University of California,Santa Barbara

Senior Thesis

Run Time Efficiency and the AKS Primality Test

Author:
Roeland Singer-Heinze

Supervisor:
Dr. Jeffrey Stopple

May 30, 2013

Abstract. A primality test is an algorithm which determines if any given number is prime. Until the AKS (Agrawal-Kayal-Saxena) Primality Test was unveiled, there was no previous test that managed to be concurrently general, unconditional, predictable in behavior, and have polynomial bound for run time. The original proof of the AKS algorithm showed that its asymptotic time complexity has a run-time of $\tilde{O}(\log^{21/2} n)$, but it is suspected to be possibly as low as $\tilde{O}(\log^6 n)$. This paper will involve a discussion of the basic proof and ideas behind the AKS Primality Test. I will also discuss how the speed of the algorithm affects its runtime efficiency on a computer and why this is important. I will conclude by studying the latest conjectures and research concerning how the algorithm for the AKS Primality Test may be improved.

Contents

1. Introduction

This paper will be a thorough analysis of the AKS Primality Test. Previous primality proving algorithms were either too laborious to compute in particular cases, or were dependent upon unproven ideas such as the Riemann Hypothesis. The AKS Primality Test is an elegant solution to this age old problem that dates back to times of Gauss and Fermat. It is based on the following neat characterization of primes.

Agrawal, Kayal and Saxena. *For given integer $n \geq 2$, let r be a positive integer less than n such that the order of n modulo r, $o_r(n)$, is $> \log^2 n$. Then n is prime if and only if*

(1) *n is not a perfect power,*
(2) *n has no prime factors $\leq r$,*
(3) *and $(x+a)^n \equiv x^n + a \mod (n, x^r - 1)$ for each integer a such that $1 \leq a \leq \sqrt{r}\log n$.*

We will explain later what "order" and "$\equiv$" mean in this context. Noting that here the congruences are between polynomials of x, and not between two sides evaluated at each integer x. This characterization of prime numbers may seem overly complicated, but as we will see, it is a natural development of the ideas behind finding a quick foolproof algorithm for determining whether a given integer is prime.

One of the key ideas in approaches to primality testing is the Child's Binomial Theorem.

Theorem (Child's Binomial Theorem)**.** *Let R be a commutative ring of characteristic $p > 0$ where p is prime. Then for $x, y \in R$ we have*

$$(x+y)^p = x^p + y^p$$

In particular, for all $a, b \in \mathbb{Z}$ we have

$$(a+b)^p \equiv a^p + b^p \mod p$$

Proof. The proof of the Child's Binomial Theorem follows from the Binomial Theorem, which states that for field F with characteristic p, $n \in \mathbb{Z}^+$, and $x, y \in \mathbb{F}$ we have

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

$$\Rightarrow (x+y)^p = \sum_{k=0}^{n} \frac{n!}{k!(n-k)!} x^{n-k} y^k$$

Notice that p divides $\frac{p!}{k!(p-k)!}$ unless $k = p$ or $k = 0$ (The fraction would then reduce to 1). Because F is characteristic p, then the coefficients of all these terms goes to zero and we are left with

$$(x + y)^p = x^p + y^p.$$

It is straightforward to see then that for $a, b \in \mathbb{Z}$ that for prime p we have

$$(a + b)^p \equiv a^p + b^p \mod p.$$

This is because $\mathbb{Z}_p$ is the field of integers modulo p, which has characteristic p, and we can just apply the previous formula. $\square$

This result is interesting as it allows us to deduce another important property.

Theorem (Fermat's Little Theorem). *If p is a prime integer, then p divides $a^p - a$ for all integers a. In other words, we have*

$$a^p \equiv a \mod p$$

Proof. We proceed by induction on $a \geq 0$. Obviously the base base is when $a = 0$, and $0^p \equiv 0 \mod p$. Now lets assume that this is true for $a = k$. Lets show that this is true for $a = k + 1$. By the Child's Binomial Theorem we have that

$$(k + 1)^p \equiv k^p + 1^p \mod p.$$

By our inductive hypothesis we know that $k^p \equiv k \mod p$ and trivially we know that $1^p = 1$. Thus, it follows that

