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 was the first to determine whether any given number is prime or composite within polynomial time. The authors received the 2006 Gödel Prize and the 2006 Fulkerson Prize for this work.

Contents

Importance

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, Miller's version of the Miller-Rabin 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 primality test 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

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 some a coprime to n.[1] Note that x should be understood as a formal symbol.

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:

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

While the relation (**1**) constitutes a primality test in itself, verifying it takes exponential time: the brute force approach would require the expansion of the $(x - a)^n$ polynomial and a reduction (mod n) of the resulting $n + 1$ coefficients.

The congruence is an equality in the polynomial ring $\mathbb{Z}_n[x]$. Evaluating in a quotient ring of $\mathbb{Z}_n[x]$ creates an upper bound for the degree of the polynomials involved. The AKS evaluates the equality in $\mathbb{Z}_n[x]/(x^r - 1)$, making the computational complexity dependent on the size of r. For clarity,[1] this is expressed as the congruence

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

which is the same as:

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

for some polynomials f and g.

Note that all primes satisfy this relation (choosing $g = 0$ in (**3**) gives (**1**), which holds for n prime). This congruence can be checked in polynomial time when r is polynomial to the digits of n. The AKS algorithm evaluates this congruence for a large set of a values, whose size is polynomial to the digits of n. The proof of validity of the AKS algorithm shows that one can find an r and a set of a values with the above properties such that if the congruences hold then n is a power of a prime.[1]

History and running time

In the first version of the above-cited paper, the authors proved the asymptotic time complexity of the algorithm to be $\tilde{O}(\log(n)^{12})$ (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 $\tilde{O}(\log(n)^6)$.

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. Owing 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 $\tilde{O}(\log(n)^{10.5})$. Using additional results from sieve theory, this can be further reduced to $\tilde{O}(\log(n)^{7.5})$.

In 2005, Carl Pomerance and H. W. Lenstra, Jr. demonstrated a variant of AKS that runs in $\tilde{O}(\log(n)^{6})$ operations, where n is the number to be tested – a marked improvement over the initial $\tilde{O}(\log(n)^{12})$ bound in the first version of the AKS paper.[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 $\tilde{O}(\log(n)^{3})$ if Agrawal's conjecture is true; however, a heuristic argument by Hendrik Lenstra and Carl Pomerance suggests that it is probably false.[1]

The algorithm

The algorithm is as follows:[1]

Input: integer $n > 1$.

1. Check if n is a perfect power: if $n = a^b$ for integers $a > 1$ and $b > 1$, output *composite*.
2. Find the smallest r such that $\mathrm{ord}_r(n) > (\log_2 n)^2$. (if r and n are not coprime, then skip this r)
3. For all $2 \le a \le \min(r, n-1)$, check that a does not divide n: If $a|n$ for some $2 \le a \le \min(r, n-1)$, output *composite*.
4. If $n \le r$, output *prime*.
5. For $a = 1$ to $\left\lfloor \sqrt{\varphi(r)} \log_2(n) \right\rfloor$ do

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

6. Output *prime*.

Here $\mathrm{ord}_r(n)$ is the multiplicative order of n modulo r, $\log_2$ is the binary logarithm, and $\varphi(r)$ is Euler's totient function of r.

Step 3 is shown in the paper as checking $1 < (a,n) < n$ for all $a \le r$. It can be seen this is equivalent to trial division up to r, which can be done very efficiently without using gcd. Similarly the comparison in step 4 can be replaced by having the trial division return *prime* once it has checked all values up to and including $\lfloor \sqrt{n} \rfloor$.

Once beyond very small inputs, step 5 dominates the time taken. Most improvements made to the algorithm have concentrated on reducing the size of r which makes the core operation in step 5 faster, and in reducing the size of s, the number of loops performed in step 5.[4] Typically these changes do not change the computational complexity, but can lead to many orders of magnitude less time taken, e.g. Bernstein's final version has a theoretical speedup by a factor of over 2 million.

Proof of validity outline

For the algorithm to be correct, all steps that identify n must be correct. Steps 1, 3 and 4 are trivially correct, since they are based on direct tests of the divisibility of n. Step 5 is also correct: since (2) is true for any choice of a coprime to n and r if n is prime, an inequality means that n must be composite.

