Factoring and Primality Testing

Daniel Liu

April 8, 2024

Basic Algorithms

Sieve of Eratosthenes

List out all numbers up to n, and strike out all multiples of $2, 3, 5, \dots$ and so on. The left over numbers are primes.

Trial Division

Repeatedly divide n by different primes up to $\sqrt{n}$. If any prime divides n, it is composite, if not, n is prime.

Algorithms - Fermat's Little Theorem

Fermat's Little Theorem

If p is prime, and a is not a multiple of p, then $a^{p-1} = 1 \pmod{p}$

Fermat Test - Direct Application

If $a^{n-1} \neq 1 \pmod{n}$ for some a who is not a multiple of n, then n is not prime.

Miller-Rabin Test - Derived from Fermat's

We write $n - 1 = 2^s d$, and check if $a^d = 1 \pmod{n}$, or $a^{2^r d} = -1 \pmod{n}$ for some $0 \leq r < s$. If none of these are true, n is not prime.

Turns into **probabilistic** primality tests.

Galactic Algorithm - AKS

Based on Fermat's Theorem on Polynomials:

Theorem

$$(x + a)^n = x^n + a \pmod{n}$$

if and only if n is prime.

The algorithm saves time by only checking the equality for a small but specific selection of polynomials $\rightarrow$ **Fastest** theoretical deterministic primality test.
Miller-Rabin performs better in practice.

Factoring - Pollard's $p - 1$ Algorithm

From earlier:

Fermat's Little Theorem

If p is prime, and a is not a multiple of p, then $a^{k(p-1)} = 1 \pmod{p}$ for all positive integers k

From here, if $x = 1 \pmod{p}$ for a factor p of n, then both $x - 1$ and n are multiples of p, so $\gcd(x - 1, n)$ would yield a multiple of p. To maximize our odds of this happening, we choose x to be generated from taking exponents of a random base, in hopes of x looking like $a^{k(p-1)}$ for some factor p of n.

Quantum Factoring - Shor's Algorithm

Basic Idea:

Factor n

If $a^q = 1 \pmod{n}$, then by difference of squares,

$$\left(a^{\frac{q}{2}} - 1\right)\left(a^{\frac{q}{2}} + 1\right) = 0 \pmod{n},$$

and we have a chance that one of the two terms contain a non-trivial factor of n.

The task to find q is called the discrete logarithm, but Quantum Computation makes this easy $\rightarrow$ We have a factoring algorithm:

1. Find a q with a random base a so that $a^q = 1 \pmod{n}$
2. Check $\gcd(a^{\frac{q}{2}} \pm 1, n)$ to find a factor of n
3. Repeat if the last step returned n