$$(k + 1)^p \equiv k + 1 \mod p.$$

This concludes our proof of the theorem for $a = k + 1$. $\square$

Perhaps the first question this theorem raises for us is if this sufficient to distinguish primes from composites. Unfortunately, it is not since for example,

$$2^{341} \equiv 2 \mod 341,$$

but $341 = 11 \cdot 31$. However, if we use $a = 3$ instead of $a = 2$, we get

$$3^{341} \equiv 168 \mod 341,$$

so we are still able to prove that 341 is composite. In this case we would call 341 a Fermat pseudoprime base 2. The natural question here then would be if there are composite numbers that satisfy Fermat's Little Theorem no matter what value of a we choose. Lamentably again, there are since for example,

$$a^{1105} \equiv a \mod 1105,$$

for all values of a. Yet $1105 = 5 \cdot 13 \cdot 17$, so it is not prime. This is problematic since it means that there are composite numbers that are Fermat pseudoprimes to any base value, and so by itself, Fermat's Little Theorem is not a sufficient primality test. We call these composite numbers Carmichael numbers.

Definition. *A Carmichael number is a composite positive integer n which satisfies Fermat's Little Theorem for all base values of a. In other words,*

$$a^n \equiv a \mod n$$

for all integers a.

It has been proven that there are an infinite amount of Carmichael numbers, so it is not a problem that can be ignored. Thus we need a method that can determine whether a given number is prime when we think it is prime. Furthermore, this method must be efficient, otherwise it would be useless for practical purposes.

In order to prove the main theorem of Agrawal, Kayal, and Saxena, we will need a couple additional facts from elementary number theory and abstract algebra.

Definition. *A perfect power is a number n of the form a^k where $a > 1, k \geq 2$ are positive integers.*

Definition. *We define Euler's totient function $\phi(n)$ as the number of positive integers less than or equal to n that are relatively prime to n.*

Theorem. *For every finite field $\mathbb{F}_q$, the multiplicative group $\mathbb{F}_q^*$ of nonzero elements of $\mathbb{F}_q$ is cyclic.*

Proof. Assume that $q \geq 3$. Let $h = q - 1$, the order of $\mathbb{F}_q^*$, and let $h = p_1^{r_1} p_2^{r_2} \cdots p_m^{r_m}$ be its prime factor decomposition. Then for each i, $1 \leq i \leq m$, the polynomial $x^{h/p_i} - 1$ has at most h/p_i roots in $\mathbb{F}_q$. Since $h/p_i < h$, it follows that there are nonzero elements of $\mathbb{F}_q$ that are not roots of this polynomial. We let a_i be one of these elements, and set $b_i = a_i^{h/p_i^{r_i}}$. So then $C = 1$, and so the order of b_i divides $p_i^{r_i}$. Thus it has form $p_i^{s_i}$ for some $0 \leq s_i \leq r_i$. But since

$$b^{h/p_i^{r_i-1}} = a_i^{h/p_i} \neq 1,$$

then the order of b_i is precisely $p_i^{r_i}$.

Now let $b = b_1 b_2 \cdots b_m$. We claim that b has order h, which would make it a generator for the group. Suppose on the contrary that the order of b is a proper divisor of h. Thus it is a divisor of at least one of the m integers h/p_i, with $1/leqi/leqm$. Without loss of generality, say it is h/p_1. Therefore,

$$1 = b^{h/p_1} = b_1^{h/p_1} b_2^{h/p_1} \cdots b_m^{h/p_1}.$$

So then if $2 \leq i \leq m$, then $p_i^{r_i}$ divides h/p_1, and so b_i^{h/p_1}. This means $b_1^{h/p_1} = 1$. Thus the order of b_1 must divide h/p_1, which is impossible since the order of b_1 is $p_1^{r_1}$. Thus $\mathbb{F}_q^*$ is a cyclic group with generator b. $\square$

2. Proof of the AKS Algorithm

Since Fermat's Little Theorem alone is not a sufficient primality test, then we must approach the problem from a different angle. This is where the work of Agrawal, Kayal, and Saxena comes in. Their ideas are based on a result of the Child's Binomial theorem.