The difficult case of the algorithm is step 6. Its proof of correctness is based on the upper and lower bounds of a multiplicative group in $\mathbb{Z}_n[x]$ constructed from the $(X + a)$ binomials that are tested in step 5. Step 4 guarantees that these binomials are $\left\lfloor \sqrt{\varphi(r)} \log_2(n) \right\rfloor$ distinct elements of $\mathbb{Z}_n[x]$. For the particular choice of r, the bounds produce a contradiction unless n is prime or a power of a prime. Together with the test of step 1, this implies that n is always prime at step 6.[1]

Example 1: *n* = 31 is prime

Input: integer $n = 31 > 1$.

```
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[Composite]]
  ];
  a=n^(1/2)...n^(1/4)={5.568, 3.141, 2.360}
```

```
Find the smallest r such that O_r(n) > (log_2 n)^2.
  maxk=⌊(log_2 n)^2⌋;
  maxr=Max[3, ⌈(Log_2 n)^5⌉]; (*maxr really isn't needed*)
  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
```

```
If 1 < gcd(a,n) < n for some a ≤ 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} >/1
```

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

  31 > 29
```

```
For a = 1 to ⌊√φ(r) log_2(n)⌋ do
  if (X+a)^n ≠ X^n+a (mod 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]√ϕ[r]];
  For[a=1, a ≤ max, a++,
    If[PolyModulo[(x+a)^n]-PolynomialRemainder[x^n+a, x^r-1, x]≠0,
      Return[Composite]
    ]
  ];

  $(x+a)^{31}$ =
    $a^{31}$ +$31a^{30}x$ +$465a^{29}x^2$ +$4495a^{28}x^3$ +$31465a^{27}x^4$ +$169911a^{26}x^5$ +$736281a^{25}x^6$ +$2629575a^{24}x^7$ +$7888725a^{23}x^8$ +$20160075a^{22}x^9$ +443

  PolynomialRemainder [$(x+a)^{31}$, $x^{29}$-1] =
    $465a^2$ +$a^{31}$ +$(31a+31a^{30})x$ +$(1+465a^{29})x^2$ +$4495a^{28}x^3$ +$31465a^{27}x^4$ +$169911a^{26}x^5$ +$736281a^{25}x^6$ +$2629575a^{24}x^7$ +$7888725a^{23}x^8$ +2

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

  {$1^{31}$-1=0 (mod 31), $2^{31}$-2=0 (mod 31), $3^{31}$-3=0 (mod 31), ..., $26^{31}$-26=0 (mod 31)}
```

```
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] = $x+2x^2+0x^3$

[5]

References

1. Agrawal, Manindra; Kayal, Neeraj; Saxena, Nitin (2004). "PRIMES is in P" (http://www.cse.iitk.ac.in/users/manindra/algebra/primality_v6.pdf) (PDF). *Annals of Mathematics*. **160** (2): 781–793. doi:10.4007/annals.2004.160.781 (https://dx.doi.org/10.4007%2Fannals.2004.160.781). JSTOR 3597229 (https://www.jstor.org/stable/3597229).
2. H. W. Lenstra Jr. and Carl Pomerance, "Primality testing with Gaussian periods (http://www.math.dartmouth.edu/~carlp/PDF/complexity12.pdf)", preliminary version July 20, 2005.
3. H. W. Lenstra jr. and Carl Pomerance, "Primality testing with Gaussian periods (http://www.math.dartmouth.edu/~carlp/aks041411.pdf)", version of April 12, 2011.
4. Daniel J. Bernstein, "Proving Primality After Agrawal-Kayal-Saxena (https://cr.yp.to/papers/aks.pdf)", version of January 25, 2003.
5. See AKS Talk page for a discussion on why 'Example 2: n is not Prime past Step 4' is missing.

Further reading

- 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 (https://zbmath.org/?format=complete&q=an:1058.11070).

External links

- Weisstein, Eric W. "AKS Primality Test" (http://mathworld.wolfram.com/AKSPrimalityTest.html). *MathWorld*.

- R. Crandall, Apple ACG, and J. Papadopoulos (March 18, 2003): On the implementation of AKS-class primality tests (https://web.archive.org/web/20140219064936/http://www.dm.unito.it/~cerruti/ac/aks-crandall.pdf) (PDF)
- Article by Borneman, containing photos and information about the three Indian scientists (http://www.ams.org/notices/200305/fea-bornemann.pdf) (PDF)
- Andrew Granville: It is easy to determine whether a given integer is prime (http://www.ams.org/bull/2005-42-01/S0273-0979-04-01037-7/home.html)
- The Prime Facts: From Euclid to AKS (http://www.scottaaronson.com/writings/prime.pdf), by Scott Aaronson (PDF)
- The PRIMES is in P little FAQ (http://www.instantlogic.net/publications/PRIMES%20is%20in%20P%20little%20FAQ.htm) by Anton Stiglic
- 2006 Gödel Prize Citation (http://www.sigact.org/Prizes/Godel/2006.html)
- 2006 Fulkerson Prize Citation (http://www.ams.org/notices/200611/comm-fulkerson.pdf)
- The AKS "PRIMES in P" Algorithm Resource (http://fatphil.org/maths/AKS)
- Grime, Dr. James. "Fool-Proof Test for Primes - Numberphile" (https://www.youtube.com/watch?v=HvMSRWTE2mI&feature=youtu.be) (video). Brady Haran. [the video describes the exponential time relation (1), which it calls AKS]