Primality Testing and Factorization Algorithms

Xinke Guo-Xue

December 2023

1 Introduction

In the paper we explore a few primality tests and factoring algorithms.

The history of primality tests is rich. The AKS primality test was developed in 2002 and was the first general, deterministic, unconditionally correct polynomial time algorithm for factoring. Before the AKS primality test, primality tests were only three of general, deterministic, unconditionally correct, and polynomial time. We will not go over the AKS algorithm, but it stands as a cornerstone of primality testing.

We will give an overview of the Fermat Primality Test, Solovay Strassen, and the Miller Rabin Test.

There are no known non-quantum polynomial time algorithms for factoring large numbers. The RSA cryptosystem hinges on the difficulty of the factoring problem to encrypt messages. We will explore how the RSA cryptosystem works. We will also explore Pollard's Rho Algorithm to get a sense of how factoring algorithms can work when designed well.

2 Deterministic Primality tests

Wilson's theorem can be used to check if a number is composite.

2.1 Wilson's Theorem

Wilson's Theorem says that if n is prime, then $(n-1)! + 1 \equiv 0 \mod n$. This is if and only if. If n is not prime, then $(n-1)! \not\equiv -1 \pmod{}$, since n would share a factor $k > 1$ with $(n-1)!$. Hence to check if n is composite, we can check if $(n-1)! \equiv 0 \pmod{n}$.

Wilson's Theorem was stated by the English mathematician John Wilson in the 1700's, though John Wilson was not the first to state this theorem.

3 Probabilistic primality tests

Many primality tests do not deterministically check if a number is prime or not, but check if a number is most likely prime or not.

3.1 Fermat Primality Test

The Fermat Primality test is as follows: Pick an a relatively prime to n. Then, if a^{n-1} is not congruent to 1 mod n, then n is likely a composite number.

However, there is a chance that n is still composite even if $a^{n-1} \equiv 1 \pmod{n}$. For example $2^{340} \equiv 1 \pmod{341}$, but $341 = 11 \cdot 31$. When this happens, n is said to be a Fermat liar, as it fails the Fermat Primality test. 341 is said to be a pseudoprime base 2.

However, if we had picked an a such that $a^{n-1} \not\equiv 1 \pmod{p}$, then a is said to be a Fermat witness for the compositeness of n.

In general, if n has one Fermat witness, then half the residue classes mod n are Fermat witnesses.

However, one can repeatedly check numbers mod n to test the compositeness of n. If each check has a $1/2$ probability of failing, then checking 100 times will have a $1/2^{100}$ chance of failing, which is good enough for most purposes.

Example 1. Suppose we want to check if 1001 is prime. If we check $a = 2$, then

$$2^{1001} \equiv 562 \pmod{1001}.$$

Hence 1001 is not prime.

3.2 Solovay Strassen

The Solovay Strassen Primality test is a primality test devised by Robert M. Solovay and Volker Strassen in 1977.

Let $\left(\frac{a}{n}\right)$ be the Jacobi symbol. Euler's criterion says that when n is prime,

$$\left(\frac{a}{n}\right) \equiv a^{\frac{n-1}{2}} \pmod{p}.$$

If we do not know the primality of n, then if we find a single a such that the above congruence fails, then we know n is composite.

If this happens, then a is called an **Euler witness** for the compositeness of n.

In general, if n is prime, half of the residue classes relatively prime to n are Euler witnesses for n, so a probabilistic test can be designed by repeatedly choosing a.

Example 2. Suppose we choose $n = 15$. Then if we choose $a = 2$, then

$$2^7 \equiv 128 \equiv 8 \not\equiv \left(\frac{2}{15}\right)$$

so 15 is not prime, and 2 is an Euler witness for the compositeness of 15.

3.3 Miller Rabin Test

The Miller Rabin test was invented by Gary Miller in a paper published in 1977.

The Miller Rabin test uses the following two facts:

In a prime modulus

- $a^{p-1} \equiv 1 \pmod{p}$.
- The only square roots of 1 modulo p are 1 and -1.

The Miller Rabin test is as follows:

Check if $a^{n-1} \equiv 1 \pmod{n}$. If yes, then we also check if $a^{\frac{n-1}{2}} \equiv 1 \pmod{n}$. If yes, we continue this process. If the first time $a^{(n-1)/2^r}$ is not 1, it is actually -1, then n is likely prime, but if the first time $a^{(n-1)/2^r}$ is not 1, it is something other than -1 or 1, then n is not prime, since in a prime modulus, the only square roots of 1 are ± 1. If the first time $a^{n/2^r}$ for $r = 1, 2, 3 \ldots$ is not 1 is when it is -1, and n is said to pass the Miller Rabin test, and n is likely prime.

We can formalize the test as follows:

Let 2^s be the largest power of 2 dividing $n-1$, so $n-1 = 2^s q$ for some odd number q. Each member of the sequence

$$a^{n-1} = a^{2^s q}, a^{2^{s-1} q}, \ldots a^q$$

is a square root of the preceding number.

