CLASS 14, GIVEN ON 10/22/2010, FOR MATH 25

1. Fast exponentiation mod n

Today we will discuss how one might use Fermat's Little Theorem and some variations on it to test whether a number n is prime or composite. Recall that FLT says that if p is a prime, and $p \nmid a$, then $a^{p-1} \equiv 1 \mod p$. So if n is an integer, and we happen to find a number a such that $n \nmid a$ and $a^{n-1} \not\equiv 1 \mod n$, then n must be composite. We call this a *compositeness test*, since if we find that $a^{n-1} \not\equiv 1 \mod n$, then n must be composite. If we find $a^{n-1} \equiv 1 \mod n$, then we cannot conclude anything about n – we will see that n might be prime, or n might be composite. In any case, we will come back to this point later.

For this to actually be a useful test, we want to know that the calculation of $a^{n-1} \mod n$ can be carried out reasonably rapidly. For instance, perhaps n is a 100-digit number; this is beyond the reach of trial division, so $n \approx 10^{100} \approx 2^{300}$. Trial division would take something on the order of $\sqrt{n} \approx 2^{150}$ possible trial divisions, which is a gigantic number.

So suppose we want to calculate $a^{n-1} \mod n$, for some $1 < a < n$. Clearly $n \nmid a$. The most naive approach to calculating this congruence class is to multiply a by itself $n-2$ times, which would still take far too many operations to be of use. But there is a way to drastically cut back on the number of multiplications which one needs to carry out, using an idea called *successive squaring.*

The idea is simple. Suppose we want to calculate something like a^{16}. Instead of multiplying a by itself 15 times, we can instead calculate a^2, then square that to obtain a^4, then square that to get a^8, and finally square that to get a^{16}. This takes a total of $4 = \log_2 16$ operations instead of 15. More generally, if we want to calculate $a^k \mod n$, we begin by writing k in base 2 (binary). For example, suppose

$$k = a_r 2^r + a_{r-1} 2^{r-1} + \ldots + a_1 2 + a_0,$$

where each $a_i = 0$ or 1. To compute $a^k \mod n$, we first calculate each of the r powers $a^2 \mod n, a^4 \mod n, a^8 \mod n, \ldots, a^{2^r} \mod n$. This takes a total of r multiplications mod n. To compute $a^k \mod n$, we simply multiply together the appropriate powers of a. For instance, if $k = 2^4 + 2^2 + 1$, we would multiply $a^{2^4}, a^{2^2}, a^{2^1}$ together to calculate $a^k \mod n$. There are a maximum of r multiplications here as well, so to calculate $a^k \mod n$ required a total of at most $2r$ multiplications mod n, and also a little bit of storage space to store the 2^ith powers of a. Now notice that $r \leq \log_2 k + 1$, so even if k is of size 2^{300}, we still have $r \leq 301$, so that we need no more than about 600 multiplications mod n to compute $a^k \mod n$. This is clearly a huge gain over the naive approach of just multiplying a by itself $n-2$ times!

Example. Compute the least positive residue of $3^{196} \mod 263$. We first write the exponent in binary: $196 = 2^7 + 2^6 + 2^2$. We now compute the 2^ith powers of $3 \mod 263$, either by hand or with the assistance of a calculator:

$$3^2 \equiv 9 \mod 263, 3^4 \equiv 81 \mod 263, 3^8 \equiv 81^2 \equiv 249 \mod 263, 3^{16} \equiv 249^2 \equiv 196 \mod 263,$$
$$3^{32} \equiv 196^2 \equiv 18 \mod 263, 3^{64} \equiv 18^2 \equiv 61 \mod 263, 3^{128} \equiv 61^2 \equiv 39 \mod 263.$$

We have $3^{196} = 3^{128} \cdot 3^{64} \cdot 3^4$, so $3^{196} \equiv 39 \cdot 61 \cdot 81 \equiv 183 \mod 263$.

