AKS primality test

From Wikipedia, the free encyclopedia

The **AKS primality test** (also known as **Agrawal–Kayal–Saxena primality test** and **cyclotomic AKS test**) is a deterministic primality-proving algorithm created and published by Manindra Agrawal, Neeraj Kayal, and Nitin Saxena, computer scientists at the Indian Institute of Technology Kanpur, on August 6, 2002, in a paper titled "PRIMES is in P".[1] The algorithm determines whether a number is prime or composite within polynomial time. The authors received the 2006 Gödel Prize and the 2006 Fulkerson Prize for this work.

Contents [hide]

Importance [edit]

AKS is the first primality-proving algorithm to be simultaneously *general*, *polynomial*, *deterministic*, and *unconditional*. Previous algorithms had been developed for centuries and achieved three of these properties at most, but not all four.

- The AKS algorithm can be used to verify the primality of any **general** number given. Many fast primality tests are known that work only for numbers with certain properties. For example, the Lucas–Lehmer test works only for Mersenne numbers, while Pépin's test can be applied to Fermat numbers only.
- The maximum running time of the algorithm can be expressed as a **polynomial** over the number of digits in the target number. ECPP and APR conclusively prove or disprove that a given number is prime, but are not known to have polynomial time bounds for all inputs.
- The algorithm is guaranteed to distinguish **deterministically** whether the target number is prime or composite. Randomized tests, such as Miller–Rabin and Baillie–PSW, can test any given number for primality in polynomial time, but are known to produce only a probabilistic result.
- The correctness of AKS is **not conditional** on any subsidiary unproven hypothesis. In contrast, the Miller test is fully deterministic and runs in polynomial time over all inputs, but its correctness depends on the truth of the yet-unproven generalized Riemann hypothesis.

While the algorithm is of immense theoretical importance, it is not used in practice. For 64-bit inputs, the Baillie–PSW is deterministic and runs many orders of magnitude faster. For larger inputs, the performance of the (also unconditionally correct) ECPP and APR tests is *far* superior to AKS. Additionally, ECPP can output a primality certificate that allows independent and rapid verification of the results, which is not possible with the AKS algorithm.

Concepts [edit]

The AKS primality test is based upon the following theorem: An integer n (≥ 2) is prime if and only if the polynomial congruence relation

$$(x-a)^n \equiv (x^n - a) \pmod{n} \qquad (1)$$

holds for all integers a coprime to n (or even just for some such integer a, in particular for $a = 1$).[1] Note that x is a free variable. It is never substituted by a number; instead you have to expand $(x-a)^n$ and compare the coefficients of the x powers.

This theorem is a generalization to polynomials of Fermat's little theorem, and can easily be proven using the binomial theorem together with the following property of the binomial coefficient:

$\binom{n}{k} \equiv 0 \pmod{n}$ for all $0 < k < n$ if and only if *n* is prime.

While the relation (1) constitutes a primality test in itself, verifying it takes exponential time. Therefore, to reduce the computational complexity, AKS makes use of the related congruence

$$(x-a)^n \equiv (x^n - a) \pmod{(n, x^r - 1)} \qquad (2)$$

which is the same as:

$$(x-a)^n - (x^n - a) = nf + (x^r - 1)g \qquad (3)$$

for some polynomials *f* and *g*. This congruence can be checked in polynomial time with respect to the number of digits in n, because it is provable that r need only be logarithmic with respect to n. Note that all primes satisfy this relation (choosing *g* = 0 in (3) gives (1), which holds for *n* prime). However, some composite numbers also satisfy the relation. The proof of correctness for AKS consists of showing that there exists a suitably small *r* and suitably small set of integers *A* such that, if the congruence holds for all such *a* in *A*, then *n* must be prime.

History and running time [edit]

In the first version of the above-cited paper, the authors proved the asymptotic time complexity of the algorithm to be $\tilde{O}(\log^{12}(n))$ (using Õ from big O notation). In other words, the algorithm takes less time than the twelfth power of the number of digits in *n* times a polylogarithmic (in the number of digits) factor. However, the upper bound proved in the paper was rather loose; indeed, a widely held conjecture about the distribution of the Sophie Germain primes would, if true, immediately cut the worst case down to $\tilde{O}(\log^6(n))$.

