An acceleration of the AKS prime identification algorithm

Stephen Meredith Williams
email zen323167@zen.co.uk
Barcombe Services
3a Priory Court
Nettleton
CHIPPENHAM
Wiltshire SN14 7NY
UK

January 25, 2024

Abstract

In its standard form, the AKS prime identification algorithm is deterministic and polynomial time but too slow to be of practical use. By dropping its deterministic attribute, it can be accelerated to an extent that it is practically useful, though still much slower than the widely used Miller-Rabin-Selfridge-Monier (MRSM) algorithm based on the Fermat Little Theorem or the Solovay-Strassen algorithm based on the Euler Criterion. The change made, in the last stage of AKS, is to check a modular equation not for a long sequence of values but for a single one. Another change is to reduce, arbitrarily, the size of the parameter r giving the degree of the polynomial used for the last stage.[1][2]

[1]**Key Words:** AKS primality testing, single–value, Carmichael numbers, binomial theorem, acceleration, algorithm

[2]**MathSciNet Classification** 11Y11

0.1 Report

In 2004, Manindra Agrawal, Neeraj Kayal and Nitin Saxena described [1] a general, unconditional, deterministic, polynomial-time algorithm that identifies an input number as prime or composite. After their surname initials it is generally known as AKS. It was seen as a breakthrough, but the run-time polynomial in question was of degree 12, so high that the algorithm did not supplant existing methods of identifying primality that are probabilistic or conditional (under the Extended Riemann Hypothesis). These methods matter particularly when it comes to cryptographic systems such as RSA, whether in constructing large semi-primes to implement such a code or in attempts by way of factoring the semi-prime to attack it.

AKS depends upon the binomial theorem:

$$(x+y)^n = \sum_{i=0}^{n} \binom{n}{i} x^{n-i} y^i.$$

Since $\binom{n}{i} = \frac{n!}{i!(n-i)!}$ and $m! = m(m-1)(m-2)\ldots 3.2.1$ it can be seen that when n is prime every coefficient of the expansion except the first and the last ones is divisible by n. In other words,

$$(x+y)^p \equiv x^p + y^p \bmod p$$

for prime p. We also have by the Fermat Little Theorem that if n is prime

$$y^n \equiv y \bmod n$$

and so

$$(x+a)^n \equiv x^n + a \bmod n \tag{1}$$

(1) can form the basis of a valid but slow primality test, especially as many computer languages will generate the binomial coefficients for you. In GP-PARI the short function: **for(i=1,n-1,if(binomial(n,i)%n,return(0)));1;** implements this idea easily and successfully, but for quite small n there are problems with the stack size.

What is special about AKS, is that, to get around the problem of computing the high-degree polynomial $(x+y)^n$, one computes the polynomial modulo some small-degree polynomial. This is additional to computing it modulo n as well. A simple polynomial of degree r is $x^r - 1$.

In AKS, the double modular equality

$$(x+a)^n \equiv x^n + a \bmod (n, x^r - 1) \tag{2}$$

is checked for a long sequence of values of a, starting with 1 and finishing with $\sqrt{\phi(r)}\log(n)$ where ϕ is the Euler phi symbol, whose value for prime r is $r-1$. The acceleration being proposed is achieved largely by replacing

this sequence with any single value of a. Normally, in experimentation with this idea, $a = 2$ was used.[3] *Very surprisingly, this has never been found to give the wrong answer with a single proviso to be added.* The answers are all checked, e.g., by whether the number is in a file of primes, or else by probabilistic algorithm MRSM, which uses a converse of the Fermat Little Theorem on a number of "witnesses" to primality.

The proviso mentioned above occurs earlier in the algorithm. In (2) a value r appears which is the degree of the modular polynomial. This value is specified at the beginning of AKS (once it has been checked that n is not a perfect power). The proviso is that

$$n^2 \not\equiv 1 \bmod r \tag{3}$$

AKS stipulates that

$$\mathrm{o}_r(n) > \log^2 n \tag{4}$$

that is, that the order of n modulo r should be greater than the square of log n. This is part of the proof of the algorithm as deterministic. It produces large values of r, but the algorithm will in practice run and terminate with many different and smaller values of r. Since the run-time is proportional to $r^{3/2}$ [5] (p. 16), small values of r give a shorter run-time. Taking this to the extreme, the smallest value of r that works is 5, however that contravenes (3) very often. If it does, this implementation then tries $r = 29$. If that contravenes (3) then $r = 37$, is tried, then 41, 53, 59, 67, 71, 83, 89, 101, 103, 127, 131 will be tried if necessary. These values are those that appear often in attempts to find r under (4) (using the algorithm given by [5], p. 16, section 3b.3). If none of them were to work, the implementation would fall back on its initialization as $r = 1$, which would lead to a panic and stop the program (something that has never yet happened).

For a test prime of 117 decimal digits (388 binary digits) namely $(12^{109} + 1)/13$ the **run-time** on an i7 desktop system was 432 seconds for the full sequence, but only half a second on the same system with 2 as the only a. For a test composite of 121 decimal digits (402 binary digits) namely $(12^{113} + 1)/13$ the run-time was again half a second with 2 as the only a but now the same with the full sequence programmed as the loop exits at the first double modular equality (2) to be checked. All these times are much slower than MRSM (the approach based on witnesses to primality) which only takes 1ms for both of these numbers of about 400 bits each. Benchmarking for some Mersenne primes confirms this general picture.