The main point behind this method is that it is computationally very feasible, using modern computers, to calculate $a^k \mod n$, even if k (and n) are numbers with hundreds, thousands, or even tens of thousands of digits. Also notice that in computers, we already get the binary expansion of k for free, since computers store numbers in binary. In summary, we have the following algorithm for computing $a^k \mod n$:

Algorithm 1 (Fast exponentiation mod n)**.** *Let $a, k, n > 0$. To compute $a^k \mod n$ using at most $2\log_2 k$ multiplications $\mod n$, first write k in binary. Then compute $a, a^2, a^{2^2}, a^{2^3}, \ldots, a^{2^r} \mod n$, where r is an integer such that $2^r \leq k < 2^{r+1}$. Finally, compute $a^k \mod n$ by multiplying the appropriate powers of a already computed, using the binary expansion of k to tell us which powers to multiply together.*

2. Fermat's Little Theorem as a compositeness test

Let us return to the question of testing whether n is prime or composite. As we said earlier, if there is an a such that $1 < a < n$, and $a^{n-1} \not\equiv 1 \mod n$, then n must be composite. This yields the following algorithm:

Algorithm 2 (Fermat compositeness test)**.** *Let n be a positive integer we which to test for compositeness, and let a be any integer with $1 < a < n$. If $a^{n-1} \not\equiv 1 \mod n$, then n is composite. If $a^{n-1} \equiv 1 \mod n$, the test is inconclusive.*

Example. Let $a = 2, n = 2257$. (One could use trial division to test n, but let's use FLT as a compositeness test.) One calculates that $2^{2256} \equiv 2193 \mod 2257$, which is clearly not $1 \mod 2257$, so 2257 is composite. Notice that while this test conclusively tells us that n is composite, it does not actually provide a factor of n.

On the other hand, the next example shows that sometimes composite numbers can slip through this test.

Example. Let $a = 2, n = 341$. Then $2^{340} \equiv 1 \mod 341$, but $341 = 11 \cdot 31$.

This example leads to the following definition:

Definition 1. *Let $a > 1$ be a fixed positive integer. We say that a composite number n is a* (Fermat) pseudoprime to the base *a if $a^{n-1} \equiv 1 \mod n$.*

In other words, a pseudoprime to the base a is mimicking the behavior of a prime number with respect to the FLT compositeness test, despite the fact that it is not prime. The previous example shows that 341 is a pseudoprime to the base 2.

To answer the question of how reliable the Fermat test is as a compositeness test requires us to be able to answer questions like how many psuedoprimes to base a there are. This is beyond the scope of this class, but suffice it to say that in practice the above test usually works quite well, in the sense that given a 'random' composite integer n, and a 'random' base a, $a^{n-1} \not\equiv 1 \mod n$ is 'usually' true.

However, are there any cases of composite numbers n in which it is practically impossible to use the Fermat compositeness test to determine whether n is actually composite? The answer is yes!

Definition 2. *Let n be a positive composite integer. We say that n is a* Carmichael numbers *if $a^n \equiv a \mod n$ for all a.*

A Carmichael number is thus a number for which the Fermat compositeness test is inconclusive for practically every value of a. It certainly is inconclusive if $\gcd(a, n) = 1$, because then we know that $a^{n-1} \equiv 1 \mod n$. If $\gcd(a, n) > 1$, then $a^{n-1} \equiv 1 \mod n$ is impossible, so we would be able to detect that n is composite eventually if we kept trying different choices of a. As a practical matter, however, this is just as slow as trial division (if not slower), since we would have to run across an a which, if not a divisor of n, at least had nontrivial gcd with n, which for instance would be very unlikely if n were a product of two large primes.

Example. We will show that 561 is a Carmichael number. First, notice that $3|561$, so 561 is composite. We want to show that $a^{561} \equiv a \mod 561$ for any a.

First, we factor $561 = 3 \cdot 11 \cdot 17$. Proving that $a^{561} \equiv a \mod 561$ is equivalent to proving that $a^{561} \equiv a \mod 3, a^{561} \equiv a \mod 11, a^{561} \equiv a \mod 17$.

To show that $a^{561} \equiv a \mod 3$, first notice that if $3|a$ then this is automatically satisfied. So suppose $3 \nmid a$; then FLT says that $a^2 \equiv 1 \mod 3$, or $a^{560} \equiv 1 \mod 3$. But then $a^{561} \equiv a \mod 3$, as desired. Similarly, for $a^{561} \equiv a \mod 11$, if $11 \nmid a$, then $a^{10} \equiv 1 \mod 561$, or $a^{560} \equiv 1 \mod 561$. And for $a^{561} \equiv a \mod 17$, if $17 \nmid a$, then $a^{16} \equiv 1 \mod 561$. Since $560 = 16 \cdot 35$, again we have $a^{560} \equiv 1 \mod 17$. So 561 really is a Carmichael number.

