PRIMES IS IN P, SLOWLY

MICHAEL STAY

1. Introduction

The problem of determining whether a number has any proper divisors goes back a long way. The sieve of Eratosthenes is a Greek algorithm from 240 BC; it's faster than trial division, but not much. In fact, the time both algorithms take is proportional to the square root of the number being tested. For every two digits, they take ten times longer:

$$T(n) \simeq 10^{1/2 \log n}.$$

Notice that for this algorithm, the time is "exponential in log n"— that is, log n appears in the exponent. An efficient algorithm should only be "polynomial in $\log n$"—that is, the time should increase like the number of digits raised to a power, such as

$$T(n) \simeq (\log n)^5 + 3(\log n)^2$$

or something similar, where the exponents are all constants and log n only appears in the base of an exponentiation.

Algorithms can be either deterministic (they always give an answer, yes or no) or probabalistic (they can say "I don't know"). One polynomial probabalistic algorithm that tests if a number is *not* prime is based on Fermat's little theorem:

$$b^{n-1} = 1 \mod n$$

if b is not a multiple of n and n is prime. Leonhard Euler noticed that he could test many numbers easily for primality, since if the equation doesn't hold for even one such value of b, then the modulus n is not prime. For example, when n=5, a prime:

$$\begin{array}{llllll} 1^4 & = 1 & = 0 * 5 + 1 & 2^4 & = 16 & = 3 * 5 + 1 \\ 3^4 & = 81 & = 16 * 5 + 1 & 4^4 & = 256 & = 51 * 5 + 1 \end{array}$$

On the other hand, if $n = 6$, a composite number, we can take $b = 2$ and get

$$2^{n-1} = 2^5 = 32 = 2 \mod 6.$$

Since $2^{n-1} \neq 1 \mod n$, then n is not prime.

Beware! If this test succeeds for many values of b, it does not tell you that n *is* prime: the Carmichael numbers are those n for which this test succeeds for all b that do not divide n. 561 is one example of a Carmichael number; it passes the test for all b which are not multiples of 11 or 51. So sometimes the test can tell you if the number is composite; the rest of the time it says, "I don't know."

P is the class of problems that one can solve in deterministic polynomial time in the length of the input. In this case, the length of the input is the number of digits in the number to be tested. We would like to know if testing a number for primality is in this class of problems. In the introduction to their paper, Agrawal *et al.* discuss many of the previous algorithms that *almost* got there. One was deterministic, but depended on an unproven hypothesis; another eliminated that dependence at the expense of becoming probabalistic.

Yet another was deterministic and ran in pretty-darned-close-to-polynomial time:

$$T(n) \simeq (\log n)^{O(\log \log \log n)},$$

so the exponent was proportional to log log log n, which is almost constant. For example, if we wrote a digit of the number to be tested on every particle in the observable universe (!), log n would be about 10^{100}, log log n about 100, and log log log n about 2. So it was *very* close to being in P.

But Agrawal, Kayal, and Saxena (AKS) finally gave an algorithm that is provably in P. The algorithm is short—only six steps! The proof that the algorithm works is relatively short, too, and fairly simple. The only part requiring math past college algebra is using Galois fields.

The AKS algorithm uses a test similar to Fermat's little theorem, in that it raises a polynomial to a power and checks that it matches something that's universally true only for primes:

$$(X + a)^p = X^p + a \mod p$$

How do we calculate $(X + a)^n$? Well, $(X + a)$ is a binomial, so we use Pascal's triangle. The coefficients of the terms in $(X + a)^n$ are the numbers on the nth row of Pascal's triangle. If n is a prime, then all the numbers except the first and last will be zero mod n. This is the property that AKS use to check whether n is prime or not.

