MATH 25 CLASS 9 NOTES, OCT 10 2011

Contents

1. A short introduction to primality testing and factorization 1
2. Generating lists of prime numbers: the sieve of Eratosthenes 4

1. A short introduction to primality testing and factorization

When studying primality and factorization, perhaps the two most natural computational questions are the following:

- Given a positive integer n, how can we determine if n is prime or not? How fast can we do this?
- Given a positive integer n, how can we find the factorization of n? How fast can we do this?

The second question can be reformulated as "Given a positive integer n, how can we find a nontrivial factor of n or show that none exists?", since if we have an algorithm answering this question, the algorithm can be applied repeatedly until we end up with a prime factorization.

The first question is called *primality testing*, and in computer science is sometimes called PRIMES, while the second question is called *integer factorization*. Notice that integer factorization is at least as hard as primality testing, because if you know the factorization of an integer, then you immediately know whether that number is prime or not.

It is a fact of current mathematical knowledge that, as of right now, primality testing is much easier than factorization. This may sound somewhat strange, but makes some sense once one learns that all fast primality tests are unable to reliably exhibit any actual factors of n if they discover that n is not prime. A lot of modern research in mathematics and computer science centers around these two questions, because primality testing and integer factorization have turned out to be of fundamental importance in cryptography. Later in this class we will take a look at the RSA algorithm, which is one currently used method of encryption used to secure information transfer over the Internet.

In any case, we can definitely find *some* algorithm to answer the two questions above. After all, to find the factorization of an integer n, just start dividing it by successively larger integers d, starting with $d = 2$, until one finds $d \mid n$. This has to eventually happen, since $n \mid n$. This method of finding factors, where one simply tries to divide n by successively larger integers, is known as *trial division*.

As a matter of fact, the first number d which divides n must be a prime. For if not, say $p \mid d$, then $p \mid n$, and we would have discovered that p divided n prior to d dividing n, contradicting the fact that d is the smallest nontrivial divisor of n. So trial division will provide us with the smallest prime factor, say p_1, of n. One then applies trial division to n/p_1, to find a prime factor p_2; it may be the case that $p_1 = p_2$, but in any case one then applies trial division to $n/(p_1p_2)$. Eventually one has to have $n = p_1p_2 \dots p_r$, for not necessarily distinct primes p_i, and this is the factorization of n.

Of course, trial division also works as a primality test. If one finds that $p|n$ for some prime $p < n$, then n cannot be prime, and if no such p divides n, then n is prime. This formulation of trial division seems to need about n divisions in the worse case, since if n is indeed a prime, one needs to test all the numbers from 2 to $n-1$ as factors of n.

However, there is a lot of extraneous computation in this naive version of trial division. The following proposition shows that we really only need about $\sqrt{n}$ divisions in the worse case:

Proposition 1 (Lemma 2.14 of text)**.** *If n is composite, then n has a nontrivial factor d such that $d \leq \sqrt{n}$.*

Proof. Since n is composite, write $n = ab$ with $1 < a, b < n$. Then one of $a, b \leq \sqrt{n}$, for if not, then $ab > \sqrt{n}^2 = n$, which contradicts the fact that $n = ab$. □

This proposition tells us that we only need to trial divide by d up to $\sqrt{n}$ to determine whether n is prime (or to find a factor of n). For instance, if $n = 101$ (which turns out to be prime), we need to only check n for divisibility by $2, 3, 4, \dots, 10$ to either find a factor or conclude that n is prime. Try actually testing for divisibility by these numbers.

In this example, you probably realized that even doing trial divisions by $2, 3, 4, \dots, 10$ has extraneous computation. After all, if $4|n$, then $2|n$. And we've already said that the smallest nontrivial factor, if one exists, of a number n must be a prime. So in reality we need only test n for divisibility by primes $p \leq \sqrt{n}$. In the case of $n = 101$, this means we need only test for divisibility by $2, 3, 5, 7$. So, for example, $2 \nmid 101$ since 101 is not even, $3 \nmid 101$ since $3 \nmid 2$, and 2 is the sum of the digits of 101, $5 \nmid 101$ since 101 does not end in $5, 0$, and $7 \nmid 101$ since one checks that $7 \cdot 14 = 98, 7 \cdot 15 = 105$. Hence 101 is prime.

So let's summarize what's happened so far. In our first, very crude version of trial division, in the worst case we needed about n trial divisions to prove a number prime. A simple observation allowed us to lower that number to $\sqrt{n}$ trial divisions. And as a matter of fact we need only check for divisibility by primes up to $\sqrt{n}$. Let's think about how much of a saving that gets us.

To determine how many divisions we need to do in the worse case scenario with this version of trial division, we need to know how many prime numbers there are less than $\sqrt{n}$; that is, we want an estimate for $\pi(\sqrt{n})$. If you believe the prime number theorem (which is true), then

$$\pi(\sqrt{n}) \approx \frac{\sqrt{n}}{\log \sqrt{n}} = \frac{2\sqrt{n}}{\log n}.$$

So evidently we get a savings by a factor of $\log n$ when we only do trial division by primes up to $\sqrt{n}$, instead of all integers up to $\sqrt{n}$. Unfortunately, $\log n = o(x^\delta)$ for any $\delta > 0$, so for large n this saving is rather small compared to the total amount of computation required.

However, there is one substantial practical problem to using this version of trial division. For instance, suppose I asked you to test a six digit number, say of size around 10^6, for primality. We know that we need to only test divisibility by primes up to about 1000. But what are the primes up to 1000? Evidently you need a precomputed list of all the primes up to $\sqrt{n}$ if you want to carry this version of trial division out, and this requires some amount of computer time to generate. If you ask a computer to use trial division to test a number of size 10^{20}, say, one would need to store something like 10^9 or 10^{10} primes in memory. So perhaps it is not practical to only try divisibility by primes for moderately large n (of course, the meaning of 'moderately' depends on the context).

