paper has $k = 7.5$.

The exponent k can be reduced to 6, and maybe even further, at the expense of more complicated algorithms or (worse) unproved assumptions.

Example

Take $n = 1729 = 7 \times 13 \times 19$. This is a Carmichael number, so

$$a^n = a \mod n$$

for all integers a. However,

$$(x+1)^n \neq x^n + 1 \mod n \;.$$

In fact, working mod n (i.e. in $(\mathbb{Z}/n\mathbb{Z})[x]$),

$$(x+1)^n = x^{1729} + 247x^{1722} + \cdots + 247x^7 + 1 \;.$$

We can more easily verify that $(x+1)^n \neq x^n + 1 \mod n$ by working mod $(x^5 - 1)$ as well as mod n: we find that

$$(x+1)^n = 134x^4 + 1330x^3 + 532x^2 + 1330x + 134$$

in $(\mathbb{Z}/n\mathbb{Z})[x]/(x^5 - 1)$.

Here $x^5 - 1$ acts rather like a hash function: it lets us sum every fifth term in the binomial expansion of $(x+1)^n$, thus reducing $n+1$ terms to five. The computation involves polynomials of degree at most eight.

The Key Theorem

Theorem (AKS-Bernstein-Morain, 2002)

Suppose that $n, r, s > 0$, where r is prime and q is the largest prime factor of $r - 1$. Suppose that

$$\binom{q+s-1}{s} > n^{2\lfloor\sqrt{r}\rfloor} \;, \tag{2}$$

that n has no prime factor $\le s$, and that $n^{(r-1)/q} \mod r \notin \{0, 1\}$. Finally, suppose that

$$(x-a)^n = x^n - a \tag{3}$$

in $(Z/nZ)[x]/(x^r - 1)$ for $1 \le a \le s$. Then n is a prime power.

Remarks

This formulation is given by Morain. In a primality test, after selecting r and s satisfying (2), it takes time $\widetilde{O}(rs\lambda^2)$ to check (3), so we want to minimise rs.

The original AKS algorithm has $r = O(\lambda^6)$, $q \ge 2s$, $s = O(\lambda\sqrt{r}) = O(\lambda^4)$, time $\widetilde{O}(\lambda^{12})$. The proof is "elementary", but too long to give in this lecture.

Reducing the Exponent

A *Sophie-Germain prime* is a prime q such that $r = 2q + 1$ is also prime, e.g. $q = 11$, $r = 23$.

It is *conjectured* that there are infinitely many Sophie-Germain primes, and that the number up to N is asymptotic to $2C_2N/(\ln N)^2$, where $C_2 \approx 0.66016$ is the Hardy-Littlewood twin-prime constant. Proving this conjecture is probably as difficult as proving the same conjecture for twin primes (in other words, difficult!).

If the Sophie-Germain conjecture is true, then $r = O(\lambda^2)$, $s = O(\sqrt{r}\lambda) = O(\lambda^2)$, and the time bound is reduced to $\widetilde{O}(rs\lambda^2) = \widetilde{O}(\lambda^6)$.

In fact, it's possible to get $\widetilde{O}(\lambda^6)$ without using the Sophie-Germain conjecture, but using a more sophisticated algorithm based on "Gaussian periods" (Lenstra and Pomerance).

Experimental Results

The following table gives some times for a Magma implementation of the AKS algorithm (with Lenstra & Bernstein's improvements) on a 1 Ghz Pentium.

Times marked "(est)" are estimated from the time taken for one of the s iterations, or by extrapolation, assuming the exponent $k = 6$.

n	r	s	time
$10^9 + 7$	43	315	1.0 sec
$10^{19} + 51$	67	5427	750 sec
$10^{49} + 9$	491	28801	32 hours
$10^{100} + 267$	3541	58820	1 year (est)
$2^{511} + 111$			13 years (est)
$2^{1023} + 1155$			840 years (est)

Table 1: The (improved) AKS algorithm

Comparison with Other Algorithms

The following table gives times for Magma implementations of the Rabin-Miller, ECPP and AKS algorithms on a 1 Ghz machine. In all cases the number tested was $10^{100} + 267$.

Algorithm	trials	time
Rabin-Miller	1	0.003 sec
Rabin-Miller	10	0.03 sec
Rabin-Miller	100	0.3 sec
ECPP		2.0 sec
AKS		1 year (est)

Table 2: Various algorithms

Testing 1024-bit Numbers

Numbers of about 1024 bits are of interest in cryptography. Testing numbers of this size on a 1GHz computer:

- Rabin-Miller takes less than one second.
- ECPP takes about 23 seconds.
- AKS would take more than 800 years!

