Lecture Notes on Primality Testing

May 8, 2007

There are many situations in which the primality of a given integer must be determined. For example, fingerprinting requires a supply of prime numbers, as does the RSA cryptosystem (where the primes should typically have hundreds of bits).

A theoretical breakthrough in 2002, due to Agrawal, Kayal and Saxena has given us a deterministic polynomial time algorithm for primality testing. However, in practice randomized algorithms are more efficient and continue to be used. These algorithms date back to the 1970's and caused a surge in the study of applied number theory.

A Simple Deterministic Algorithm

Given an odd integer n, we wish to determine whether n is prime or composite. Consider the following deterministic algorithm:

for $a = 2, 3, ..., \lfloor\sqrt{n}\rfloor$ **do**
    if $a|n$
    then output "composite" and halt
output "prime"

This algorithm is obviously correct. However, because the for-loop has $O(\sqrt{n})$ iterations, the algorithm does not have running time polynomial in the number of input bits. (Consider the case where n is an integer with hundreds or thousands of bits; then $\sqrt{n}$ is an enormous number!) Other, more sophisticated algorithms based on prime number sieves are a bit more efficient but still suffer from a similar drawback.

Randomized Algorithms

Our first randomized algorithm is based on the following standard theorem:

Fermat's Little Theorem: *If p is prime, then $a^{p-1} = 1 \bmod p$ for all $a \in \{1, ..., p-1\}$.*

In particular, for a given integer n, if there exists an $a \in \{1, ..., n-1\}$ such that $a^{n-1} \neq 1 \bmod n$, then surely n is composite. This fact suggests the following algorithm, known as "Fermat's Test":

pick $a \in \{1, ..., n-1\}$ uniformly at random
if $\gcd(a, n) \neq 1$
then output "composite" and halt
else if $a^{n-1} \neq 1 \bmod n$
then output "composite"
else output "prime"

Computing $\gcd(a, n)$ can be done in time $O(\log n)$ by Euclid's algorithm, and a^{n-1} can be computed in $O(\log^2 n)$ time by repeated squaring, so this algorithm runs in time polylogarithmic in n (i.e., polynomial in the input size). It is also extremely efficient in practice.

Error probability

Clearly the algorithm is always correct when n is prime. However, when n is composite it may make an error if it fails to find a "witness," i.e., a number a such that $a^{n-1} \neq 1 \bmod n$. Unfortunately, there are composite numbers, known as "Carmichael numbers," that have no witnesses. The first three CN's are 561, 1105, and 1729. [**Exercise**: Prove that 561 is a CN. Hint: $561 = 3 \times 11 \times 17$.] These numbers are guaranteed to fool Fermat's Test.

However, it turns out that CN's are the *only* bad inputs for the algorithm, as we now show. In what follows, we use the notation $\mathbb{Z}_n$ to denote the additive group of integers mod n, and $\mathbb{Z}_n^*$ the multiplicative group of integers coprime to n (i.e., with $\gcd(a, n) = 1$).

Theorem: *If n is composite and not a Carmichael number, then* $\Pr[\text{Error}] \leq \frac{1}{2}$.

Proof: Let $S_n = \{a \in \mathbb{Z}_n^* : a^{n-1} = 1 \bmod n\}$, i.e., S_n is the set of $a \in \mathbb{Z}_n^*$ that are *not* witnesses. Clearly S_n is a subgroup of $\mathbb{Z}_n^*$ (because it contains 1 and is closed under multiplication). Moreover, it is a *proper* subgroup since n is not a CN and therefore by definition there is at least one witness $a \notin S_n$. By Lagrange's Theorem (a standard theorem from Group Theory), the size of any subgroup must divide the size of the group, so we may conclude that $\frac{|S_n|}{|\mathbb{Z}_n^*|} \leq \frac{1}{2}$. ■

Fortunately, CN's are rare: there are only 255 of them less than 10^8. For this reason, Fermat's Test actually performs quite well in practice. Indeed, even the simplified deterministic version which performs the test only with $a = 2$ is sometimes used to produce "industrial grade" primes. This simplified version makes only 22 errors in the first 10,000 integers. It has also been proved for this version that

$$\lim_{b \to \infty} \Pr[\text{Error on random } b\text{-bit number}] \to 0.$$

For values of b of 50 and 100, we get $\Pr[\text{Error}] \leq 10^{-6}$ and $\Pr[\text{Error}] \leq 10^{-13}$ respectively. However, as in many other situations considered in this course, we don't want to assume anything about the input (such as, that it is random) and would like to have an algorithm that is guaranteed to have small error probability on *every* input.