Theorem. *An integer n is prime if and only if $(x+b)^n \equiv x^n + b \mod n$ in $\mathbb{Z}[x]$ where $gcd(b,n) = 1$.*

Proof. This follows directly from the Child's Binomial Theorem. □

A valid primality test then would be to compute $(x+b)^n - (x^n + b) \mod n$ and see if n divides each coefficients. However, this would be very slow considering that to compute $(x+b)^n (\mod n)$, we would need to store n coefficients. To get around the problem of a high-degree polynomial, one could simply compute the polynomial modulo some small degree polynomial. This is in addition to computing it modulo n as well. A simple polynomial of degree r would be $x^r - 1$. All we would need to verify then is that

$$(x+b)^n \equiv x^n + b \mod (n, x^r - 1) \tag{1}$$

This is true for any prime n because of our theorem above, and this can be computed fairly quick. The only question here is whether this fails to hold for all composite n, which would make this a true primality test. The main theorem of Agrawl, Kayal, and Saxena, which is stated at the beginning, provides a necessary modification of this congruence. It succeeds for primes and fails for composites, and is also polynomial time bounded. We now proceed to prove it.

We assume we are given an odd integer $n > 2$, which we know is not a perfect power. Furthermore, for our positive integer r less than n such that $o_r(n) > \log^2 n$, n contains no primes factors $\leq r$. Lastly, the following congruence must hold for each integer a where $1 \leq a \leq \sqrt{r} \log n$.

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

We know that if n is prime, then these assumptions hold because of our theorem above. Thus, we only need to show that if n is composite, then these hypotheses cannot hold. Assume that n is composite and that p is one of its prime factors. Then, by our above congruence we have that

$$(x+a)^n \equiv x^n + a \mod (p, x^r - 1). \tag{3}$$

Now it is possible to factor the polynomial $x^r - 1$ into irreducible factors in $\mathbb{Z}[x]$ as $\prod\limits_{d|r} \Phi_d(x)$. $\Phi_d(x)$ is the dth cyclotomic polynomial, whose roots are the primitive dth roots of unity, and each one is irreducible in $Z[x]$. However, each cyclotomic polynomial $\Phi_r(x)$ might not be irreducible in $(\mathbb{Z}/p\mathbb{Z})[x]$. So let $h(x)$ be an irreducible factor of $\Phi_r(x) \mod p$. This implies that

$$(x+a)^n \equiv x^n + a \mod (p, h(x)) \tag{4}$$

The congruence classes $\mod (p, h(x))$ can be seen as elements of the ring $\mathbb{F} \equiv \mathbb{Z}[x]/(p, h(x))$. This is isomorphic to the field of p^m elements where m is the degree of $h(x)$. Thus we can see that the non-zero elements of $\mathbb{F}$ form a cyclic group of order $p^m - 1$ and since $\mathbb{F}$ contains x, an element of order r, then r must divide $p^m - 1$. Since $\mathbb{F}$ is a field, its congruence classes are much more straightforward than those in (2), whose congruences do not correspond to a field.

Now consider the set H of the elements $\mod (p, x^r - 1)$ generated multiplicatively by $x + a_i$ where $0 \leq a_i \leq \lfloor \sqrt{r} \log n \rfloor$. Then let G be the (cyclic) subgroup of $\mathbb{F}$ generated multiplicatively by $x + a_i$. G is cyclic because it is a finite subgroup of the multiplicative group of $\mathbb{F}$ since all of its elements are non-zero. This is because if $x + a = 0$ in $\mathbb{F}$, then $x^n + a = (x+a)^n = 0$ in $\mathbb{F}$. This implies then that $x^n = -a = x$ in $\mathbb{F}$, and that $n \equiv 1(\mod r)$. But then our order d of r equals 1, which is a contradiction to our hypothesis. We can view G here as the reduction of $H \mod (p, h(x))$.

Consider now that if $g(x) = \prod_{0 \leq a \leq \lfloor \sqrt{r} \log n \rfloor} (x+a)^{e_a} \in H$, then by (3) we have that

$$g(x)^n = \prod_a ((x+a)^n)^{e_a} \equiv \prod_a (x^n + a)^{e_a} = g(x^n) \mod (p, x^r - 1)$$

So define S to be the set of positive integers k such that $g(x^k) \equiv g(x)^k \mod (p, x^r - 1)$ for all $g \in H$. It follows then that $g(x^k) \equiv g(x)^k$ in $\mathbb{F}$ for each $k \in S$, and so the Child's Binomial Theorem holds for elements of G in this field for any exponent in S. This includes p and n which are elements of S.

