Lecture 14: Primality Testing

Algorithmic Number Theory (Fall 2014)
Rutgers University
Swastik Kopparty
Scribe: Nathan Fox

1 Overview

Given a positive integer n, we want to determine whether n is prime in time $\text{polylog}\, n$. Since the input size for this problem is $\log n$, this would mean an efficient algorithm for testing primality. Since the 1970s, we have had results of the following types:

- Efficiently-verifiable certificates of primality. (See Lecture 13.)
- Efficient randomized algorithms[1].
- Efficient deterministic algorithms under the Generalized Riemann Hypothesis.

Here, we will establish a deterministic test, due to Agrawal, Kayal, and Saxena, that checks whether n is prime in time $\text{polylog}\, n$.

2 Randomized Algorithms

2.1 First Attempt: Using Fermat's Little Theorem

The following is a well-known result about prime numbers.

Theorem 1. *[Fermat's Little Theorem] If n is a prime number and a is any integer, then $a^n \equiv a \pmod{n}$.*

The immediate question is whether Theorem 1 can be turned into an efficient randomized test for primality. If the converse of Theorem 1 were true, then testing this congruence for a random integer a would constitute a randomized primality test. (Though the efficiency of this test would still need to be established separately.) It turns out, though, that the converse is not true. There exist composite numbers n such that for all integers a, $a^n \equiv a \pmod{n}$ (the so-called *Carmichael Numbers*). So, no primality test based entirely on Fermat's Little Theorem can exist.

[1] By *efficient randomized primality test*, we mean an algorithm $A(n)$ that, for every n

- runs in time $\text{polylog}\, n$
- uses randomness
- if n is prime, $\Pr(\text{output of } A(n) = \text{PRIME}) = 1$
- if n is composite, $\Pr(\text{output of } A(n) = \text{COMPOSITE}) \geq \alpha$ for some constant α (say, 0.9) independent of n

2.2 The Agrawal-Biswas Algorithm

The test will be based on the following theorem.

Theorem 2. *A positive integer n is prime if and only if $(X+1)^n \equiv X^n + 1 \pmod{n}$ in $\mathbb{Z}[X]$.*

To prove this theorem, we will extract one lemma. (We isolate this lemma, not because it is particularly difficult, but because we are going to use it again later.)

Lemma 3. *Let n be a positive integer. If n is composite, then for any prime factor p of n, n does not divide $\binom{n}{p}$.*

Proof. Write n as $n = pm$ for some prime p and positive integer m. We have that

$$\begin{aligned}\binom{n}{p} &= \frac{n\,(n-1)\,(n-2)\cdots(n-p+1)}{p\,(p-1)\cdots 1} \\ &= \left(\frac{n}{p}\right)\left(\frac{(n-1)\,(n-2)\cdots(n-p+1)}{(p-1)\cdots 1}\right) \\ &= m\left(\frac{(n-1)\,(n-2)\cdots(n-p+1)}{(p-1)\cdots 1}\right).\end{aligned}$$

Since $p \mid n$, no term in the numerator of the fraction in parentheses is divisible by p. So, $\frac{1}{m}\binom{n}{p}$ is not divisible by p. This implies that $\binom{n}{p}$ is not divisible by n, as required. □

We will now prove Theorem 2.

Proof. By the Binomial Theorem,

$$(X+1)^n = X^n + 1 + \sum_{j=1}^{n-1} \binom{n}{j} X^{n-j}.$$

If n is prime, then for all $j \in [n-1]$, $n \mid \binom{n}{j}$. This implies that $(X+1)^n \equiv X^n + 1 \pmod{n}$, as required.

If n is composite Lemma 3 implies that the coefficient on X^{n-p} in $(X+1)^n$ is nonzero mod n, as required. □

Theorem 2 says that we can test primality of n by testing polynomial equality mod n. The natural way of testing polynomial equality with a randomized algorithm is to evaulate both polynomials and check if equality holds. This leads to the following candidate algorithm:

Algorithm 1.

- *Pick $x \in [n]$ uniformly at random.*
- *If $(x+1)^n \equiv x^n + 1 \pmod{n}$, then output* PRIME*, else output* COMPOSITE*.*

This natural agorithm does not work. For example, it fails for the Carmichael numbers. The reason Algorithm 1 can fail is that this method of comparing polynomials only works when the degree of the polynomials is smaller than the number of possible inputs to the polynomials. In this case, there are n inputs, but both polynomials have degree n.