For example, let $n = 7$. Then the seventh row of Pascal's triangle is

$$1 \quad 7 \quad 21 \quad 35 \quad 35 \quad 21 \quad 7 \quad 1.$$

If we look at the numbers mod 7, we get

$$1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1$$

The 1 at the beginning is the coefficient on X^n, and the 1 at the end is the coefficient on a^n. So

$$(X + a)^n = X^n + a^n = X^n + a$$

for all a if and only if n is prime. For example, if $n = 15$, then the 15th row is

$$1 \quad 15 \quad 105 \quad 455 \quad 1365 \quad 3003 \quad 5005 \quad 6435 \quad 6435 \quad 5005 \quad 3003 \quad 1365 \quad 455 \quad 105 \quad 15 \quad 1$$

and modulo 15, we have

$$1 \quad 0 \quad 0 \quad 5 \quad 0 \quad 3 \quad 10 \quad 0 \quad 0 \quad 10 \quad 3 \quad 0 \quad 5 \quad 0 \quad 0 \quad 1$$

The problem with checking that a row has this form is that the row has $n+1$ entries, and $n + 1$ eventually gets bigger than log n raised to the power of *any* constant. There are way too many entries to check! So given two polynomials $a(X), b(X)$, we calculate the polynomial $(X + a)^n$ and then reduce modulo the polynomial $(X^r - 1)$ and the number we're testing, n.

How does that behave? Since $X^r - 1$ is equivalent to zero, X^r is equivalent to 1. When we multiply X^{r-1} by X, it just wraps around! So reducing modulo $X^r - 1$ is the same as chopping up our resulting polynomial into pieces r coefficients long and adding them all together.

If n is prime, then our row of Pascal's triangle is pretty simple mod n:

$$1 \quad 0 \quad 0 \quad \ldots \quad 0 \quad 0 \quad 1.$$

When we look at

$$(X + a)^n \mod (X^r - 1), n,$$

we get r coefficients, two of which should be 1 and the rest zeros (well, unless r divides $n-1$, in which case the ones land on top of each other and we get a single two with the rest zeros).

Unfortunately, once we reduce modulo $(X^r - 1)$, we get composite numbers like the Carmichael numbers that pass a lot of the tests. If a number n passes Euler's test for some base b—that is, if

$$b^{n-1} = 1 \mod n$$

—then we call n an "Euler pseudoprime to the base b." Carmichael numbers are Euler psudoprimes to nearly every base. Likewise, if a number n passes AKS's test for some polynomial b and modulus (X^r-1), we could call it an "AKS pseudoprime to the base b mod (X^r-1)." AKS were too humble to name the test after themselves, so instead they called such numbers "introspective to the base b mod $(X^r - 1)$."

Their proof shows that if you choose r right, and choose enough different values of $(a(X) + b(X))$, and n passes the test for all of them, then n must be a prime. The first step of the algorithm checks if n is some integer raised to an integer power greater than 1. The proof constructs two [??] different groups, and then puts bounds on their size. They were able to show that if the algorithm outputs PRIME, then there's a lower bound on the size of one of the groups. On the other hand, if n is not a prime power, then the upper bound on the size of that group is less than the bound to output PRIME. So if the algorithm outputs PRIME, it's true!

2. The algorithm and the proof

Now let's look at the algorithm in more detail:

1. Test if n is a power; that is, test if $n = j^k$ for some integers $j \geq 1$ and $k > 1$. If so, output COMPOSITE.

This step is straightforward. The rest of the algorithm can only tell if n is a prime raised to some power; this checks to see that n is not a smaller number raised to some integer power. I.e. it checks if the square root of n is an integer, then checks if the cube root of n is an integer, then the fourth root and so on until the results get smaller than 2. It doesn't need to go any further, since 1 to any power is 1.

2. Find the smallest r such that $o_r(n) > 4(\log n)^2$.

The symbol $o_r(n)$ is the "order of n modulo r," the smallest positive power k for which $n^k - 1 = 0 \mod r$. It's the point at which the powers of $n \mod r$ start repeating themselves.

Consider $n = 5$. If we look at $r = 7$, then the powers of 5 are

$$\begin{aligned}
5^0 &= 1 \mod 7\\
5^1 &= 5\\
5^2 &= 4\\
5^3 &= 6\\
5^4 &= 2\\
5^5 &= 3\\
5^6 &= 1 \leftarrow \text{ starts repeating here, so } o_7(5) = 6\\
5^7 &= 5\\
5^8 &= 6\\
&\ldots
\end{aligned}$$

On the other hand, if we look at $r = 3$, then the powers of 5 are

$$\begin{aligned}
5^0 &= 1 \mod 3\\
5^1 &= 2\\
5^2 &= 1 \leftarrow \text{ starts repeating here, so } o_3(5) = 2\\
5^3 &= 2\\
5^4 &= 1\\
5^5 &= 2\\
5^6 &= 1\\
&\ldots
\end{aligned}$$

Here's a table of $o_r(5)$ for the first few values of r.

r	$o_r(5)$			
2	1	$5^1 - 1$	$= 4$	$= 0 \mod 2$
3	2	$5^2 - 1$	$= 24$	$= 0 \mod 3$
4	1	$5^1 - 1$	$= 4$	$= 0 \mod 4$
6	2	$5^2 - 1$	$= 24$	$= 0 \mod 6$
7	6	$5^6 - 1$	$= 15624$	$= 0 \mod 7$
etc.				