If n is prime, every member of this sequence is 1, and the first member of this sequence that is not 1 is -1. If this happens, then n is said to be a strong probably prime to base a. n is said to pass the Miller Rabin test for this particular value of a in this case.

If this does not happen, then a is said to be a *witness* for the compositeness of n.

However, it is possible that n passes the Miller Rabin test for a particular value of a but is composite. Then n is said to be a *strong pseudoprime* to the base a, and n is said to be a *strong liar*.

It turns out that the probability that n passes the Miller Rabin test but is composite is at most $1/4$. Thus, repeatedly choosing different values of a to run the test on will exponentially decrease the probability that the test incorrectly detects it is prime.

Example 3. Suppose we wanted to find if 91 is prime. Suppose we pick $a = 3$. Then

$$3^{90} \equiv 1 \pmod{91}.$$

Hence 91 passes the Fermat Primality test with the base $a = 3$.

However,

$$3^{45} \equiv 27 \pmod{91}$$

so 3 is a witness for the compositeness of 91.

4 Factorization Algorithms

People have been trying to factor integers for centuries, as this question is central in mathematics. The first factorization algorithm was trial division. Fermat's Difference of Squares Factorization gave a way to find factors top-down. There is no known non-quantum way to factor a large integer in polynomial time.

4.1 Fermat's Factorization Method

Fermat's Factorization Method is a way to factor certain integers. It is named after Pierre de Fermat.

Example 4. Suppose you want to factor 2491. Note that it is close to 2500, a perfect square. Then note it differs from 2500 by a perfect square. Hence it factors as

$$(2491 = 2500 - 9 = 50^2 - 3^2 = (50+3)(50-3).$$

This is Fermat's Factorization Method.

We can formalize Fermat's Factorization Method as follows: Find the smallest perfect square k^2 above n. Check if $k^2 - n$ is a perfect square. If yes, proceed to factor n as a difference of squares. If not, check the next biggest perfect square k_2^2 above n, and repeat the process.

Since any odd integer and any even integer not 2 (mod 4) can be written as a difference of squares, this method can be made to work to factor any number not 2 (mod 4).

If n has only large prime factors, this may work faster than trial division. In practice, it may be good to try Fermat's Factorization Method on a large odd number before attempting trial division (testing divisibility by small primes).

4.2 Pollard Rho Algorithm

Pollard Rho is a way to find a factor of a large number. This algorithm was devised by John Pollard in 1975.

Suppose that m is a large composite number whose smallest prime divisor is p. If we choose k integers $u_1, u_2, \ldots u_k$ at random, with k large compared to $\sqrt{p}$ but small compared to $\sqrt{m}$, then, if the u_i behave randomly mod p and mod m, it is likely that the u_i will be distinct $\pmod{m}$, but not distinct $\pmod{p}$. That is, there are probably integers i, j with $1 \leq i \leq j \leq k$ such that $1 < (u_i - u_j, m) < m$.

Each pair (i, j) can be tested by the Euclidean Algorithm, but the task of inspecting all $\binom{k}{2}$ pairs will take quite some time. To reduce the time this step takes, we can generate the u_i in the following way:

We generate u_i via a recursion of the form $u_{i+1} = f(u_i)$, where $f(u)$ is a polynomial with integral coefficients. The only condition on $f(u)$ is that it should generate a sequence of numbers that "looks random" mod m and mod p. It turns out that $f(u) = u^2 + 1$ works well.

The advantage of generating the u_i this way is that if $u_i \equiv u_j \pmod{d}$, then $u_{i+1} = f(u_i) \equiv f(u_j) = u_{j+1} \pmod{d}$, so the sequence u_i eventually becomes periodic modulo d with period $j - i$. If we put $r = j - i$, then $u_s \equiv u_t \pmod{d}$ whenever $s \equiv t \pmod{r}$, $r \geq i$, $s \geq i$. If we let s be the least multiple of r that is greater than or equal to i, then we take $t = 2s$. Then $u_s \equiv u_{2s} \pmod{d}$. That is, among the numbers $u_{2s} - u_s$, we expect to find one for which $1 < (u_2 s - u_s, m) < m$, with s of size roughly comparable to $\sqrt{p}$.

Note this may not find the smallest prime factor of m but it will find some prime of m if it finds one at all. If it does not find a prime factor, we choose different initial values.

Example 5. Suppose we want to find a proper divisor of $m = 77$.

We take $u_0 = 1$ and $u_{i+1} = u_i^2 + 1 \pmod{m}$. Then the first few values of u_i are

$$1, 2, 5, 26, 61, 26, 61, 26, 61, \ldots$$

The first time when the sequence becomes periodic is when $u_4 = 26 = u_8 = 26$, so $(26, 77) > 1$. We can check this is indeed true, as $13 \mid 77$ and $13 \mid 26$.