Instead, the following modification of the naive Algorithm 1, due to Agrawal and Biswas, yields an efficient randomized primality test based on Theorem 2.

Algorithm 2.

- *Choose a monic degree-3 polynomial* $Q(X) \in \mathbb{Z}/n\mathbb{Z}[X]$ *uniformly at random.*
- *Test if* $(X+1)^n \equiv X^n + 1 \pmod{n, Q(X)}$.
- *If yes, then output* PRIME*, else output* COMPOSITE.

The only step whose efficiency requires justification is the computation of the powers of X and $X+1$. But, this is easily accomplished in polynomial time using repeated squaring.

We will now prove the correctness of Algorithm 2.

Proof. We need to show that Algorithm 2 always outputs PRIME when n is prime, and it outputs COMPOSITE a constant fraction of the time when n is composite. By Theorem 2, Algorithm 2 outputs PRIME when n is prime.

So, assume that n is composite. By Theorem 2, we know that $(X+1)^n - X^n - 1$ is nonzero mod n. Let p be any prime factor of n. The number of irreducible degree-3 polynomials in $\mathbb{F}_p$ is $\frac{p^3-p}{3} < \frac{p^3}{3}$. So the probability that $Q(X)$ is irreducible mod p is less than $\frac{1}{3}$. Since p divides n, the probability that $Q(X)$ is irreducible mod n is less than $\frac{1}{3}$.

Now, assume that Algorithm 2 outputs PRIME with probability α on input n. This means that with probability α, $(X+1)^n \equiv X^n + 1 \pmod{n, Q(X)}$. This means that $(X+1)^n - X^n + 1$ is divisible by at least $\left(\alpha - \frac{2}{3}\right) n^3$ distinct irreducible polynomials of degree 3. By the Chinese Remainder Theorem, $(X+1)^n - X^n + 1$ is then divisible by the product of these irreducible polynomials, so we have that its degree, which is n, is at most $3\left(\alpha - \frac{2}{3}\right) n^3$. So, we have

$$\begin{aligned} & n \geq 3\left(\alpha - \frac{2}{3}\right) n^3 \\ \Rightarrow\ & n \geq (3\alpha - 2)\, n^3 \\ \Rightarrow\ & \frac{1}{n^2} \geq 3\alpha - 2 \\ \Rightarrow\ & 2 + \frac{1}{n^2} \geq 3\alpha \\ \Rightarrow\ & \alpha \leq \frac{2}{3} + \frac{1}{3n^2}. \end{aligned}$$

So, if n is composite, Algorithm 2 outputs PRIME with probability at most $\frac{2}{3} + \frac{1}{3n^2}$. So, it outputs COMPOSITE with probability at least $1 - \left(\frac{2}{3} + \frac{1}{3n^2}\right) \geq \frac{1}{4}$, which is a constant independent of n, as required. □

3 The Deterministic Agrawal-Kayal-Saxena Algorithm

We will now establish an efficient, deterministic primality test by "de-randomizing" the Agrawal-Biswas Algorithm. This algorithm is due to Agrawal, Kayal, and Saxena. First, we will prove the following generalization of Theorem 2.

Theorem 4. *Let n and a be positive integers such that a is not divisible by n. Then, n is prime if and only if $(X+a)^n \equiv X^n + a \pmod{n}$ in $\mathbb{Z}[X]$.*

Proof. By the Binomial Theorem,

$$(X+a)^n = X^n + a^n + \sum_{j=1}^{n-1} \binom{n}{j} X^{n-j} a^j.$$

If n is prime, then for all $j \in [n-1]$, $n \mid \binom{n}{j}$. This implies that $(X+a)^n \equiv X^n + a^n \pmod{n}$. By Fermat's Little Theorem, $a^n \equiv a \pmod{n}$, so $(X+a)^n \equiv X^n + a \pmod{n}$, as required.

Now, consider the case where n is composite and where n has a prime factor p that a does not. Lemma 3 implies that the binomial coefficient on X^{n-p} in $(X+a)^n$ is nonzero mod n, and the factor a^p will not re-introduce that prime. So, the entire coefficient on X^{n-p} is nonzero mod n, as required.

