Corollaries of Fermat's Little Thm

We are in the process of looking at several formulas that are proven correct by using Fermat's Little Theorem that for prime p

$$b^p = b \bmod p \quad \text{(for all integers } b\text{)}$$

All these formulas are *useful* only if we have a way to evaluate $x^e \,\%\, m$ with large e.

Fortunately, the *fast modular exponentiation* algorithm provides this.

We almost *never* directly compute a large power of an integer without repeatedly reducing along the way. If you find yourself doing this, stop.

Multiplicative inverses by Fermat

Although the most direct means of computing multiplicative inverses modulo *general* moduli m is the extended Euclidean algorithm, simple exponentiation can be used to compute multiplicative inverses modulo *primes*.

That is, for p prime and for p not dividing b, the multiplicative inverse of b modulo p is

$$b^{-1} = b^{p-2} \bmod p$$

Indeed,

$$b \cdot b^{p-2} = b^{p-1} = 1 \bmod p$$

by Fermat's Little Theorem.

The run-time is very roughly comparable to extended Euclid.

Powers and roots mod m

Again, we say that a is an e^{th} **power modulo** m if there is an integer b such that

$$b^e = a \bmod m$$

In that case, say that b is an e^{th} **root of** a **modulo** m.

For $e = 2$ we talk about square roots and squares.

For $e = 3$ we talk about cube roots and cubes.

Principal square roots

Again, for p prime and $p = 3 \bmod 4$, *if* a is a square mod p then

$$\text{square root of } a \text{ modulo } p = a^{(p+1)/4}$$

Remark: If a is not a square modulo p this formula does not produce a square root of it modulo p. If you do not know in advance that a is a square mod p you *must* check.

Generally use *fast modular exponentiation* to compute this. *It is not explicit that the result be reduced modulo p, but it is desirable! Always reduce it!*

For square a mod p this formula gives the **principal square root** of a, distinguished not only by being the output of this formula, but also it itself is a *square* modulo p. The other square root $-b$ of a is *not* a square mod p.

Comments

Finding square roots modulo *composite* numbers n is essentially equivalent to being able to **factor** n, which is presumably hard for large n. We will see in just a little bit that if we have an **oracle** that takes square roots modulo $n = p \cdot q$ then we can factor n.

But finding square roots modulo primes is relatively easy.

But, even for primes, where would such formulas come from?

Why do we have a formula only for $p = 3 \bmod 4$? (There are more complicated square root formulas, and a general algorithm, for general primes.)

In fact, there is an easier formula for certain cube roots:

Cube roots: easy case

Another use of Fermat's Little Theorem.

Let p be a prime with $p \neq 1 \bmod 3$.

If we imagine that there is an exponent r such that b^r is a cube root of b modulo p, what should r be? We want

$$(b^r)^3 = b \bmod p$$

Superficially, we might want $r \cdot 3 = 1$, but this is impossible. But, from Fermat, for p not dividing b

$$b^{p-1} = 1 \bmod p$$

so if $r \cdot 3 = 1 + \ell(p-1)$ then

$$(b^r)^3 = b^{1+\ell(p-1)} = b \cdot (b^{p-1})^\ell$$

$$= b \cdot 1^\ell = b \bmod p$$

So for $r > 0$ a *multiplicative inverse* of 3 modulo $p-1$,

$$b^r = \text{cube root of } b \bmod p$$

Remark: This is simpler than the square root case we already treated, since in that square root case the simpler case would be that $p \neq 1 \bmod 2$, which doesn't happen.

Remark: No checking is *necessary* with this formula, unlike the square root formula.

Remark: With hindsight it is not surprising that there is a computationally accessible formula for cube roots modulo *primes*. But, as with square roots, if we can take cube roots modulo a composite number n then we can (probabilistically) **factor** n.

Remark: We *do* use the fact that 3 is prime, so that if 3 doesn't divide $p-1$ then $\gcd(3, p-1) = 1$, so the multiplicative inverse of 3 will exist modulo $p-1$.

Remark: Take a *positive* multiplicative inverse of 3 modulo $p-1$, or fast modular exponentiation won't apply.

