CS271 Randomness & Computation **Spring 2018**

Lecture 4: January 25

Instructor: Alistair Sinclair

4.1 A randomized algorithm for primality testing

In the last lecture we saw an efficient randomized algorithm for primality testing with one-sided error that has bounded error probability on all inputs except for Carmichael numbers. We will now present a more sophisticated algorithm (usually attributed to Miller and Rabin) that deals with all inputs, including 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 $g \in \mathbb{Z}_p^*$. (Actually this holds slightly more generally, 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 4.1 *a is a* **quadratic residue** *if $\exists\ x \in \mathbb{Z}_p^*$ such that $a = x^2 \mod p$. We say that x is a* **square root** *of a.*

Claim 4.2 *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 $g^{\frac{j}{2}+\frac{p-1}{2}}$

Proof: Easy exercise. (Hint: note that $p - 1$ is even.) ■

Note that 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 4.3 *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 $\mathbb{Z}_{35}$, $6^2 = 1$.

The idea of the algorithm is to search for non-trivial square roots of 1. Specifically, assume that n is odd, and not a prime power. (We can detect perfect 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 the last term is 1 (otherwise we have failed the Fermat test and n is composite). 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:

if $n > 2$ is even or a perfect power **then** output "composite"
compute r, R s.t. $n-1 = 2^r \cdot R$ [*Note: R is odd*]
pick $a \in \{1, ..., n-1\}$ 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 $b_r [= a^{n-1}] \neq 1 \bmod n$ **then** output "composite" and halt
else if $b_0 = 1 \bmod n$ **then** output "prime" and halt
else let $j = \max\{i : b_i \neq 1\}$
if $b_j \neq -1$ **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.

Notice that the output "composite" is always correct. However the algorithm may err when it outputs "prime". It remains to show that the error probability is bounded when n is composite; we will do this next.

The algorithm begins by testing to see if a randomly chosen a passes the test of Fermat's little theorem. If $a^{n-1} \neq 1 \bmod n$, then we know that n is composite, otherwise we continue by searching for a nontrivial square root of 1. We examine the sequence of descending square roots beginning at $a^{n-1} = 1$ until we reach an odd power of a:

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

There are three cases to consider:

1. The powers are all equal to 1.
2. The first power (in descending order) that is not 1 is -1.
3. The first power (in descending order) that is not 1 is a nontrivial root of 1.

In the first two cases we fail to find a witness for the compositeness of n, so we guess that n is prime. In the third case we have found that some power of a is a nontrivial square root of 1, so a is a witness that n is composite.

4.1.1 The likelihood of finding a witness

We now show that if n is composite, we are fairly likely to find a witness.

Claim 4.4 *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 4.5 *Call $s = 2^i R$ a **bad** power if $\exists x \in \mathbb{Z}_n^*$ such that $x^s = -1 \bmod n$.*

Lemma 4.6 *For any bad power s, $S_n = \{x \in \mathbb{Z}_n^* : x^s = \pm 1 \bmod n\}$ is a proper subgroup of $\mathbb{Z}_n^*$.*

We will first use the lemma to prove claim 4.4, and then finish by proving the lemma.

Proof of Claim 4.4: Let $s^* = 2^{i^*} R$ be the largest bad power in the sequence $R, 2R, 2^2R, \ldots, 2^r R$. (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 Lemma 4.6. Consider any non-witness a. One of the following cases must hold:

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

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

In either case, we claim that $a \in S_n$. In case (i), $a^{s^*} = 1 \bmod n$, 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 $s^* \geq 2^i R$, implying $a^{s^*} = \pm 1 \bmod n$ 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, we see that

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

■

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

Proof of Lemma 4.6:

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

$$\begin{aligned} y &= x \bmod n_1; \\ y &= 1 \bmod n_2. \end{aligned}$$

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

Since $y = x \bmod n_1$ 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 $\mathbb{Z}_n^*$. ■

4.1.2 Notes and some background on primality testing

The above ideas are generally attributed to both Miller [M76] and Rabin [R76]. More accurately, the randomized algorithm is due to Rabin, while Miller gave a deterministic version that runs in polynomial time assuming the Extended Riemann Hypothesis (ERH): specifically, Miller proved under the ERH that a witness a of the type used in the algorithm is guaranteed to exist within the first $O((\log n)^2)$ values of a. Of course, proving the ERH would require a major breakthrough in Mathematics.

A tighter analysis of the above algorithm shows that the probability of finding a witness for a composite number is at least $\frac{3}{4}$, which is asymptotically tight.