The last case to consider is when every prime factor of n is also a prime factor of a. In this case, every prime factor of n divides a^n at least n times. Every prime factor of n divides n at most n times, so we must have n dividing a^n. This means that the constant term in $(X+a)^n$ is 0 mod n. But, a is not 0 mod n, so $(X+a)^n \not\equiv X^n + a \pmod{n}$, as required.

(Note that without the assumption that n not divide a, the statement is actually false.) □

We can now state the AKS algorithm.

Algorithm 3.

- *If n is a perfect power, output* COMPOSITE.
- *Let $R = \log^5 n$, $A = \log^6 n$.*
- *If for some $b \leq r$, $b \mid n$, output* COMPOSITE.
- *For each $r \leq R$, $a \leq A$, test if $(X+a)^n \equiv X^n + a \pmod{n, X^r - 1}$.*
- *If all are identities, output* PRIME, *else output* COMPOSITE.

The fact that Algorithm 3 runs in polynomial time follows from the following observations:

- Testing for perfect powers requires testing only $\log n$ many powers, each of which can be done in $\log n$ time using binary search.
- The constants R and A are polynomials in $\log n$.

- Powers of $X + a$ and X can be computed quickly by repeated squaring.

Also, Algorithm 3 is clearly deterministic. So, all that remains is to show that Algorithm 3 outputs PRIME if and only if n is prime. Our analysis will use the following definition and lemma.

Definition 5. *The* order *of n mod r, denoted $\mathrm{Ord}_r(n)$, is the smallest positive integer m such that $n^m \equiv 1 \pmod{r}$.*

Lemma 6. *There exists a positive integer $r \leq R$ such that $\mathrm{Ord}_r(n) \geq \log^2(n)$.*

Proof. Let

$$N = \prod_{i=1}^{\lfloor \log^2(n) \rfloor} \left(n^i - 1\right).$$

Let r be such that $\mathrm{Ord}_r(n) < \log^2(n)$. Then, r divides N. We see that $N = n^{\Theta\left(\log^3(n)\right)}$. So, N has at most $\log N = O\left(\log^4(n)\right)$ prime factors. But, there are $\frac{\log^5(n)}{\log\log^5(n)}$ primes less than R, so there is some r less than R not dividing N. Such an r has $\mathrm{Ord}_r(n) \geq \log^2(n)$, as required. □

We will now prove that Algorithm 3 outputs PRIME if and only if n is prime.

Proof. If n is prime, then Theorem 4 implies that Algorithm 3 outputs PRIME, as required. If n is a perfect power, then Algorithm 3 outputs COMPOSITE, as required. If n has a small factor, then Algorithm 3 outputs COMPOSITE, as required. So, all that remains to check is the case where n is composite, not a perfect power, and has no small factors.

Suppose that n is as above, and suppose that r is such that $\mathrm{Ord}_r(n) \geq \log^2(n)$ (guaranteed to exist by Lemma 6). Let p be a prime dividing n such that $p \not\equiv 1 \pmod{r}$. (Such a p exists because $n \not\equiv 1 \pmod{r}$.) Let $\mu_r = \left\{\alpha \in \overline{\mathbb{F}_p} \mid \alpha^r = 1\right\}$ be the set of r^{th} roots of unity in $\overline{\mathbb{F}_p}$ (the algebraic closure of $\mathbb{F}_p$). Assume for a contradiction that Algorithm 3 outputs PRIME on input n. Then, for all $\alpha \in \mu_r$ and for all $a \leq A$, $(\alpha + a)^n = \alpha^n + a$ (in $\overline{\mathbb{F}_p}$).

Let $S = \{Z + a \mid a \leq A\} \subseteq \mathbb{F}_p[Z]$, and let $T = \{n\}$. (These are initial settings; we will add more elements to these sets shortly.) We observe that for all $Q(Z) \in S$ and for all $m \in T$, for all $\alpha \in \mu_r$, $Q(\alpha)^m = Q(\alpha^m)$. We will call this the property that *m and Q commute*. We wish to find more elements m and polynomials Q that commute. We will add the elements to T and the polynomials to Q. We already know that $(\alpha + a)^p = \alpha^p + a$ by Fermat's Little Theorem. So, we can add p to T.

Now, observe that if Q_1 and Q_2 commute with m, then $Q_1 Q_2$ commutes with m, and observe that if m_1 and m_2 commute with Q, then $m_1 m_2$ commutes with Q. This will let us expand S and T further.

See the notes for Lecture 15 for the conclusion of this proof. □