Corollary: With prime p and 3 not dividing $p - 1$, every integer is a cube modulo p.

Proof: We have a formula which works!

///

Example: Find a cube root of 11 modulo 29.

First, by trial division 29 is a prime, and $29 = 2 \bmod 3$, so the formula applies. By brute force, $3 \cdot 19 = 57 = 1 \bmod (29 - 1)$, so

$$\text{cube root } 11 \bmod 29 = 11^{19} \,\%\, 29$$

The successive states in fast modular exponentiation are $(11, 19, 1)$, $(11, 18, 11)$, $(5, 9, 11)$, $(5, 8, 26)$, $(25, 4, 26)$, $(16, 2, 26)$, $(24, 1, 26)$, $(24, 0, 15)$. The last value of the third component, 15, is the desired result. Thus, 15 is a cube root of 11 modulo 29.

(Check? Yes, $15^3 \,\%\, 29 = 11$.)

Prime-order roots, easy case

The same discussion applies replacing 3 and cube roots by *prime* e and e^{th} roots:

For p a prime with $p-1$ *not* divisible by e, let r be the multiplicative inverse of e modulo $p-1$. Then any a is an e^{th} power modulo p, and an e^{th} root of a is given by the formula

$$e^{\text{th}} \text{ root of } a \text{ modulo } p = a^r$$

Proof: Let $er = 1 + \ell(p-1)$. Then

$$(a^r)^e = a^{er} = a^{1+\ell(p-1)}$$

$$= a \cdot (a^{p-1})^\ell = a \cdot 1^\ell = a \bmod p$$

by Fermat's Little Theorem. ///

Prime-order roots, harder case

Now a family of cases subsuming the earlier square root formula for prime $p = 3 \bmod 4$.

Theorem: Let e be a prime. Let p be a prime such that $e|(p-1)$ but e^2 does *not* divide $p-1$. That is, e does not divide the integer $(p-1)/e$. Let r be a multiplicative inverse of e modulo $(p-1)/e$. *If* a is an e^{th} power modulo p, then

$$e^{\text{th}} \text{ root of } a \text{ modulo } p = a^r$$

and this is the only one of the e^{th} roots of a which is itself an e^{th} root.

Remark: If a is *not* an e^{th} power mod p then this formula does *not* produce an e^{th} root. Thus, if you do not know in advance that a is an e^{th} power modulo p, then you *must check* whether or not $b = a^r \,\%\, p$ is an e^{th} root. That is, compute $b^e \,\%\, p$ and see if you get a.

RSA setup

One-time preparation: *Alice* chooses two large random primes p, q, from 10^{100} to 10^{600} depending on the desired security. She computes the **RSA modulus** $n = p \cdot q$. She chooses **encryption exponent** e and computes the **decryption exponent** d which is the multiplicative inverse of e modulo $(p-1)(q-1)$. She publishes n, e and keeps d secret. Primes p and q can be thrown away.

Encryption: To encrypt a *plaintext* message x and send it to Alice on an insecure channel. (Suppose $1 < x < n$) Bob computes and transmits $y = x^e \,\%\, n$.

Decryption: When Alice receives the ciphertext y, she computes $y^d \,\%\, n$, which *is* the plaintext x.

Some details

Alice does not find two primes $\sim 10^{200}$ by trial division.

Alice computes e^{-1} mod $(p-1)(q-1)$ by extended Euclid.

Bob computes $x^e \,\%\, n$ by fast modular exponentiation.

Alice to compute $y^d \,\%\, n$ by fast modular exponentiation

$y^d \,\%\, n = x$ because of Euler's Theorem (an extension of Fermat's Little Theorem).

Apparently, though not *provably*, an eavesdropper Eve cannot factor n into $p \cdot q$, and cannot take e^{th} roots modulo the composite number n, so cannot decrypt.

Finding big primes

To acquire **200-digit prime numbers** trial division would not succeed in the lifetime of the universe using all the computational power of the internet, etc.

Trial division confirms that a number is prime by **failing to factor it**.

It turns out that **primality testing is much easier than factoring**.