One can compromise between the version of trial division which tests all numbers up to $\sqrt{n}$ vs primes up to $\sqrt{n}$. For instance, it is obvious that even numbers bigger than 2 are not prime, so we can just skip testing for divisibility by these numbers. This gives a savings by a factor of 2, since half of all numbers are even. In practice, it is easy to recognize even numbers, since we need only look at the last digit, and for computers it is especially easy since they store numbers as base 2 and only need to check whether the last digit is 0 or 1. If one were slightly more ambitious, one could also not test divisibility by numbers which are multiples of other small primes, like 3 or 5, and get more savings.

In practice, given current computer speeds, a personal computer can test a number for primality/factors using trial division in a few seconds for numbers of size perhaps of order 10^{15} or so (this might be off slightly, and will depend on how new the computer being used is).

There have been much more sophisticated primality and factorization methods developed, many in the past half century, which perform much better, both theoretically and in practice, than trial division. One feature is that many of these primality tests cannot actually return factors if they tell you that a number is composite. For a long time, it was believed that primality testing had a polynomial-time algorithm which would correctly determine whether a number is composite or prime(that is, an algorithm whose worse-case run time was a polynomial in the number of digits of n). In 2002, the team of Agrawal, Kayal, and Saxena exhibited such an algorithm. Amazingly enough, the algorithm used elementary ideas, and maybe even more amazing, Kayal and Saxena were undergraduates when they did this work!

It might be worth pointing out that in practice, the AKS algorithm is not what is actually used to test for primality on modern computer systems. There are other algorithms which might run in polynomial time: they do indeed run in polynomial time if certain unproved conjectures in mathematics (for example, the proof of the correctness of the polynomial-time deterministic version of the 'Miller-Rabin' test is based on a generalization of the Riemann Hypothesis) are assumed to be true. There are also other algorithms which are *probabilistic*: these algorithms may sometimes mistakenly tell you that a composite number is actually prime. Why would one bother with these tests? The most obvious reason is that they are much faster than even the best *deterministic* tests – even the ones conditional on unproven statements. (Deterministic means that a test definitely tells you whether a number is prime or composite, with no possibility of error, as opposed to a probabilistic test.) Another reason is that the probability of error can be made very small. For instance, the Miller-Rabin test (the probabilistic version) has error rate about $(1/4)^n$, where n is the number of runs of the test, and if the test is fast it is not harmful to run the test 100 or 200 times. And in practice, one is easily willing to accept an error of order 10^{-50}, say, since it is much more likely that something totally crazy, like asteroids hitting the Earth, widespread power failures, or catastrophic computer hardware error, will occur!

2. Generating lists of prime numbers: the sieve of Eratosthenes

Suppose you want to find all the prime numbers between 1 and N. One could test each number between 2 to N using your favorite method, such as trial division. But this a lot of work; after all, if $N = 100$, say, and you were doing this by hand, you would have to trial divide large two-digit numbers, which doesn't sound like a lot of fun. Furthermore, you do some redundant work: for instance, you test every odd number for divisibility by 2, which you already know isn't going to happen, and you test every multiple of 3 which is not even for divisibility by 3, even though you know that isn't going to happen.

The following simple systematic method for tabulating all the primes up to N was developed by the ancient Greek Eratosthenes. One starts with a list of the integers from 2 to N. Look at the smallest number not yet crossed off in the list; right now this number is 2. Cross off every number which is a multiple of 2, except 2 itself. Every number we cross off obviously is not prime, since it is a multiple of the prime 2. Once we have eliminated every multiple of 2 from our list, we look at the smallest number not yet crossed off and not known to be prime yet, which is $p = 3$. This number must be prime, since it is not a multiple of any number smaller than it (except 1). We cross off every proper multiple of 3 from our list. We continue like this, at each step looking at the smallest number not yet crossed off and not known to be prime. This number must be a prime p, since it is not divisible by any prime than p (since it has yet to be crossed off). We cross off every proper multiple of p in our list.

When does this algorithm end? It definitely ends when we reach a point where there are no more non-crossed off numbers on our list. But it actually ends earlier than this. Notice that we can stop once have eliminated multiples of a prime p for all $p < \sqrt{N}$, since every number between 1 and N must be divisible by some prime $\leq \sqrt{N}$.

Each step of this algorithm, where we eliminate proper multiples of a prime p, is sometimes called 'sieving' the multiples of p. The reasoning behind this name is clear; just like a sieve in real life, we are eliminating numbers which satisfy some property. Here is a quick sampler of the sieve of Eratosthenes applied to $N = 30$. We start with a list of all the numbers from 2 to 30:

$$2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30.$$

Eliminate all proper multiples of 2. We will also underline the number 2 to indicate that it is a prime number which we have sieved by already:

$$\underline{2}, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29.$$

The smallest number on this list not known to be prime yet is 3, so we know 3 is prime. Eliminate all proper multiples of 3 that are on this list:

$$\underline{2}, \underline{3}, 5, 7, 11, 13, 17, 19, 23, 25, 29.$$

Now $p = 5$ is a prime, and we sieve by 5:

$$\underline{2}, \underline{3}, \underline{5}, 7, 11, 13, 17, 19, 23, 29.$$

Since $7 > \sqrt{30}$, we have sieved by every prime less than $\sqrt{30}$, so whatever is left is the list of all primes between 2 and 30.