Hence by randomly generating $\geq \sqrt{m}$ integers from 1 to m, we found two that differ by a difference that shares a proper divisor with m with not unlikely probability.

4.3 Diffie Hellman Key Exchange

Suppose Alice and Bob want to exchange a shared private key with which to encrypt messages. However, their messages may be intercepted by an eavesdropper Eve, and they have no way of knowing when/if Eve eavesdrops on their messages. They can do so without meeting each other in person via the Diffie Hellman Key Exchange.

The Diffie Hellman Key Exchange is as follows:

Alice and Bob agree upon a prime modulus p and a primitive root g of that prime modulus. These can both be public knowledge.

Alice picks a number a and keeps it secret, while Bob picks a number b and keeps it secret. Alice computes $A = g^a$ and sends it to Bob, and Bob computes $A^b = (g^a)^b$.

Meanwhile Bob computes $B = g^b$ and sends it to Alice, which Alice then takes and computes $B^a = (g^b)^a$.

Now, they both have a shared private key g^{ab}. Note it is fine that g^a and g^b are public, since computing g^{ab} given g, g^a, g^b is hard - this is equivalent to the Discrete Logarithm problem, which is known to be hard to solve.

Note Alice has no knowledge of what Bob's number of b is - she just knows g, g^a, and g^{ab}. Similarly. Bob has no knowledge of what Alice's number a is, he just knows g, g^b, g^{ab}.

Hence Alice and Bob can share a secret key with which to encrypt messages securely online despite the presence of a possible eavesdropper. In practice, one can convert a message consisting of a string of letters into a number, which can be encoded as g^{ab} in a big enough prime modulus.

The Diffie Hellman scheme was published by Whitfield Diffie and Martin Hellman in 1976.

4.4 RSA algorithm

Suppose Bob wants to send a message m securely to Alice. However, Eve the eavesdropper is eavesdropping on the only channel Alice and Bob can communicate on. How can Bob trasmit his message securely to Alice without Eve being able to decode it?

The RSA algorithm offers a possible solution. The RSA algorithm is as follows.

Pick two large primes p and q It is common to let p and q to be around 100 digits long, and it is common to let p and q differ in length by a few digits - p and q should not be too close to each other.

Let $n = pq$. p and q are kept secret, but n is released as part of the public key, and perhaps shared via the Diffie Hellman key exchange, though this may be difficult if n is around 200 digits.

Compute $\lambda(n)$, where $\lambda(n)$ is the Carmichael totient function, which is defined to be the least m such that $a^m \equiv 1 \pmod{n}$ for every a relatively prime to n. It turns out here that $\lambda(n) = \text{lcm}((p-1, q-1))$. It is also acceptable to replace $\lambda(n)$ with $\phi(n)$ here, where ϕ is the Euler Totient function.

Choose an e such that $e < \lambda(n)$ and $\gcd(\lambda(n), e) = 1$. It is common to let e be $2^{16} + 1$. e is part of the public key along with n.

Determine $d = e^{-1} \pmod{\lambda(n)}$. d is the private key exponent.

Now suppose Bob wants to send a private message to Alice now. Suppose Bob's message is encoded as the integer m.

He sends it securely with the encryption function $E(m) = m^e$ which uses Alice's public key:

$$c = E(m) = m^e \pmod{n}.$$

Alice then picks up this message and decodes it with the decryption function $D(m) = m^d$

$$D(m^e) = (m^e)^d = m \pmod{n}.$$

Because the Discrete Logarithm problem is hard, Eve cannot compute m given just $c = m^e$. She also does not know how n factors, so she doesn't know $\lambda(n)$. Therefore she cannot compute the inverse of d mod $\lambda(n)$ to decode the message either.

Alice can give her public key (n, e) to Bob publicly, while keeping her private key d secret. Bob simply has to encrypt all his message with the encryption function $E(m) = m^e$ and Alice the decode them with her private key.

The term "RSA" comes from Ron Rivest, Adi Shamir, and Leonard Adleman, who publicly described this algorithm in 1977.

5 References

Conrad, Keith. Fermat's test.

https://kconrad.math.uconn.edu/blurbs/ugradnumthy/fermattest.pdf

Conrad, Keith. The Solovay-Strassen Test.

https://kconrad.math.uconn.edu/blurbs/ugradnumthy/solovaystrassen.pdf

Diffie, Whitfield, and Martin Hellman. New Directions in Cryptography. IEEE Transactions on Information Theory. 1976. https://ee.stanford.edu/ hellman/publications/24.pdf

Lynn, Ben. Primality Tests. Stanford Crypto Notes.

https://crypto.stanford.edu/pbc/notes/numbertheory/millerrabin.html

Niven, Ivan. Herbert Zuckerman, Hugh Montgomery. *The Theory of Numbers.* Fifth Edition. 1991.

Rivest, R. L., A Shamir, L Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. 1977. https://web.archive.org/web/20230127011251/http://people.csail.mit.edu/rivest/Rsapaper.pdf