To prove our claim we will establish upper and low bounds on the size of G to derive a contradiction. For the upper bound we first need to prove a couple of lemmas.

Lemma 1. *If $a, b \in S$, then $ab \in S$.*

Proof. Let $g(x) \in H$. We want to prove that if $a, b \in S$, then $g(x)^{ab} \equiv g(x^{ab}) \mod (p, x^r - 1)$. So since $b \in S$, then $g(x^b) \equiv g(x)^b \mod (p, x^r - 1)$. By substituting in x^a for x we get $g((x^a)^b) \equiv g(x^a)^b \mod (p, (x^a)^r - 1)$. Notice though that $x^r - 1$ divides $x^{ar} - 1$, and therefore, our equation is $\mod (p, x^r - 1$. Thus,

$$g(x)^{ab} = (g(x)^a)^b \equiv g(x^a)^b \equiv g((x^a)^b) \equiv g(x^{ab}) \mod (p, x^r - 1)$$

and so we have obtained the desired result. □

Lemma 2. *If $a, b \in S$ and $a \equiv b \mod r$, then $a \equiv b \mod |G|$.*

Proof. Let $g(x) \in \mathbb{Z}[x]$. It is simple to see that $u - v$ divides $g(u) - g(v)$ since if

$$g(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

then,

$$g(u) - g(v) = a_n(u^n - v^n) + a_{n-1}(u^{n-1} - v^{n-1}) + \cdots + a_1(u - v)$$

and clearly $u - v$ divides every term. So since $a \equiv b \mod r$, then $a - b = mr$ for some integer m. Thus r divides $a - b$ and so $x^r - 1$ divides $x^{a-b} - 1$, which divides $x^a - x^b$. Therefore since $x^a - x^b$ divides $g(x^a) - g(x^b)$, $x^r - 1$ divides $g(x^a) - g(x^b)$. We deduce that if $g(x) \in H$, then

$$g(x)^a \equiv g(x^a) \equiv g(x^b) \equiv g(x)^b \mod (p, x^r - 1).$$

And if $g(x) \in G$, then $g(x)^a \equiv g(x)^b \mod (p, h(x))$ which implies that $g(x)^{a-b} \equiv 1$ in $\mathbb{F}$. Since G is a cyclic group though, we can take g to be a generator of G and because $|G|$ divides the order of a generator, then $|G|$ divides $a - b$. □

We now use the lemmas to establish an upper bound on $|G|$. Let R be the subgroup of $(\mathbb{Z}/r\mathbb{Z})^*$ generated by n and p. By our hypothesis, n cannot be a power of p. Therefore integers of the form $n^i p^j$ with $i, j \geq 0$ are distinct. Furthermore, the amount of such integers with $0 \geq i, j \geq \sqrt{|R|}$ is greater than $|R|$. Thus two of these integers must be congruent modulo r, say

$$n^i p^j \equiv n^l p^k \mod r.$$

By Lemma 1 these integers are both in S, and by Lemma 2 their difference is divisible by $|G|$. Thus we have that

$$|G| \le |n^i p^j - n^l p^k| \le (np)^{\sqrt{|R|}} - 1 < n^{2\sqrt{|R|}} - 1.$$

We carefully take notice here that $n^i p^j - n^l p^k$ is non-zero since n is not a prime nor a perfect power. This bound, however, can be improved by replacing n above with n/p to get

$$|G| \le n^{\sqrt{|R|}} - 1 \tag{5}$$

as our new upper bound. We only need to show then that $n/p \in S$. So because n has order $d \mod r$, then $n^d \equiv 1 \mod r$. Thus, $x^{n^d} \equiv x \mod (x^r - 1)$. Now suppose that $a \in S$ and that $b \equiv a \mod (n^d - 1)$. Then we see that $x^r - 1$ divides $x^{n^d} - x$, which divides $x^b - x^a$, which divides $g(x^b) - g(x^a)$ for any $g(x) \in H$. So by Lemma 1, if $g(x) \in H$, then