Naturally, the utility of the claim that the algorithm works with a single value of a depends on the extent to which the claim's **correctness** has been

[3] The only known exception is $a = 1$, which is also *quite* effective as a single value, but fails seven times within the first million, for $n = 473$, 6443, 9701, 10153, 35621, 137419, and 896203. Though there were no further failures for $10^6 < n \leq 10^7$, seven more were found for $10^7 < n < 10^8$.

tested. The following numbers have been checked with an implementation in Rust[4] and the outcome confirmed mainly by MRSM:[5]

- natural numbers to 10^8
- prime numbers to 10^9, which amounts to 50847534 numbers
- some particularly challenging composites, i.e. strong pseudo-primes to low base [9, 7]: 113 such numbers were all confirmed composite
- a hundred million randomly chosen natural numbers between 10^8 and 10^{10}
- a hundred thousand randomly chosen natural numbers between $10^{200}+10^8$ and $10^{200}+10^{10}$
- for everyday prime identification e.g. in the course of factoring, single–value AKS is routinely used by the author as a backup to MRSM.

The algorithm as modified follows. It needs to be emphasized that there is no *guarantee* that the decision as between prime and composite is correct. It is merely that this algorithm has never been found to give a false result. MRSM is probabilistic and will give the wrong answer very occasionally, but with 25 witnesses so rarely that it can be taken as correct. If there ever were to be a contradiction, it would have to be resolved by another algorithm, such as that of Solovay-Strassen based on the Euler criterion or that of Lucas.

Input: integer $n > 131$
1. Check that n is not a perfect power
2. The set $\{5, 29, 37, 41, 53, 59, 67, 71, 83, 89, 101, 103, 127, 131\}$ is used to choose r. Starting from the left, the first value in the set that obeys (3) is used.
3. If $1 < (a, n) < n$ for some $a < r$, output COMPOSITE.
4. If

$$(x+2)^n \equiv x^n + 2 \bmod (n, x^r - 1)$$

output PRIME else output COMPOSITE

Algorithm for Primality Testing

The sub-algorithms needed for the checking of stage 4 are given in books like [11] (algorithms 3.44 and 3.45) and the treatments in [12] and [4] (algorithm 8.2.1) and [3] (algorithms 4.5.1, or 4.5.9 for D. J. Bernstein's quartic time variant) are also recommended. The single–value algorithm is able to

[4] using the rug package when arbitrary-precision arithmetic was needed
[5] Another implementation that was in Rust, using 25 witnesses.

demonstrate the primality of repunit-1031 in 12 seconds and Mersenne-2203 (which is 969 decimal digits) in about half a second. As well, repunit-1021 is shown composite in about 12 seconds too.

An issue requiring further research is why (3) is necessary. It is a fact that if it is omitted, there are failures of the algorithm presented above. Stipulation (3) was included in experimentation because it is part of a conjecture sometimes attributed to Agrawal.[10] This would have the effect of reducing the degree of the polynomial for the AKS run-time, originally 12, which is why it is so impractical, to 3. A modified form of the conjecture is endorsed by [10] (Conclusion) but is more elaborate than the algorithm above, namely, that a second value $a = -1$ should be added to $a = 2$ for the algorithm to be valid. As stated previously, the present work suggested that *any* single value of a, not just 2, will do. However, the algorithms presented by [11] will not permit a negative a.

0.2 Afterword

As well as checking stage 4 for indeterminate x with these sub-algorithms like [11] (algorithms 3.44, 3.45), it is also possible, *and much faster,* to omit r and check the congruence (1) for $a = 2$ and specific values of x. Experimentation led to prime values of $x \in \{5, 7, 11, \ldots, 89, 97\}$ being used. Intervening composites or many more primes as x did not affect the results. These were that the only values of n misidentified were Carmichael numbers (and that all of those were misclassified as prime). There are 646 of these below 10^9 [6, 9] but, indeed, an infinite number in all [2]. [8] gives much more on the distribution of Carmichael numbers, including the database work of R. G. E. Pinch. A test for primality could check that a number classified as prime on this basis was not a Carmichael number, since these are so rare, but such a test would have no advantage over tests based on a converse of the Fermat Little Theorem.

Bibliography

[1] Agrawal, M., Kayal, N., and Saxena, N. Primes is in P. *The Annals of Mathematics 160* (2004), 781–793.

[2] Alford, W., Granville, A., and Pomerance, C. There are infinitely many Carmichael numbers. *The Annals of Mathematics 139* (1994), 703–722.

[3] Crandall, R., and Pomerance, C. B. *Prime numbers: a computational perspective*, vol. 182. Springer Science & Business Media, 2006.

[4] Dietzfelbinger, M. *Primality Testing in Polynomial Time: From Randomized Algorithms to "Primes is in P"*. Springer, 2004.

[5] Granville, A. It is easy to determine whether a given integer is prime. *Bulletin of the American Mathematical Society 42* (2004), 3–38.

[6] Jaeschke, G. The Carmichael numbers to 10^{12}. *Mathematics of Computation 55* (1990), 383–389.

[7] Jaeschke, G. On strong pseudoprimes to several bases. *Mathematics of Computation 61* (1993), 915–926.

[8] Jameson, G. Finding Carmichael numbers. *The Mathematical Gazette 95* (2011), 244–255.

[9] Pomerance, C., Selfridge, J. L., and Wagstaff, S. On the distribution of pseudoprimes to 25.10^9. *Mathematics of Computation 37* (1980), 587–593.

[10] Popovych, R. A note on Agrawal conjecture. Tech. rep., Cryptolology e-print archive, 2009.

[11] Wagstaff, S. S. *The joy of factoring*, vol. 68. American Mathematical Soc., 2013.

[12] Yan, S. Y. *Primality Testing and Integer Factorization in Public-Key Cryptography*. Kluwer, 2004.