The existence of Carmichael numbers does not augur well for using the Fermat test as a definitive test for compositeness. After all, suppose you wanted to test a number n for compositeness, and started computing $a^{n-1} \mod n$ for various random values of a. Suppose you kept getting $1 \mod n$ back. What can you conclude? Well, as a definitive matter, you can conclude nothing. After all, maybe n is prime – then you would expect to keep getting $1 \mod n$. But maybe n is a Carmichael number, and you haven't been lucky enough to pick a with $\gcd(a, n) > 1$. (Or maybe n isn't even a Carmichael number, and you just happened to pick a such that $a^{n-1} \equiv 1 \mod n$.)

The question of, say, what the probability that the Fermat test will return "composite" if you feed it a random composite number n is a very good and interesting one, but also is one that is beyond the scope of this class. Another interesting question is how many Carmichael numbers there are. As a matter of fact, it is a difficult theorem of Alford, Granville, and Pomerance in 1992 that there are infinitely many Carmichael numbers, and they show that there are at least $x^{2/7}$ Carmichael numbers of size up to x if x is large.

So the Fermat compositeness test works 'most of the time', but if we keep getting inconclusive results, we cannot conclude anything about whether the number n we are testing is prime or not. In particular, since Carmichael numbers exist, to conclusively rule out a number as prime using the Fermat test would require at least as much work as trial division, since we would have to choose a with $\gcd(a, n) > 1$.

Is there a way to slightly modify the Fermat test so that numbers like Carmichael numbers, which fool the test almost all the time, do not exist anymore? The answer is yes!

3. Strong psuedoprimes and the Miller-Rabin test

The following theorem provides a slight modification of the Fermat compositeness test:

Theorem 1. *Let p be an odd prime, and let $p - 1 = 2^e m$, where m is odd. (So 2^e is the largest power of 2 dividing $p - 1$.) Let a be an integer such that $p \nmid a$. Then one of the following is true:*

- *Either $a^m \equiv 1 \mod p$, or*
- *$a^{2^r m} \equiv -1 \mod p$ for some r satisfying $0 \leq r < e$.*

Proof. We use the two basic facts that $a^{p-1} \equiv 1 \mod p$ (Fermat's Little Theorem), and that the equation $x^2 \equiv 1 \mod p$ has the two solutions $x \equiv \pm 1 \mod p$.

We know that $a^{2^e m} \equiv 1 \mod p$, by Fermat's Little Theorem. Because $x^2 \equiv 1 \mod p$ has the two solutions $x \equiv \pm 1 \mod p$, and $(a^{2^{e-1}m})^2 = a^{2^e m}$, we must have $a^{2^{e-1}m} \mod \pm 1 \mod p$. If this number is $\equiv -1 \mod p$, we are done; if not, just repeat this argument with 2^{e-1} in place of 2^e. If we find that $a^{2^{e-2}m} \equiv -1 \mod p$, we are done, if not, repeat this argument with 2^{e-2} in place of 2^{e-1}. We continue in this way until we either find some r with $a^{2^r m} \equiv -1 \mod p$, or we eventually end up with $a^m \equiv 1 \mod p$. □

We can use this theorem as a compositeness test. Suppose we want to test an odd number n for compositeness. Then we write $n - 1 = 2^e m$, for some odd m, $e > 0$. We then calculate $a^m \mod n$, which is quick to do. We then compute $a^{2^r m} \mod n$, which means we just repeatedly square $a^m \mod n$, until we reach $a^{2^e m} \mod n$. We look at this list and check whether the conclusion of the above theorem is true: that is, whether one of these numbers is $\equiv -1 \mod n$, and if not, whether $a^m \equiv 1 \mod n$ is true. If no $a^{2^r m} \equiv -1 \mod n$, and also $a^m \not\equiv 1 \mod n$, then we know that n is composite. If, on the other hand, this is true, we cannot conclude anything about the primality or compositeness of n. Call a composite number n which passes this test for a given number a a *strong pseudoprime to the base a*.

Examples.