$$g(x)^{n^d} \equiv g(x^{n^d}) \mod (p, x^r - 1)$$

because $n \in S$. Thus $g(x^{n^d}) \equiv g(x) \mod (p, x^r - 1)$, as $x^r - 1$ divides $x^{n^d} - x$, so that $g(x)^{n^d} \equiv g(x) \mod (p, x^r - 1)$. But this implies that $g(x)^b \equiv g(x)^a \mod (p, x^r - 1)$ since $n^d - 1$ divides $b - a$. So we get that

$$g(x^b) \equiv g(x^a) \equiv g(x)^a \equiv g(x)^b \mod (p, x^r - 1)$$

as $a \in S$ which implies that $b \in S$. Now let $b = n/p$ and $a = np^{\phi(n^d-1)-1}$ so that $a \in S$ by Lemma 1 since $p, n \in S$. Moreover, $b \equiv a \mod (n^d - 1)$ so $b = n/p \in S$ by the above and so we got our upper bound.

Now that we have established an upper bound on $|G|$, lets establish a lower bound. To derive a contradiction, we want to show that there are many distinct elements of G. So if $f(x), g(x) \in \mathbb{Z}[x]$ with $f(x) \equiv g(x) \mod (p, h(x))$, then $f(x) - g(x) \equiv h(x)k(x) \mod p$ for some polynomial $k(x) \in \mathbb{Z}[x]$. Therefore, if f and g have smaller degree than h, then $k(x) \equiv 0 \mod p$ and so $f(x) \equiv g(x) \mod p$. Thus all polynomials of the form $\prod_{1 \le a \le \sqrt{r}\log n} (x + a)^{e_a}$ with degree less than the degree of $h(x)$ are distinct elements of G. Hence if the order of p mod r is large, then we can find an appropriate lower bound on $|G|$.

To prove that such an r exists is non-trivial and requires tools of analytic number theory. Agrawal, Kayal, and Saxena were later able to replace this term m with $|R|$ in this result. This allowed them to give a much more elementary proof of their theorem, and gives a stronger result when they do invoke the deeper estimates.

Lemma 3. *Suppose that $f(x), g(x) \in \mathbb{Z}[x]$ with $f(x) \equiv g(x) \mod (p, h(x))$ where the reductions of f and g in $\mathbb{F}$ are in G. Then if f and g both have degree less than $|R|$, then $f(x) \equiv g(x) \mod p$.*

Proof. Let $\triangle(y) = f(y) - g(y) \in \mathbb{Z}[y]$ be considered as "reduced" in $\mathbb{F}$. Then if $k \in S$, we have that

$$\triangle(x^k) = f(x^k) - g(x^k) \equiv f(x)^k - g(x)^k \equiv 0 \mod (p, h(x)).$$

As stated before, x has order r in $\mathbb{F}$ so then $x^k : k \in R$ are all distinct roots of $\triangle(y) \mod (p, h(x))$. Notice now that $\triangle(y)$ has degree $< |R|$, but has $\geq |R|$ distinct roots $\mod (p, h(x))$. Thus $\triangle(y) \equiv 0 \mod (p, h(x))$, and this implies that $\triangle(y) \equiv 0 \mod p$ as its coefficients are independent of x. □

Now, by definition R contains all elements generated by $n \mod r$. Thus R is at least as big as d, the order of $n \mod r$ that is greater than $(\log n)^2$ by assumption. Also, $\sqrt{r} \log n$, $|R| > B$, where $B = \lfloor \sqrt{|R|} \log n \rfloor$. By Lemma 3, this implies that the products $\prod_{a \in T}(x + a)$ give distinct elements of G for every proper subset T of $0, 1, 2, \ldots, B$. Therefore, we get that

$$|G| \geq 2^{B+1} > n^{\sqrt{|R|}} - 1,$$

and this contradicts our upper bound in (5). This concludes our proof of the theorem of Agrawal, Kayal, and Saxena.

3. Runtime Analysis and Computational Complexity

Efficiency is the whole point of devising a primality test. Otherwise, if computational labour were not a concern, then none of this would even matter. We could simply use trial division to determine whether a given number is prime or not. Precisely because such approaches would be impractical, we have to devise a primality testing algorithm that is computable in a reasonable amount of time. For our purposes here, we will need to understand a little bit of computational complexity theory.