In the months following the discovery, new variants appeared (Lenstra 2002, Pomerance 2002, Berrizbeitia 2003, Cheng 2003, Bernstein 2003a/b, Lenstra and Pomerance 2003), which improved the speed of computation by orders of magnitude. Due to the existence of the many variants, Crandall and Papadopoulos refer to the "AKS-class" of algorithms in their scientific paper "On the implementation of AKS-class primality tests", published in March 2003.

In response to some of these variants, and to other feedback, the paper "PRIMES is in P" was updated with a new formulation of the AKS algorithm and of its proof of correctness. (This version was eventually published in *Annals of Mathematics*.) While the basic idea remained the same, *r* was chosen in a new manner, and the proof of correctness was more coherently organized. While the previous proof had relied on many different methods, the new version relied almost exclusively on the behavior of cyclotomic polynomials over finite fields. The new version also allowed for an improved bound on the time complexity, which can now be shown by simple methods to be $\tilde{O}(\log^{10.5}(n))$. Using additional results from sieve theory, this can be further reduced to $\tilde{O}(\log^{7.5}(n))$.

In 2005, Carl Pomerance and H. W. Lenstra, Jr. demonstrated a variant of AKS that runs in $\tilde{O}(\log^6(n))$ operations, where *n* is the number to be tested – a marked improvement over the initial $\tilde{O}(\log^{12}(n))$ bound in the original algorithm.[2] An updated version of the paper is also available.[3]

Agrawal, Kayal and Saxena suggest a variant of their algorithm which would run in $\tilde{O}(\log^3(n))$ if Agrawal's conjecture is true; however, a heuristic argument by Hendrik Lenstra and Carl Pomerance suggests that it is probably false.[1]

Algorithm [edit]

The algorithm is as follows:[1]

Input: integer *n* > 1.

1. If $n = a^b$ for integers *a* > 1 and *b* > 1, output *composite*.
2. Find the smallest *r* such that $O_r(n) > (\log_2 n)^2$.
3. If 1 < gcd(*a*,*n*) < *n* for some *a* ≤ *r*, output *composite*.
4. If *n* ≤ *r*, output *prime*.
5. For *a* = 1 to $\lfloor \sqrt{\varphi(r)} \log_2(n) \rfloor$ do

   if $(X+a)^n \neq X^n + a \pmod{X^r - 1, n}$, output *composite*;
6. Output *prime*.

Here $O_r(n)$ is the multiplicative order of *n* modulo *r*, log_2 is the binary logarithm, and $\varphi(r)$ is Euler's totient function of *r*.

If n is a prime number, the algorithm will always return *prime*: since n is prime, steps 1 and 3 will never return *composite*. Step 5 will also never return *composite*, because (2) is true for all prime numbers n. Therefore, the algorithm will return *prime* either in step 4 or in step 6.

Conversely, if n is composite, the algorithm will always return *composite*: if the algorithm returns *prime*, then this will occur in either step 4 or step 6. In the first case, since $n \le r$, n has a factor $a \le r$ such that $1 < \gcd(a,n) < n$, which will return *composite*. The remaining possibility is that the algorithm returns *prime* in step 6. The authors' article[1] proves that this will not happen because the multiple congruences tested in step 5 are sufficient to guarantee that the output is *composite*.

Example 1: n = 31 is Prime [edit]

Input: integer $n = 31 > 1$.