Another famous primality testing algorithm, with essentially the same high-level properties and relying crucially on the subgroup trick but with a rather different type of witness, was developed by Solovay and Strassen around the same time as the Miller/Rabin algorithm [SS77]. Both of these algorithms, and variants on them, are used routinely today to certify massive primes having thousands of bits (required in applications such as the RSA cryptosystem).

In 2002, Agrawal, Kayal, and Saxena made a theoretical breakthrough by giving a deterministic polynomial time algorithm for primality testing [AKS02]. However, it is much less efficient in practice than the above randomized algorithms. This algorithm was inspired by the randomized polynomial time algorithm of Agrawal and Biswas in 1999 [AB99], which uses a generalization of the Fermat test to polynomials. We will sketch this algorithm in the next section.

The algorithm we have seen has a one-sided error: for prime n, the probability of error is 0, while for composite n the probability of error is at most $\frac{1}{2}$. Adleman and Huang [AH87] came up with a polynomial time randomized algorithm with one-sided error in the opposite direction, i.e., it is always correct on composites, but may err on primes with probability at most $\frac{1}{2}$.[1] Although the Adleman-Huang algorithm is not very efficient in practice, it is of theoretical interest to note that it can be combined with the Miller-Rabin algorithm above to create a stronger *Las Vegas* algorithm for primality testing:

repeat forever
  run Miller-Rabin on n
  if Miller-Rabin outputs "composite" **then** output "composite" and halt
  run Adleman-Huang on n
  if Adleman-Huang outputs "prime" **then** output "prime" and halt

If this algorithm ever terminates it must be correct since Miller-Rabin never makes an error when it outputs "composite", and likewise for Adleman-Huang and "prime". Also, since the probability of error in each component is at most $\frac{1}{2}$, the probability of iterating t times without terminating decreases exponentially with t.

[1] More practically useful certificates of primality were developed by Goldwasser and Kilian [GK86]; to guarantee their existence one needs to appeal to an unproven assumption that is almost always true.

4.2 A deterministic algorithm for primality testing

We conclude our discussion of primality testing by sketching the route to the first polynomial time deterministic primality testing algorithm announced in 2002. This is based on another randomized algorithm due to Agrawal and Biswas [AB99], which was subsequently derandomized by Agrawal, Kayal and Saxena [AKS02]. We won't discuss the derandomization as that is not the main focus of the class.

The Agrawal-Biswas algorithm exploits a different number theoretic fact, which is a generalization of Fermat's Theorem, to find a witness for a composite number. Namely:

Fact 4.7 *For every $a > 1$ such that $gcd(a, n) = 1$, n is a prime iff $(x - a)^n = x^n - a \bmod n$.*

Exercise: Prove this fact. [Hint: Use the fact that $\binom{n}{k} = 0 \bmod n$ for all $0 < k < n$ iff n is prime.]

The obvious approach for designing an algorithm around this fact is to use the Schwartz-Zippel test to see if $(x - a)^n - (x^n - a)$ is the zero polynomial. However, this fails for two reasons. The first is that if n is not prime then $\mathbf{Z}_n$ is not a field (which we assumed in our analysis of Schwartz-Zippel); the second is that the degree of the polynomial is n, which is the same as the cardinality of $\mathbf{Z}_n$ and thus too large (recall that Schwartz-Zippel requires that values be chosen from a set of size strictly larger than the degree).

Instead, we will use a variant of fingerprinting. Rather than a prime divisor as in standard fingerprinting, in this case our witness will be a low-degree polynomial, $r(x)$, such that $(x + 1)^n \neq x^n + 1 \bmod (r(x), n)$. Plainly, if such a polynomial $r(x)$ exists then, from the above Fact, we can be sure that n is composite; while if n is prime then no such $r(x)$ can exist. Thus we will again get an algorithm with one-sided error. As usual, the challenge is to show that the density of witnesses is large enough if we pick them from a suitable set.

Here is the algorithm:

if n has a divisor less than 17 **or if** n is a perfect power **then output** "composite"
let $d = \lceil \log n \rceil$
let $r(x) = x^d + r_{d-1}x^{d-1} + \cdots + r_1 x + r_0$ where each r_i is chosen from $\mathbf{Z}_n$ uniformly at random
if $(x + 1)^n \neq x^n + 1 \bmod (r(x), n)$ **then**
    output "composite"
else
    output "prime?"

Note that this algorithm can be implemented to run in time polylog(n), since all arithmetic is done modulo the polynomial $r(x)$ (of degree $\log n$) with coefficients in $\mathbf{Z}_n$, and the power $(x + 1)^n$ can be computed by repeated squaring using $O(\log n)$ multiplications.