Definition. *A decision problem is any question or problem with a yes or no answer depending on an arbitrary number of inputs.*

We note here that our "inputs" are integers represented by binary strings.Now our problem of determining prime numbers is exactly just this. For a given integer, our input, determine if it is prime or not. The output here would be a yes or no answer, and thus primality testing falls into the class of decision problems. However, we are interested if this particular decision problem is computationally feasible. The standard convention for determining whether an algorithm is feasible if it runs in polynomial time.

Definition. *We denote by P the class of decisions problems that can be solved in polynomial time*

Essentially, there exists some polynomial p such that the algorithm runs in at most $p(n)$ inputs of length n. Our goal here is to prove that there exists a primality algorithm that is unconditional and deterministic, which resides in the class P.

For runtime analysis we need a tool for analysing our computational complexity. The standard for such analysis is big-O notation.

Definition. *We say a function $f(n)$ is of $O(a(n))$ complexity if there exists positive constants M and N such that*

$$|f(n)| \leq M|a(n)|$$

for all $n \geq N$.

This notation will provide an upper bound on the growth rate of our function. We are interested in algorithms that are solvable in polynomial time.

Definition. *An algorithm is solvable in polynomial time if the number of steps to complete the algorithm for a given input is $O(n^k)$ for some positive integer k, where n is the complexity of the input.*

For example, consider if our algorithm $f(n) = 3n^2+5$. We can show that its computational complexity is $O(n^2)$ by choosing $M = 4$ and $N = 3$, since for all $n \geq N$,

$$3n^2 + 5 \leq 4n^2.$$

Before we proceed further we will need one last tool for analysing the time complexity of our algorithm.

Definition. *A function $f(n)$ is of $\tilde{O}(a(n))$ complexity if $f(n)$ is of $O(a(n)\log^k a(n))$ complexity.*

Essentially this is just a shorthand notation that ignores logarithmic factors since they are irrelevant to determining whether or not an algorithm is polynomial time bounded (Because $\log n < n$).

To explain where such logarithmic factors come up in our calculations, we have to first understand that arithmetic between integers on a computer is done in binary. This is to say that the size of our input, the number of binary digits, directly impacts the operations upon these bits. For example, consider two integers, x and y, with no more than n binary digits. Then obviously, addition and subtraction would take no more than n steps, which would give us an algorithmic computational complexity of $O(n)$.

The AKS algorithm is of course a bit more complicated, but the idea of using "bit operations" on the binary numbers remains the same. We can summarize the algorithm in the following steps.

The AKS Primality Test

On input n, where n is a positive integer

```
(1) If ($n = a^b$ for $a \in \mathbb{N}$ and $b > 1$), output COMPOSITE.
(2) Find the smallest $r$ such that $o_r(n) > \log^2 n$.
(3) If $1 < gcd(a, n) < n$ for some $a \leq r$, output COMPOSITE.
(4) If $n \leq r$, output PRIME.
(5) For $a = 1$ to $\lfloor \sqrt{\phi(r)} \log n \rfloor$
(6) If $((X + a)^n \neq X^n + a( \mod X^r - 1, n))$, output COMPOSITE.
(7) Output PRIME
```

A key part of determining the run time of the algorithm is getting a bound on r. By computing modulo $X^r - 1$, we were trying to reduce the amount of terms in the polynomial.

Theorem. *For $n \geq 6$, there exists an $r \in [\log^5 n, 2\log^5 n]$ such that*

$$o_r(n) > \log^2 n. \tag{6}$$

Proof. For easier notation let $B = (\log n)^5$. Assume our theorem isn't true. Then $o_r(n) \leq (\log n)^2$ for every prime $r \in [B, 2B]$, so that their product divides $\prod_{i \leq \log^2 n}(n^i - 1)$. Now we know by the prime number theorem, that the product of primes between B and $2B$ is $\geq 2^B$. But then we get that

$$2^B \leq \prod_{B \leq r \leq 2B} r \leq \prod_{i \leq \log^2 n} (n^i - 1) < n^{\sum_{i \leq \log^2 n} i} < 2^{\log^5 n},$$

for $n \geq 6$, which is a contradiction. $\square$

Now before we prove the time complexity of the algorithm, we summarize some already known results.