For the algorithm to work, AKS had to prove that there would be a number in the second column that would be big enough [bigger than $4(\log n)^2$] while keeping the number in the first column small enough [smaller than a polynomial in $\log n$—here, they show it's smaller than $16(\log n)^5$]. The proof is by contradiction.

First, AKS state the fact that

$$\text{LCM}(1, 2, \ldots, m) > 2^m \text{ for all } m > 3,$$

where LCM is the "least common multiple," the smallest number that's divisible by everything from 1 to m. I'm not going to prove this fact about the LCM here, but I'll outline the idea. It turns out that a stronger statement is true: the product of all the primes in the interval $[1, m]$ is greater than 2^m for $m > 3$. If you multiply two numbers, you add the logs. Each prime p in the interval $[1, m]$ contributes

$(\log p)/(\log 2)$ bits to the sum of the logs; the probability that a number k is prime is very nearly $1/(logk)$, so the total number of bits that the primes less than or equal to m contribute is around

$$\frac{m\log k}{\log 2\log k} = \frac{m}{log2} \simeq 1.44m \text{ bits.}$$

If a number is 3 digits long, it's around 10^3. In the same way, if a number is $1.44m$ bits long, the number is about $2^{1.44m}$, which is obviously bigger than 2^m for $m \geq 1$.

Specifically, then, if we take the LCM of all the numbers r from 1 up to $16(\log n)^5$, we'll get a number that's bigger than $2^{16(\log n)^5}$.

Now consider the product

$$\prod_{i=1}^{s}(n^i - 1) = (n^1 - 1)(n^2 - 1)(n^3 - 1)\cdots(n^s - 1).$$

If $o_r(n)$ is less than s, then r divides this product evenly, because by the definition of $o_r(n)$, $n^{o_r(n)} - 1 = 0 \mod r$. For example, let $s = 10, n = 5$, and $r = 7$. Then the product is

$$(5^1 - 1)(5^2 - 1)\cdots(5^6 - 1)\cdots(5^{10} - 1).$$

From the table above, we find that $o_r(n) = 6$, which is less than 10, and that $(5^6 - 1) = 2232 * 7$. So 7 divides this product.

Let's look at the r that are small enough, and assume that there are no r for which $o_r(n)$ is big enough:

$$r \leq s = \lfloor 4(\log n)^2 \rfloor.$$

How big is the product above? Well, we know that $i \leq s$, so we can say

$$\prod_{i=1}^{s} n^i - 1 \leq \prod_{i=1}^{s} n^s = n^{s^2}$$

and

$$n^{s^2} = n^{16(\log n)^4}.$$

Now, $n = 2^{\log n/\log 2}$, so

$$\prod_{i=1}^{s} n^i - 1 \leq n^{16(\log n)^4} = 2^{16(\log n)^5/log2} < 2^{16(\log n)^5},$$

and we get a contradiction.

3. Do trial division on n for $a = 2$ to r. If any a divides n, output COMPOSITE.
4. If $r > \sqrt{n}$, output PRIME.

These steps check a few numbers to see if they happen to be factors. If one of them is, then the number obviously isn't prime. If none of them are, and r is bigger than the square root of n, then n has to be prime: if one factor is bigger than the square root of n, the other has to be smaller, and we just checked all those.

5. for $a = 1$ to $\lfloor 2\sqrt{\phi(r)}\log n \rfloor$
   if $(X + a)^n \neq (X^n + a) \mod (X^r - 1), n$
      output COMPOSITE
6. Output PRIME

This is the loop that checks to see if n passes enough AKS pseudoprime tests—to see if it's introspective in enough cases to be sure that n is prime.