Primality testing does not try to factor numbers.

Factoring big numbers is hard, despite modern factorization techniques much better than trial division.

The number of primes less than x is roughly $x/\ln x$ (the *natural* log of x).

It is not true that primes are uniformly distributed, but in hunting for primes we pretend so in order to form first-approximation expectations.

If primes *were* evenly distributed (which they're *not*) then near x primes would be about $\ln x$ apart.

Thus, in hunting for primes near x expect to examine $\frac{1}{2}\ln x$ candidates:

For $x \sim 10^{20}$ we have $\frac{1}{2}\ln x \sim 23$

For $x \sim 10^{100}$ we have $\frac{1}{2}\ln x \sim 115$

For $x \sim 10^{500}$ we have $\frac{1}{2}\ln x \sim 575$

So if each test takes only a little time we should expect to quickly find a prime wherever we look.

Fermat pseudoprimes

One more time: **Fermat's Little theorem:** If p is prime, then for any integer b we have $b^p \,\%\, p = b$.

Contrapositive: For integer n if $b^n \,\%\, n \neq b$ for some b then n is **composite**.

The *converse* is false, but not *very* false...

Thus, we have

Converse-with-disclaimer: If $b^p \,\%\, p = b$ then p is *fairly likely* to be prime, but may not be.

n is a **Fermat pseudoprime base** b if $b^n = b \bmod n$.

Remark:

Usage of the phrase *pseudoprime* is not consistent.

My usage is that a number that has passed a primality test (Fermat, Miller-Rabin, etc.) is a **pseudoprime**.

Sometimes *pseudoprime* is meant to imply *non-prime*, though having passed a primality test such as Fermat. But for large numbers which have passed pseudoprimality tests we may never know *for sure* whether or not they're prime or composite ...

Another usage is to call a number that has passed a test a **probable prime**.

But this term is dangerously close to **provable prime**, which is sometimes used to describe primes with **certificates** of primality.

There are only 172 non-prime Fermat pseudoprimes base 2 under 500,000 versus 41,538 primes, a false positive rate of less than 0.41%

There are only 49 non-prime Fermat pseudoprimes base 2 and 3 under 500,000, a false positive rate of less than 0.118%

There are only 32 non-prime Fermat pseudoprimes base 2, 3, 5 under 500,000

There are still 32 non-prime Fermat pseudoprimes base 2, 3, 5, 7, 11, 13, 17 under 500,000

561 1105 1729 2465 2821 6601 8911 10585 15841 29341 41041 46657 52633 62745 63973 75361 101101 115921 126217 162401 172081 188461 252601 278545 294409 314821 334153 340561 399001 410041 449065 488881

n is a **Carmichael number** if it is a *non-prime* Fermat pseudoprime to *every* base b.

In 1994 Alford, Granville, and Pomerance showed that there are infinitely-many Carmichael numbers.

And it appears that among *large* numbers Carmichael numbers become more common.

Nevertheless, the Fermat test is a very fast way to test for *compositeness*, and is so easy and cheap that it is still the best first approximation to primality.

It is **cheap** because $b^n \,\%\, n$ can be computed in $\sim \log n$ steps, not n...

For example, use the Fermat pseudoprime test base 2 to test 15. We compute (by fast modular exponentiation): successive states are $(2, 15, 1)$, $(2, 14, 2)$, $(4, 7, 2)$, $(4, 6, 8)$, $(1, 3, 8)$, $(1, 2, 8)$, $(1, 1, 8)$, $(1, 0, 8)$. Thus, $2^{15} \,\%\, 15 = 8 \neq 2$, so we have *proven* that 15 is composite.

Using the Fermat pseudoprime test base 7 to test 25, we have successive states $(7, 25, 1)$, $(7, 24, 7)$, $(24, 12, 7)$, $(1, 6, 7)$, $(1, 3, 7)$, $(1, 2, 7)$, $(1, 1, 7)$, $(1, 0, 7)$. Thus, $7^{25} \,\%\, 25 = 7$. Thus 25 is a Fermat pseudoprime base 7.

But by trial division $25 = 5^2$ is *not* a prime.