1. If $n = a^b$ for integers $a > 1$ and $b > 1$, output *composite*.

   For [b=2, b<$\log_2$(n), b++,
     a=$n^{1/b}$;
     If [a is integer, Return

   a=$n^{1/2}$...$n^{1/4}$={5.568, 3.141, 2.360}

2. Find the smallest r such that $O_r(n) > (\log n)^2$.

   maxk=$\lfloor(\log_2 n)^2\rfloor$;
   maxr=Max[3, $\lceil(\text{Log}_2\, n)^5\rceil$]; (*maxr really isn't needed*)
   r=2;
   nextR=True;
   For [r=2, nextR && r < maxr, r++,
     nextR=False;
     For [k=1,(!nextR) &&k ≤ maxk, k++,
       nextR=(Mod[n^k, r]==1 || Mod[n^k, r]==0)
     ]
   ];
   r--; (*the loop over increments by one*)

   r = 29

3. If $1 <$ gcd$(a,n) < n$ for some $a \le r$, output *composite*.

   For [a=r, a > 1, a--,
     If [(gcd=GCD[a,n]) > 1 && gcd < n, Return[Composite]]
   ];

   gcd={GCD(29,31)=1, GCD(28,31)=1, ..., GCD(2,31)=1} $\ngtr$ 1

4. If $n \le r$, output *prime*.

   If [n ≤ r, Return[Prime]]; (* this step may be omitted if n > 5690034 *)

   31 > 29

5. For $a = 1$ to $\lfloor\sqrt{\varphi(r)}\log(n)\rfloor$ do

   if $(X+a)^n \neq X^n+a \pmod{X^r - 1, n}$, output *composite*;

   φ[x_]:=EulerPhi[x];
   PolyModulo[f_]:=PolynomialMod[PolynomialRemainder[f,x^r-1,x],n];
   max=Floor[Log[2,n]√φr];
   For[a=1, a ≤ max, a++,
     If[PolyModulo[$(x+a)^n$]-PolynomialRemainder[x^n+a, x^r-1, x]≠0,
       Return[Composite]
     ]
   ];

   $(x+a)^{31}$ =

$a^{31} + 31a^{30}x + 465a^{29}x^2 + 4495a^{28}x^3 + 31465a^{27}x^4 + 169911a^{26}x^5 + 736281a^{25}x^6 + 2629575a^{24}x^7 + 7888725a^{23}x^8 + 20160075a^{22}x^9 + 44352165a^{21}x^{10} + 84672315a^{20}x^{11} + 141120525a^{19}x^{12} + 206253075a^{18}x^{13} + 265182525a^{17}x^{14} + 300540195a^{16}x^{15} + 300540195a^{15}x^{16} + 265182525a^{14}x^{17} + 206253075a^{13}x^{18} + 141120525a^{12}x^{19} + 84672315a^{11}x^{20} + 44352165a^{10}x^{21} + 20160075a^9x^{22} + 7888725a^8x^{23} + 2629575a^7x^{24} + 736281a^6x^{25} + 169911a^5x^{26} + 31465a^4x^{27} + 4495a^3x^{28} + 465a^2x^{29} + 31ax^{30} + x^{31}$

PolynomialRemainder $[(x+a)^{31}, x^{29}-1] =$

$465a^2 + a^{31} + (31a+31a^{30})x + (1+465a^{29})x^2 + 4495a^{28}x^3 + 31465a^{27}x^4 + 169911a^{26}x^5 + 736281a^{25}x^6 + 2629575a^{24}x^7 + 7888725a^{23}x^8 + 20160075a^{22}x^9 + 44352165a^{21}x^{10} + 84672315a^{20}x^{11} + 141120525a^{19}x^{12} + 206253075a^{18}x^{13} + 265182525a^{17}x^{14} + 300540195a^{16}x^{15} + 300540195a^{15}x^{16} + 265182525a^{14}x^{17} + 206253075a^{13}x^{18} + 141120525a^{12}x^{19} + 84672315a^{11}x^{20} + 44352165a^{10}x^{21} + 20160075a^9x^{22} + 7888725a^8x^{23} + 2629575a^7x^{24} + 736281a^6x^{25} + 169911a^5x^{26} + 31465a^4x^{27} + 4495a^3x^{28}$