Dealing with Carmichael numbers

We will now present a more sophisticated algorithm (usually attributed to Miller and Rabin) that deals with Carmichael numbers. First observe that, if p is prime, the group $\mathbb{Z}_p^*$ is cyclic: $\mathbb{Z}_p^* = \{g, g^2, \cdots, g^{p-1} = 1\}$ for some generator $g \in \mathbb{Z}_p^*$. (Actually this holds for $n \in \{1, 2, 4\}$ and for $n = p^k$ or $n = 2p^k$ where p is an odd prime and k is a non-negative integer.) Note that then $\mathbb{Z}_p^* \cong \mathbb{Z}_{p-1}$. For example, here is the multiplication table for $\mathbb{Z}_7^*$, which has 3 and 5 as generators:

$\mathbb{Z}_7^*$	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	4	6	1	3	5
3	3	6	2	5	1	4
4	4	1	5	2	6	3
5	5	3	1	6	4	2
6	6	5	4	3	2	1

Definition: *a is a* **quadratic residue** *if $\exists\, x \in \mathbb{Z}_p^*$ such that $a = x^2 \bmod p$. We say that x is a* **square root** *of a.*

Thus a quadratic residue is just a perfect square in $\mathbb{Z}_p^*$. The following claim follows immediately from the cyclic structure of $\mathbb{Z}_p^*$ (exercise):

Claim: *For a prime p,*

(i) $a = g^j$ is a quadratic residue iff j is even (i.e., exactly half of Z_p^* are quadratic residues).

(ii) Each quadratic residue $a = g^j$ has exactly two square roots, namely $g^{\frac{j}{2}}$ and $g^{\frac{j}{2}+\frac{p-1}{2}}$.

As an example, from the above table it can be seen that 2, 4, and 1 are quadratic residues in $\mathbb{Z}_7^*$. We obtain the following corollary, which will form the basis of our primality test:

Corollary: *If p is prime, then 1 has no non-trivial square roots in $\mathbb{Z}_p^*$, i.e., the only square roots of 1 in $\mathbb{Z}_p^*$ are ± 1.*

In $\mathbb{Z}_n^*$ for composite n, there may be non-trivial roots of 1: for example, in $\mathbb{Z}_{35}$, $6^2 = 1$.

We can now present our improved primality test. The idea is to search for non-trivial square roots of 1. Specifically, assume that n is odd, and not a prime power. (We can detect prime powers in polynomial time and exclude them: Exercise!). Then $n-1$ is even, and we can write $n - 1 = 2^r R$ with R odd. We search by computing $a^R, a^{2R}, a^{4R}, \cdots, a^{2^r R} = a^{n-1}$ (all mod n). Each term in this sequence is the square of the previous one, and (assuming that a fails the Fermat Test, in which case we would be done anyway) the last term is 1. Thus if the first 1 in the sequence is preceded by a number other than -1, we have found a non-trivial root and can declare that n is composite. More specifically the algorithm works as follows:

for input n.
pick an $a \in \{1, ..., n-1\}$ uniformly at random
if $\gcd(a, n) \neq 1$ **then** ouput "composite" and halt
compute $b_i = a^{2^i R} \bmod n,\ i = 0, 1, \cdots r$
if $b_r [= a^{n-1}] \neq 1$ **then** output "composite" and halt
else if $b_0 = 1$ **then** output "prime" and halt
else let $j = \max\{i : b_i \neq 1\}$
if $b_j \neq -1 \bmod n$ **then** output "composite"
else output "prime"

For example, for the Carmichael number $n = 561$, we have $n - 1 = 560 = 2^4 \times 35$. If $a = 2$ then the sequence computed by the algorithm is $a^{35} \bmod 561 = 263$, $a^{70} \bmod 561 = 166$, $a^{140} \bmod 561 = 67$, $a^{280} \bmod 561 = 1$, $a^{560} \bmod 561 = 1$. So the algorithm finds that 67 is a non-trivial square root of 1 and therefore concludes that 561 is not prime.

It remains to show that the error probability is bounded when n is composite (so that the witness property of $a = 2$ in the above example is in fact not a fluke).

Analysis

If $a^{n-1} \neq 1$ mod n then we know that n is composite, otherwise the algorithm continues by searching for a nontrivial square root of 1. It examines the descending sequence of square roots beginning at $a^{n-1} = 1$:

$$a^{n-1}, a^{(n-1)/2}, a^{(n-1)/4}, \ldots, a^R.$$

There are three cases to consider:

(i) The powers are all equal to 1.