- Let a, b be integers with $|a|, |b| \leq n$. Then $b \mod a$ can be computed in $O(\log^2 n)$ time. By [2], this follows from a recursive division algorithm given in.

- Let a, b be integers with $|a|, |b| \leq n$. Then ab can be computed in $\tilde{O}(\log n)$ time. By [2], this follows from the Fast Fourier Transform algorithm given , which allows d-digit numbers to be multiplied in $O(d \log d)$ time. And since an integer n can be expressed with $\lceil \log n \rceil$ digits, the multiplication runs in

$$O(d \log d) = O(\log n \log \log n) = \tilde{O}(\log n) \tag{7}$$

  time for $|a|, |b| \leq n$.

- Let $p(x), q(x)$ be polynomials with degree less than or equal to r and coefficients with absolute value less than or equal to n. Then $p(x)q(x)$ can be computed in $\tilde{O}(r \log n)$ time. This also follows from the Fast Fourier Transform algorithm applied to polynomials, which allows the product of degree r polynomials to be computed with $O(r \log r) = \tilde{O}(r)$ multiplications. Since each of these multiplications takes $\tilde{O}(\log n)$ time by (7), then we get a total time complexity of $\tilde{O}(r \log n)$.

- Let $n \geq 2$ be an integer. Then by [2], there is an algorithm that tests if there are integers $a \geq 2$ and $b \geq 2$ such that $n = a^b$ in $\tilde{O}(\log^3 n)$ time.

Theorem. *The AKS algorithm runs in* $\tilde{O}(\log^{21/2} n)$ *time.*

Proof. By our summary above, step 1 of the algorithm takes $\tilde{O}(\log^3 n)$ time.

Now for step 2, we can simply try successive values of r until we find one such that $n^k \neq 1$ mod r for all $k \leq \log^2 n$. For a particular r this would involve at most $O(\log^2 n)$ multiplications modulo r. Since we are multiply modulo r each product will have factors less than r. Hence, again by our summary, each multiplication takes $\tilde{O}(\log r)$ time. Therefore the computations on each r require $\tilde{O}(\log r) \cdot \log^2 n = \tilde{O}(\log r \log^2 n)$ time. Now by our previous theorem, we know that only $O(\log^5 n)$ different r's need to be tried. Thus, the total run time of step 2 is $\log^5 n \cdot \tilde{O}(\log r \log^2 n) = \tilde{O}(\log \log^5 n \log^2 n \log^5 n) = \tilde{O}(\log^7 n)$ time.

Step 3 requires computing the gcd of r numbers, where r is bounded by our previous theorem. By [2], computing the gcd takes $O(\log n)$ time. Therefore, the total time complexity of this step is $O(r \log n) = O(\log^6 n)$.

Step 4 simply compares r and n. This can be done by counting the number of digits in n and then seeing if r has that many or more. This takes time proportional to the number of binary digits in n, so it takes about $O(\log n)$ time.

Step 5 goes through all values of a from 1 to $\lfloor \sqrt{\phi(r)} \log n \rfloor$. Since $\phi(r) \leq r - 1$ for all r, as r is prime and all integers less than it are coprime to it, then the time complexity of the step is $O(\lfloor \sqrt{\phi(r)} \log n \rfloor) = O(\sqrt{r} \log n)$.