The upper limit of the loop is a number they call L:

$$L = \lfloor 2\sqrt{\phi(r)} \log n \rfloor.$$

We need to be sure that L is only a polynomial in $\log n$, that the definition of L is at most $\log n$ raised to a constant power. We already know that r is; how about the rest of the expression? Well, $\phi()$ is Euler's totient function. It is a very interesting function, and I recommend that you go read about it. But we don't need to know anything about it here except that $\phi(r)$ is always less than r. Taking the square root divides the exponent by 2, and multiplying by $\log n$ increases it by 1. So the resulting exponent is still a constant, and we're OK.

At this point, AKS introduce a number p. This is any prime dividing n. If n is prime, then $p = n$. If n is not prime, then p is at most $n/2$, since we have to multiply p by at least 2 to get a composite number.

In the algorithm, AKS only check a special case of introspectivity where the base is a binomial, i.e. powers of the polynomial $X + a$. In the proof, however, they use arbitrary polynomials. Let $b(X)$ be a polynomial. A number m is called "introspective for $b \mod p$" if $p \mid m$ and

$$b(X)^m = b(X^m) \mod (X^r - 1), p.$$

If two numbers m and m' are introspective for $b \mod p$, then so is m times m':

$$\begin{aligned} & b(X)^{mm'} \mod (X^r - 1), p \\ =\ & b(X^m)^{m'} \mod (X^r - 1), p \\ =\ & b(X^{mm'}) \mod (X^r - 1), p \end{aligned}$$

Also, if m is introspective for any two functions $f(X)$ and $f'(X)$ mod p, then m is introspective for $f(X) \cdot f'(X) \mod p$:

$$\begin{aligned} & (f(X) \cdot f'(X))^m \mod (X^r - 1), p \\ =\ & f(X)^m f'(X)^m \mod (X^r - 1), p \\ =\ & f(X^m) f'(X^m) \mod (X^r - 1), p \end{aligned}$$

If the algorithm does not output COMPOSITEat this step, then n passed the test for L different values of $f(X)$. The prime p would also pass the tests (primes pass every primality test), so every integer of the form

$$m = n^i p^j$$

is introspective for every polynomial of the form

$$f(X) = (X + 1)^{e_1}(X + 2)^{e_2}(X + 3)^{e_3} \cdots (X + L)^{e_L}.$$

AKS call this set of integers "I" and this set of polynomials "P".

From each set, we define a group. The first group, called G, is the set of remainders you get after dividing each element of I by r. In other words, $G = I \mod r$.

For example, if $n = 15, p = 5$, then I is the set of integers

$$\begin{aligned}\{m\} = \{15^i 5^j\} = \{&1,\\ &15, 5,\\ &15^2, 15 \cdot 5, 5^2,\\ &15^3, 15^2 \cdot 5, 15 \cdot 5^2, 5^3,\\ &\ldots\}\\ = \{&1, 5, 15, 25, 75, 125, 225, 375, \ldots\}\end{aligned}$$

If $r = 4$, then $I \mod r = \{1, 1, 3, 1, 3, 1, 1, 3, \ldots\} = \{1, 3\}$.

Since n and p are coprime to r (otherwise we'd have found a factor of r as a factor of n in step 3), every element in I has an inverse modulo r.

How big is G? Well, if n is prime, then we're just looking at all the powers of $p \mod r$. As soon as $p^k = 1 mod r$, they start repeating, so there will be $k = o_r(n)$ different elements of G. If n is not prime, then there will be more. So

$$t = |G| \geq o_r(n) > 4(\log n)^2.$$

The second group is fundamentally the same. Rather than reduce $I \mod n$, we reduce $P \mod Q_r(X), n$, where $Q_r(X)$ is a cyclotomic polynomial. The nth cyclotomic polynomial Q_n is defined as

$$Q_n(X) = \prod_{\substack{1<i<n\\ gcd(i,n)=1}} X - g^i,$$

where g is a primitive nth root of unity and X is any element of the field.