A) PolynomialMod [PolynomialRemainder $[(x+a)^{31}, x^{29}-1], 31] = a^{31}+x^2$

B) PolynomialRemainder $[x^{31}+a, x^{29}-1] = a+x^2$

A) - B) $= a^{31}+x^2 - (a+x^2) = a^{31}-a$

$\max = \lfloor \log_2(31)\sqrt{\varphi(29)} \rfloor = 26$

$\{1^{31}-1=0 \pmod{31}, 2^{31}-2=0 \pmod{31}, 3^{31}-3=0 \pmod{31}, ..., 26^{31}-26=0 \pmod{31}\}$

6. Output *prime*.

   31 Must be Prime

Where PolynomialMod is a term-wise modulo reduction of the polynomial. e.g. PolynomialMod$[x+2x^2+3x^3, 3] = x+2x^2+0x^3$

References [edit]

1. ^ *a b c d e* Agrawal, Manindra; Kayal, Neeraj; Saxena, Nitin (2004). "PRIMES is in P" (PDF). *Annals of Mathematics* **160** (2): 781–793. doi:10.4007/annals.2004.160.781. JSTOR 3597229.
2. ^ H. W. Lenstra Jr. and Carl Pomerance, "Primality testing with Gaussian periods", preliminary version July 20, 2005.
3. ^ H. W. Lenstra jr. and Carl Pomerance, "Primality testing with Gaussian periods", version of April 12, 2011.

Further reading [edit]

- Dietzfelbinger, Martin (2004). *Primality testing in polynomial time. From randomized algorithms to ``PRIMES is in P*. Lecture Notes in Computer Science **3000**. Berlin: Springer-Verlag. ISBN 3-540-40344-2. Zbl 1058.11070.

External links [edit]

- Weisstein, Eric W., "AKS Primality Test", *MathWorld*.
- R. Crandall, Apple ACG, and J. Papadopoulos (March 18, 2003): On the implementation of AKS-class primality tests (PDF)
- Article by Borneman, containing photos and information about the three Indian scientists (PDF)
- Andrew Granville: It is easy to determine whether a given integer is prime
- The Prime Facts: From Euclid to AKS, by Scott Aaronson (PDF)
- The PRIMES is in P little FAQ by Anton Stiglic
- 2006 Gödel Prize Citation
- 2006 Fulkerson Prize Citation
- The AKS "PRIMES in P" Algorithm Resource
- Grime, Dr. James. "Fool-Proof Test for Primes - Numberphile" (video). Brady Haran. [the video describes the exponential time relation (1), which it calls AKS]

Primality tests	**AKS test** · APR test · Baillie–PSW · ECPP test · Elliptic curve · Pocklington · Fermat · Lucas · LUCAS–LEHMER · LUCAS–LEHMER–RIESEL · *PROTH'S THEOREM* · *PÉPIN'S* · Quadratic Frobenius test · Solovay–Strassen · Miller–Rabin
Prime-generating	Sieve of Atkin · Sieve of Eratosthenes · Sieve of Sundaram · Wheel factorization
Integer factorization	Continued fraction (CFRAC) · Dixon's · Lenstra elliptic curve (ECM) · Euler's · Pollard's rho · $p - 1$ · $p + 1$ · Quadratic sieve (QS) · General number field sieve (GNFS) · *Special number field sieve (SNFS)* · Rational sieve · Fermat's · Shanks' square forms · Trial division · Shor's
Multiplication	Ancient Egyptian · Long · Karatsuba · Toom–Cook · Schönhage–Strassen · Fürer's
Discrete logarithm	Baby-step giant-step · Pollard rho · Pollard kangaroo · Pohlig–Hellman · Index calculus · Function field sieve
Greatest common divisor	Binary · Euclidean · Extended Euclidean · Lehmer's
Modular square root	Cipolla · Pocklington's · Tonelli–Shanks
Other algorithms	Chakravala · Cornacchia · Integer relation · Integer square root · Modular exponentiation · Schoof's
Italics indicate that algorithm is for numbers of special forms · Smallcaps indicate a deterministic algorithm	

Categories: Primality tests | Finite fields