4.2.1 The likelihood of finding a witness

For the analysis, let us assume that n is composite. By the first line of the algorithm, we may also assume that n is not a prime power, and has a prime divisor $p \geq 17$. Let $Q(x) = (x + 1)^n - (x^n + 1)$. We claim first that not only do we have $Q(x) \neq 0 \bmod n$, but also $Q(x) \neq 0 \bmod p$. (**Exercise:** prove this, along similar lines to the proof of the Fact above.) Now we can argue over $\mathbf{Z}_p$ instead of $\mathbf{Z}_n$. Since $Q(x) \neq 0$ over $\mathbf{Z}_p$, it has a unique factorization (up to a constant). Also, since $r(x)$ is a random (monic) polynomial over $\mathbf{Z}_n$ it is also a random (monic) polynomial over $\mathbf{Z}_p$. And, as $Q(x)$ has degree n, it can have at most $\frac{n}{d}$ irreducible factors of degree d.

Now, we can discuss the probability that $r(x)$ is a witness to n being a composite number:

$$\begin{aligned}\Pr[r(x) \text{ is a witness}] &\geq \Pr[r(x) \text{ is irreducible and } r(x) \text{ is not a factor of } Q(x)] \\ &= \Pr[r(x) \text{ is irreducible}] - \Pr[r(x) \text{ is an irreducible factor of } Q(x)].\end{aligned}$$

A known fact is that the number of irreducible monic polynomials of degree d over $\mathbf{Z}_p$ is at least $\frac{p^d}{d} - \sqrt{p^d}$. When $p \geq 17$ and $d > 4$ this is at least $\frac{p^d}{2d}$. This allows us to lower bound the probability that $r(x)$ is irreducible:

$$\Pr[r(x) \text{ is irreducible}] \geq \frac{p^d/2d}{p^d} = \frac{1}{2d}.$$

Similarly, we can bound the probabilty that $r(x)$ divides $Q(x)$.

$$\Pr[r(x) \text{ is an irreducible factor of } Q(x)] \leq \frac{n}{d}\frac{1}{p^d} \leq \frac{1}{4d}.$$

The last inequality is due (very crudely) to the fact that $p \geq 17$ and $d = \lceil \log n \rceil$.

Putting these together, we have that

$$\Pr[r(x) \text{ is a witness}] \geq \frac{1}{2d} - \frac{1}{4d} = \frac{1}{4d} = \Omega(\frac{1}{\log n}).$$

Thus when n is composite the algorithm finds a witness with probability $\Omega(\frac{1}{\log n})$. This means that $O(\log n)$ repeated trials suffice to find a witness with high probability.

4.2.2 Derandomization

The deterministic algorithm of [AKS02] is based on a derandomization of the above algorithm. The (very rough) idea is to use a *fixed* polynomial $r(x) = x^d - 1$ for $d \approx (\log n)^2$, and to search instead through the values of a, testing whether $(x-a)^n \neq x^n - a \bmod (r(x), n)$. It turns out that checking each value of a up to about $(\log n)^2$ is guaranteed to reveal a witness if n is composite. The running time of the resulting algorithm is roughly $O((\log n)^6)$, making it not useful in practice. We omit the details.

References

[AH87] L.M. Adleman and A.M.-D. Huang, "Recognizing primes in random polynomial time," *Proceedings of the 19th ACM STOC*, 1987, pp. 462–469.

[AB99] M. Agrawal and S. Biswas, "Primality and identity testing via Chinese remaindering," *Proceedings of IEEE FOCS* 1999, pp. 202–209. Full version appeared in *Journal of the ACM* **50** (2003), pp. 429–443.

[AKS02] M. Agrawal, N. Kayal and N. Saxena, "PRIMES is in P," *Annals of Mathematics* **160** (2004), pp. 781–793. Result first announced in a preprint, August 2002: http://www.cse.iitk.ac.in/users/manindra/primality_original.pdf

[GK86] S. Goldwasser and J. Kilian, "Almost all primes can be quickly certified," *Proceedings of the 18th ACM STOC*, 1986, pp. 316–329.

[M76] G.L. Miller, "Riemann's hypothesis and tests for primality," *Journal of Computer and Systems Sciences* **13** (1976), pp. 300–317.

[R76] M.O. Rabin, "Probabilistic algorithms," in J.F. Traub (ed.), *Algorithms and Complexity, Recent Results and New Directions*, Academic Press, New York, 1976.

[SS77] R. Solovay and V. Strassen, "A fast Monte Carlo test for primality," *SIAM Journal on Computing* **6** (1977), pp. 84–85. See also *SIAM Journal on Computing* **7** (1978), p. 118.