The first few cyclotomic polynomials are

$$\begin{aligned}Q_1(X) &= X - 1\\ Q_2(X) &= X + 1\\ Q_3(X) &= X^2 + X + 1\\ Q_4(X) &= X^2 + 1\\ Q_5(X) &= X^4 + X^3 + X^2 + X + 1\\ Q_6(X) &= X^2 - X + 1\\ Q_7(X) &= X^6 + X^5 + X^4 + X^3 + X^2 + X + 1\\ Q_8(X) &= X^4 + 1\\ Q_9(X) &= X^6 + X^3 + 1\end{aligned}$$

Cool property of cyclotomic polys:

$$X^n - 1 = \prod_{d|n} Q_d(X)$$

$$\begin{aligned}
X^1 - 1 &= Q_1(X) \\
X^2 - 1 &= Q_1(X)Q_2(X) \\
X^3 - 1 &= Q_1(X)Q_3(X) \\
X^4 - 1 &= Q_1(X)Q_2(X)Q_4(X) \\
X^5 - 1 &= Q_1(X)Q_5(X) \\
X^6 - 1 &= Q_1(X)Q_2(X)Q_3(X)Q_6(X)
\end{aligned}$$

Let $h(X)$ be an irreducible factor of $Q_r(X)$ when reducing coefficients modulo p. Since $o_r(p) > 1$, the degree of $h > 1$. The second group $\mathcal{G}$ is the nonzero elements of $P \mod h(X), p$. This group is generated by the set $\{X+1, X+2, \ldots, X+L\}$ and is a subgroup of the multiplicative group of $F = F_p[X]/h(X)$. So the polynomial X is an rth root of unity in F—that is, when you raise the polynomial X to the rth power, it's congruent to 1 modulo $h(X), p$.

3. Structure of remainder of proof

Lemma 4.7 says that if the algorithm outputs PRIME, then the size of $\mathcal{G}$ is exponentially larger than the size of G and gives a lower bound for the size, $\binom{t+L-2}{L-1}$.

Lemma 4.8 says there is an upper bound on the size. If n is a prime power, the upper bound is $n^{2\lfloor\sqrt{t}\rfloor}$; otherwise it's at most half that.

Lemma 4.9 says that for the size of L chosen, the lower bound is bigger than the upper bound for products of distinct primes, so n must be a prime power; but we checked all powers except 1 at the first step. So n is prime.

4. The details

4.1. **Lemma 4.7.** First thing is to show that polynomials in P of order less than t map to different elements of F. Proof by contradiction: pick two different polynomials f, g in P of order less than t such that $f(X)^m = g(X)^m$ in F. Now define $Q(Y) = f(Y) - g(Y)$. (Q here has nothing to do with cyclotomic polynomials–I guess Agrawal just likes the letter.) Y here is an element of the field F–a polynomial in X! That's OK, though, since we can treat a polynomial like a number, raising it to powers, multiplying by constants, adding two together. So it makes sense to talk about an unknown that is a polynomial instead of (say) a complex number. It also makes sense to talk about the order of this polynomial, the largest exponent on Y. Just as nth order polynomials over the complex numbers have at most n roots, nth order polynomials on any field have at most n roots.

Every element m of G is introspective for f, g—that is, t different values of m are introspective for f, g. Remember that "m introspective for $f \mod p$" means

$$f(X^m) = f(X)^m mod(X^r - 1), p$$

Here, we're doing things mod $h(X)$, not mod $X^r - 1$. But $h(X)$ is a divisor of $X^r - 1$, so it still works.

Remember also that we checked if r and n had any factors in common in step 3. They don't.

Finally, remember that X is a primitive rth root of unity in the field F, and any power that's coprime to r of a primitive root is also primitive. For example. i is a primitive 4th root of unity, so $i^3 = -i$ is a primitive 4th root since 3 is coprime to 4.

So because every m in G is coprime to r, and X is a primitive rth root, X^m is also a primitive rth root. Also, every m is less than r (the elements of G are the distinct elements of I modulo r), so the X^m are all different (no reductions by $(X^r - 1)$ to get the same value).

That means that $Q(Y) = f(Y) - g(Y) = 0$ for at least t distinct values $Y = X^m$. It is a polynomial with at least t roots. But that's impossible, since f and g have less than t roots!

The only thing we assumed was that two elements of P with less than t roots mapped to the same element of $\mathcal{G}$. So every element of P with less than t roots is a distinct element of $\mathcal{G}$.