- First, notice that any composite n which is a strong psuedoprime to base a is also a psuedoprime to base a. Indeed, if a, n pass the strong psuedoprime test, either $a^m \equiv 1 \mod n$ is true, in which case $a^{2^e m} \equiv 1 \mod n$ is also true (we just square 1 e times), or $a^{2^r m} \equiv -1 \mod n$ is true, in which case $a^{2^{r+1}m} \equiv 1 \mod n$, and hence $a^{n-1} \equiv 1 \mod n$ is true (notice that we are using the fact that $r < e$ here).
- Recall $n = 341$ is a psuedoprime to base 2. Let's see if it's a strong pseudoprime. First, we start with $n - 1 = 340$, and factor out as many powers of 2 as possible. We see that $340 = 2^2 \cdot 85$, so $e = 2, m = 85$. We now compute $2^{85} \mod 341$, and then square this repeatedly until we reach $2^{340} \mod 341$. Either by hand or using a calculator, we find
$$2^{85} \equiv 32 \mod 341, 2^{170} \equiv 1 \mod 341, 2^{340} \equiv 1 \mod 341.$$
Since $2^{85} \not\equiv 1 \mod 341$, and $2^{85}, 2^{170} \not\equiv -1 \mod 341$, then 341 is not a strong pseudoprime to base 2, and hence is composite. So the modified version of Fermat's compositeness test catches composite numbers which the original compositeness test could not, a great sign!
- This isn't an example so much as a remark, but notice that the modified Fermat compositeness test is well-suited to run on a computer. For instance, because integers are represented in binary on a computer, to find the largest power of 2 dividing $n-1$ we just need to look at how many 0s are at the tail end of the binary representation of $n-1$. We can quickly calculate $a^m \mod n$, using successive squaring, and then we only need to square this a few more times to fully implement the modified test.

The main theoretical question now is to determine whether a situation like Carmichael numbers occurs for the modified Fermat compositeness test. Let n be a composite integer, and let $S(n)$ be the number of integers a, with $1 < a < n$, such that n is a strong psuedoprime to the base a. The following theorem is true, which is far too difficult for us to prove in this class:

Theorem 2. *For n a composite, odd integer > 9, one has $S(n) \leq n/4$.*

Another way of saying this is that if we are testing a number n for compositeness, and it happens to be composite, if we randomly choose a base a from 1 to n, there is at least a $3/4$ chance that n will be shown to be composite using the strong psuedoprime test. This theorem is the basis for what is known as the *probabilistic Miller-Rabin test*: to test a number n for compositeness, you randomly select various bases a, with $1 < a < n$, say k times. If any one of these tests returns a result of composite, you know n is composite. If n is composite but all the k tests were inconclusive, then there is at most a $1/4^k$ chance you are wrong if you claim that n is prime. Since this test is fast to run (it only involves a binary bit check and exponentiation mod n), you can probably afford to run the test a few hundred times, which will make the probability of error very small.

What if we want to use this test to definitively check whether n is composite or prime? Suppose n is composite. Since there are at most $n/4$ bases a for which the modified Fermat test is inconclusive, if we test $a = 2, 3, \ldots, n/4 + 1$, at least one of these will detect n as composite. Of course, this is an intolerably large number of cases to check.

However, it has been shown that if something known as the *generalized Riemann Hypothesis* (or rather, a special case of the GRH) is true, then as a matter of fact there is some a which will detect n as composite using the modified Fermat test satisfying $a < 2\log^2 n$. Since $2\log^2 n$ is a lot smaller than $n/4$ (if $n \approx 10^{100}, 2\log^2 n \approx 80000$, while $n/4 \approx 10^{99}$), if the GRH is true then in reality we need only test a relatively small number of a before we are guaranteed to know that n is prime or composite. Of course, one should always entertain the possibility, however unlikely, that the GRH is false.

How does this all work in practice? If you want to test whether n is prime or composite, run the probabilistic Miller-Rabin test however many times you want to get the error below a certain threshold. If you find n is composite, great; if your multiple runs are inconclusive, then it is very likely that your number is prime. At this point, you can either just assume your number is prime (which might be hazardous if your application is mission-critical, but on the other hand if the probability of error is really small, that error will be dwarfed by the possibility of other things going wrong, like your computer malfunctioning during calculations or the sun exploding), or actually prove that your number is prime using more sophisticated primality testing algorithms which we will not have time to talk about in this class.