(ii) The first power (in descending order) that is not 1 is -1.

(iii) The first power (in descending order) that is not 1 is a nontrivial root of 1.

In the first two cases the algorithm fails to find a witness for the compositeness of n, so it guesses that n is prime. In the third case it has found some power of a that is a nontrivial square root of 1, so a is a witness that n is composite.

The key fact to prove is that, if n is composite, then the algorithm is quite likely to find a witness:

Theorem: *If n is odd, composite, and not a prime power, then* $\Pr[a \text{ is a witness}] \geq \frac{1}{2}$.

To prove this claim we will use the following definition and lemma.

Definition: *Call $s = 2^i R$ a **bad** power if $\exists x \in \mathbb{Z}_n^*$ such that $x^s = -1$ mod n.*

Lemma: *For any bad power s, $S_n = \{x \in \mathbb{Z}_n^* : x^s = \pm 1 \text{ mod } n\}$ is a proper subgroup of $\mathbb{Z}_n^*$.*

We will first use the Lemma to prove the Theorem, and then finish by proving the Lemma.

Proof of Theorem: Let $s^* = 2^{i^*} R$ be the largest bad power in the sequence $R, 2R, 2^2 R, \ldots, 2^r R$. (We know s^* exists because R is odd, so $(-1)^R = -1$ and hence R at least is bad.)

Let S_n be the proper subgroup corresponding to s^*, as given by the Lemma. Consider any non-witness a. One of the following cases must hold:

(i) $a^R = a^{2R} = a^{4R} = \ldots = a^{n-1} = 1 \text{ mod } n$

(ii) $a^{2^i R} = -1 \text{ mod } n$, $a^{2^{i+1} R} = \ldots = a^{n-1} = 1 \text{ mod } n$ (for some i).

In either case, we claim that $a \in S_n$. In case (i), $a^{s^*} = 1 \text{ mod } n$, so $a \in S_n$. In case (ii), we know that $2^i R$ is a bad power, and since s^* is the largest bad power then $a^{s^*} = \pm 1 \text{ mod } n$ and so $a \in S_n$. Therefore, all non-witnesses must be elements of the proper subgroup S_n. Using Lagrange's Theorem just as we did in the analysis of the Fermat Test earlier, we see that

$$\Pr[a \text{ is not a witness}] \leq \frac{|S_n|}{|\mathbb{Z}_n^*|} \leq \frac{1}{2}.$$

This completes the proof of the Theorem. ■

We now go back and provide the missing proof of the Lemma.

Proof of Lemma: S_n is clearly closed under multiplication and hence a subgroup, so we must only show that it is *proper*, i.e., that there is some element in $\mathbb{Z}_n^*$ but not in S_n. Since s is a bad power, we can fix an $x \in \mathbb{Z}_n^*$ such that $x^s = -1$. Since n is odd, composite, and not a prime power, we can find n_1 and n_2 such that n_1 and n_2 are odd, coprime, and $n = n_1 \cdot n_2$.

Since n_1 and n_2 are coprime, the Chinese Remainder Theorem implies that there exists a unique $y \in \mathbb{Z}_n$ such that

$$\begin{aligned} y &= x \text{ mod } n_1; \\ y &= 1 \text{ mod } n_2 \end{aligned}$$

We claim that $y \in \mathbb{Z}_n^* \setminus S_n$.

Since $y = x \text{ mod } n_1$ and $\gcd(x, n) = 1$, we know $\gcd(y, n_1) = \gcd(x, n_1) = 1$. Also, $\gcd(y, n_2) = 1$. Together these give $\gcd(y, n) = 1$. Therefore $y \in \mathbb{Z}_n^*$.

We also know that

$$\begin{aligned} y^s &= x^s \bmod n_1 \\ &= -1 \bmod n_1 \quad (*) \\ y^s &= 1 \bmod n_2 \quad (**) \end{aligned}$$

Suppose $y \in S_n$. Then by definition, $y^s = \pm 1 \bmod n$.

If $y^s = 1 \bmod n$, then $y^s = 1 \bmod n_1$ which contradicts $(*)$.

If $y^s = -1 \bmod n$, then $y^s = -1 \bmod n_2$ which contradicts $(**)$.

Therefore, y cannot be an element of S_n, so S_n must be a proper subgroup of $\mathbb{Z}_n^*$. This completes the proof of the Lemma. ■