Every polynomial g in $\mathcal{G}$ has the following form:

$$g(X) = \prod_{i=1}^{L} (X+i)^{e_i}.$$

We've got a list of L terms, namely $(X+1), (X+2), \ldots, (X+L)$. Each term gets an exponent; the sum of the exponents has to be less than t, i.e. at most $t-1$. If we multply the product by 1^{e_0}, then we can make the exponents sum to exactly $t-1$. So we have $t-1+L+1 = t+L$ symbols, of which $t-1$ are stars and the rest bars. By taking the bars as dividers, we count the number of stars in each division to get the exponents on the terms. So there are at least $\binom{t+L}{t-1}$ distinct polynomials of degree $< t$ in $\mathcal{G}$.

4.2. **Lemma 4.8.** The set $\hat{I}$ is a subset of I, where the exponents on n and p are between 0 and $\lfloor\sqrt{t}\rfloor$. That's $\lfloor\sqrt{t}\rfloor + 1 \times \lfloor\sqrt{t}\rfloor + 1$ numbers, which is greater than t. [It can't be equal because $\lfloor x\rfloor + 1$ isn't $\lceil x\rceil$.] The largest element is $(np)^{\lfloor\sqrt{t}\rfloor}$.

The polynomial $f(X)$ is any element of the group $\mathcal{G}$, which is a subgroup of the field $F = F_p[X] mod h(X)$. The polynomial $Q'(Y) = Y^{m_1} - Y^{m_2}$, where $m_1, m_2 \in \hat{I}$, has coefficients taken from that field, i.e. polynomials in X. By construction, the polynomial $f(X)$ is a root of $Q'(Y)$, since $f(X)^{m_1} = f(X)^{m_2}$. Because $f(X)$ can be any element of $\mathcal{G}$, $Q'(Y)$ must have at least $|\mathcal{G}|$ elements. Since $m1 > m2$, $Q'(Y)$ is of order m_1. So m_1 is at least as big as $|\mathcal{G}|$. But m_1 is taken from the set $\hat{I}$, so it's at most $(np)^{\lfloor\sqrt{t}\rfloor}$, the largest element.

If n is not p, then $n > p$ since p is a factor of n. The largest p could be is $n/2$, since we have to multiply it by some integer > 1 to get n. So m_1, which is at least as big as $|\mathcal{G}|$, is no bigger than $1/2n^{2\lfloor\sqrt{t}\rfloor}$.

4.3. **Lemma 4.9.**

$$|G| \geq \binom{t+L}{t-1} \tag{1}$$
$$\geq \binom{L+1+\lfloor 2\sqrt{t}\log n\rfloor}{\lfloor 2\sqrt{t}\log n\rfloor} \tag{2}$$

Proof.

$$\begin{array}{ccc} t > 2\sqrt{t}\log n & \text{means} & \sqrt{t}/2 > \log n \\ t > 4(\log n)^2 & \text{so} & \sqrt{t}/2 > (2logn)/2 = logn \end{array}$$

as desired, and $2\sqrt{t}\log n > 2(\log n)^2$.

$$|G| \geq \binom{2\lfloor 2\sqrt{t}\log n\rfloor + 1}{\lfloor 2\sqrt{t}\log n\rfloor} \tag{3}$$

Proof. $L = \lfloor 2\sqrt{\phi(r)}\log n\rfloor$. Since G is a subgroup of the multiplicative group mod r, $\phi(r) = |Z_r^*| > t = |G|$.

$$|G| > 2^{\lfloor 2\sqrt{t}\log n\rfloor} \tag{4}$$

Proof.

$$\begin{aligned}\binom{2x-1}{x} &= \frac{(2x-1)\cdot 2(x-1)\cdot(2x-3)\cdot 2(x-2)\cdots(x+1)}{(x)\cdot(x-1)\cdot(x-2)\cdots 1}\\ &= 2^{x-1}\frac{(2x-1)\cdot(2x-3)\cdot(2x-5)\cdots(x+1)}{x}\\ &> x \text{ for } x > 3\end{aligned}$$

$$|G| \geq 1/2\ n^{2\sqrt{t}} \text{ (log is base 2)}$$

By Lemma 4.8, $|G| < 1/2n^{2\sqrt{t}}$ if n is not a power of p, so $n = p^k$ for some $k > 0$. But we checked if n was the power of any integer in step 1, so $n = p$ if the algorithm outputs PRIME.