Finally step 6 computes $(X + a)^n$ and $X^n + a \mod (X^r - 1, n)$. Naively it could take n multiplications to compute $(X + a)^n \mod (X^r - 1, n)$, but we can do better by squaring $(X + a) \mod (X^r - 1, n)$ $\lfloor \log n \rfloor$ times, which gives us the values of $(X + a)^{2j} \mod (X^r - 1, n)$ for $1 \leq j \leq \lfloor \log n \rfloor$. Hence we can then multiply the appropriate powers of $(X + a)^{2j} \mod (X^r - 1, n)$ to compute $(X + a)^n \mod (X^r - 1, n)$. This method requires $O(\log n)$ multiplications $\mod (X^r - 1, n)$, $\lfloor \log n \rfloor$ to compute the powers of $(X + a)$, and up to another $\lfloor \log n \rfloor$ to get $(X + a)^n$. Since since each product is taken $\mod (X^r - 1, n)$, every product has factors of degree less than or equal to r. Moreover, every product has factors with coefficients less than or equal to n. Thus, by our summary from before, each multiplication takes $\tilde{O}(r \log n)$ time. This means that computing $(X + a)^n \mod (X^r - 1, n)$ takes a total of $O(\log n) \cdot \tilde{O}(r \log n) = \tilde{O}(r \log^2 n)$ time. Because this is computed for every value of a in step 5, or $O(\sqrt{r} \log n)$ times, the total time complexity of this step is $O(\sqrt{r} \log n) \cdot \tilde{O}(r \log^2 n) = \tilde{O}(r^{3/2} \log^3 n)$ time. Since we have bounded r this gives us a runtime of $\tilde{O}(r^{3/2} \log^3 n) = \tilde{O}((\log^5 n)^{3/2} \log^3 n) = \tilde{O}(\log^{21/2} n)$. This time dominates all the other ones, so the total runtime of our algorithm is indeed $\tilde{(}\log^{21/2} n)$. $\square$

The time complexity of the algorithm may be improved by improving the bounds on r. The best possible scenario would be when $r = O(\log^2 n)$ and in that case we would get a total time complexity of $\tilde{O}(\log^6 n)$. There are a couple conjectures out there that support this. One of them is the Sophei-Germain conjecture.

Conjecture (Sophie-Germain Prime Density Conjecture). *The number of primes $q \leq m$ such that $2q + 1$ is also a prime is asymptotically $\frac{2C_2 m}{\log^2 m}$ where C_2 is the twin prime constant* *(estimated to be approximately* 0.66*). Primes q with this property are called Sophie-Germain primes.*

If this conjecture held, we would be able to conclude that $r = \tilde{O}(\log^2 n)$. By density of Sophie-Germain primes, there must exist at least $\log^2 n$ such primes between $8\log^2 n$ and $c\log^2 n(\log\log n)^2$ for suitable constant c. For any such prime q, we would either have that $o_q(n) \leq 2$ or $o_q(n) \geq \frac{q-1}{2}$. Thus any q such that $o_q(n) \leq 2$ must divide $n^2 - 1$ and so the number of such q is bounded by $O(\log n)$. So then this implies that there must exist a prime $r = \tilde{O}(\log^2 n)$ such that $o_r(n) > \log^2 n$. This r would yield a total runtime of $\tilde{O}(\log^6 n)$. Some progress has been made towards proving this conjecture due to Fouvry.

Lemma 4. *Let $P(m)$ denote the greatest prime divisor of m. There exists constants $c > 0$ and n_0 such that, for all $x \geq n_0$:*

$$\mid \left\{q \mid q \text{ is prime, } q \leq x \text{ and } P(q-1) > q^{2/3}\right\} \mid \geq c\frac{x}{\log x}.$$

Using this lemma we can derive a better runtime.

Theorem. *The asymptotic time complexity of the AKS algorithm is $\tilde{O}(\log^{15/2} n)$.*

Proof. As argued before, a high density prime q such that $P(q-1) > q^{2/3}$ implies that step 2 of the algorithm will find a $r = O(\log^3 n)$ with $o_r(n) > \log^2 n$. Using this, the time complexity of the algorithm is brought down to

$$\tilde{O}(r^{3/2}\log^3 n) = \tilde{O}((\log^3 n)^{3/2}\log^3 n = \tilde{O}(\log^{15/2} n)$$

□

Recently, Hendrik Lenstra and Carl Pomerance have come up with a modified version of the AKS algorithm whose time complexity is provably $\tilde{O}(\log^6 n)$.

4. References

[1] Andrew Granville. It is Easy to Determine Whether a Given Integer is Prime. Bull. Amer. Math. Soc. 42 (2005), 3-38 `http://ams.org/journals/bull/2005-42-01/S0273-0979-04-01037-7/`.

[2] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. Preprint. `http://www.cse.iitk.ac.in/users/manindra/algebra/primality_v6.pdf`, November 2006