Reliability of the Result

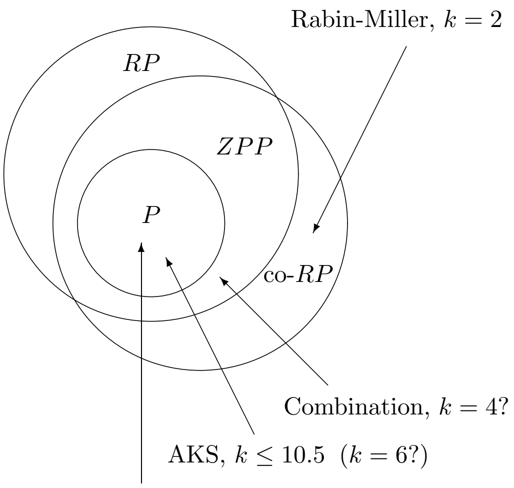
ECPP gives a certificate of primality, and the certificate can be checked quickly. It *should* be checked, to guard against hardware and/or programming errors. It is conjectured that the expected time to find a certificate is $\widetilde{O}(\lambda^4)$. The time to check a certificate is $\widetilde{O}(\lambda^3)$.

Rabin-Miller takes time $\widetilde{O}(\lambda^2)$ per trial. T trials give probability of error less than 4^{-T} if n is composite; there is no error if n is prime.

Theoretically, AKS is error-free. However, in a long computation there is a significant probability of a hardware error. Such an error would in most cases make a prime n "appear" composite; a composite n would usually still "appear" to be composite.

To be safe, one should make an independent check of the certificate (if available), or use at least two different algorithms.

Summary – Primality Testing and Complexity Classes



Rabin-Miller, $k = 2$

RP

ZPP

P

co-RP

Combination, $k = 4$?

AKS, $k \leq 10.5$ ($k = 6$?)

Lenstra-Pomerance, $k = 6$

(Conjectured) running time is $\widetilde{O}(\lambda^k)$.

Conclusions

- The AKS algorithm is theoretically significant, since it shows that PRIMES $\in P$.
- AKS is not a practical algorithm. ECPP is *much* faster. Rabin-Miller is even faster, at the price of a *minute* probability of error. A combination of Rabin-Miller and ECPP avoids this possible error, and provides an easily-checked certificate.
- The assumption that problems with polynomial-time algorithms are feasible, and other problems are intractible, is too simplistic.

  In practice, "expected" or "conjectured" or "almost" polynomial algorithms can be better than deterministic polynomial-time algorithms. The exponent k is important.

Last Words

A crude application of theory says that the AKS algorithm is best, but a realistic analysis shows that Rabin-Miller and/or ECPP are much to be preferred. In general, Monte Carlo or Las Vegas algorithms may be better than deterministic polynomial-time algorithms!

I would be more confident in the security of a cryptosystem using large primes certified by Rabin-Miller than (necessarily smaller) primes certified by AKS.

References

[1] L. M. Adleman, C. Pomerance and R. Rumely, On distinguishing prime numbers from composite numbers, *Ann. of Math.* 117 (1983), 173–206.

[2] M. Agrawal and S. Biswas, Primality and identity testing via Chinese remaindering, *Proc. IEEE Symposium on Foundations of Computer Science*, 1999, 202–209.

[3] M. Agrawal, N. Kayal and N. Saxena, PRIMES is in P, *Annals of Mathematics* 160(2): 781-793, 2004.

[4] W. Alford, A. Granville and C. Pomerance, There are infinitely many Carmichael numbers, *Ann. of Math.* 139 (1994), 703–722.

[5] A. Atkin and F. Morain, Elliptic curves and primality proving, *Math. Comp.* 61 (1993), 29–68.

[6] D. Bernstein, Distinguishing prime numbers from composite numbers, `http://cr.yp.to/primetests.html`.

[7] R. Crandall and C. Pomerance, *Prime Numbers: A Computational Perspective*, Springer-Verlag, New York, 2001.

[8] S. Goldwasser and J. Kilian, Primality testing using elliptic curves, *J. ACM* 46 (1999), 450–472. Preliminary version *STOC*, 1986, 316–329.

[9] D. E. Knuth, *The Art of Computer Programming*, Vol. 2, 3rd edition, Addison-Wesley, Menlo Park, 1997, §4.5.4.

[10] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.

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

[12] M. O. Rabin, Probabilistic algorithms for testing primality, *J. Number Theory* 12 (1980), 128–138.

[13] R. L. Rivest, A. Shamir and L. Adleman, A method for obtaining digital dignatures and public-key cryptosystems, *Comm. ACM* 21 (1978), 120–126.