Math Circles: Primality Testing and Integer Factorization

Owen Sharpe

University of Waterloo

April 3, 2024

Recap

Last time we discussed the following topics:

- Modular arithmetic.
- Fermat and Mersenne primes.
- The Fermat primality test.
- The Miller-Rabin primality test.

Modular Arithmetic Recap

The key ideas of modular arithmetic are as follows:

- We write $a \mid b$ if b is divisible by a.
- We write $a \equiv b \pmod{m}$ or say a is congruent to b mod m if $m \mid a - b$.
- We have $a \equiv b \pmod{m}$ if and only if there exists an integer q such that $a = b + mq$.
- If $a \equiv b \pmod{m}$, then $a + c \equiv b + c \pmod{m}$ and $ac \equiv bc \pmod{m}$.
- For a fixed m, we partition the integers into m congruence classes, where a and b lie in the same congruence class $[a] = [b]$ iff $a \equiv b$ mod m.

Modular Arithmetic Recap

Example

To calculate $654 \times 321 \pmod 4$, notice that $654 \equiv 2 \pmod 4$ and $321 \equiv 1 \pmod 4$. Thus $654 \times 321 \equiv 2 \times 1 \equiv 2 \pmod 4$.

Exercise

Calculate 432137×234073 *mod the following m:*

$$10, 100, 1000, 10000, 100000.$$

Modular Arithmetic Recap

Example

To calculate $6^{789} \pmod 7$, notice that $6 \equiv -1 \pmod 7$, so

$$6^{789} \equiv (-1)^{789} \equiv -1 \equiv 6 \pmod 7.$$

Exercise

Calculate $2^{35}, 2^{70}, 2^{140}, 2^{280}$, *and* $2^{560} \pmod{561}$.
Hint: $2^{32} = (2^{16})^2 = ((2^8)^2)^2$ *and so on. Then you can calculate* 2^{35} *as* $2^{32} \times 8$.

Coprimality

Definition (Coprimality)

We say that a and b are coprime when $\gcd(a, b) = 1$.

Example

The numbers 6 and 35 share no prime factors, so $\gcd(6, 35) = 1$ and they are coprime. On the other hand, the numbers 35 and 75 share the factor 5, so they are not coprime. As special cases, 1 is coprime to every integer and 0 is not coprime to any integer.

Exercise

Are 91 *and* 169 *coprime? Are* 97 *and* 99 *coprime?*

Modular Inverse

Definition (Modular Inverse)

We say that a and b are inverse mod m when $ab \equiv 1 \pmod{m}$. We write $b \equiv a^{-1} \pmod{m}$ in this situation. We say a is invertible mod m when such a b exists.

Proposition

a is invertible mod m if and only if a is coprime to m.

Example

3 and 7 are inverse mod 10, because $3 \times 7 \equiv 1 \pmod{10}$. There is no inverse of 6 mod 10, because 6 and 10 share the factor 2.

Exercise

Calculate inverses of 1, 2, 4, *and* 14 *mod* 15.
Hint: $14 \equiv -1 \pmod{15}$.

Example

Let's calculate the inverse of 41 mod 317. We have

$$\begin{aligned}
0 \times 41 + 1 \times 317 &= 317 \\
1 \times 41 + 0 \times 317 &= 41 \\
-7 \times 41 + 1 \times 317 &= 30 \\
8 \times 41 - 1 \times 317 &= 11 \\
-23 \times 41 + 3 \times 317 &= 8 \\
31 \times 41 - 4 \times 317 &= 3 \\
-85 \times 41 + 11 \times 317 &= 2 \\
116 \times 41 - 7 \times 317 &= 1.
\end{aligned}$$

Thus $116 \times 41 \equiv 1 \pmod{317}$, or $41^{-1} \equiv 116 \pmod{317}$.

Euler's Totient Function

Definition (Totient Function)

Let n be a positive integer. The Euler totient function is defined

$$\phi(n) = \#\{a \in \{0, 1, 2, \ldots, n-1\} : \gcd(a, n) = 1\}.$$

Example

The totient of 30, $\phi(30)$, is 8. The integers in $\{0, \ldots, 29\}$ which are coprime to 30 are

$$1, 7, 11, 13, 17, 19, 23, 29.$$

Exercise

Calculate $\phi(10)$, $\phi(15)$, *and* $\phi(17)$.

Proposition

If the prime factorization of n is

$$p_1^{e_1} \times \cdots \times p_k^{e_k},$$

then

$$\phi(n) = n \times \frac{p_1 - 1}{p_1} \times \cdots \times \frac{p_k - 1}{p_k}.$$

Example

The prime factorization of 300 is $2^2 \times 3 \times 5^2$. Thus

$$\phi(300) = 300 \times \frac{1}{2} \times \frac{2}{3} \times \frac{4}{5} = 80.$$

Exercise

Calculate $\phi(210)$, $\phi(216)$, and $\phi(257)$.
Hint: 257 is prime.

The Totient as an Exponent

Proposition

If $\gcd(a, n) = 1$, *then* $a^{\phi(n)} \equiv 1 \pmod{n}$.

Corollary (Fermat's Little Theorem)

Let p *be prime and* a *not divisible by* p. *Then* $a^{p-1} \equiv 1 \pmod{p}$.

Corollary

If $\gcd(a, n) = 1$, *then* $a^{-1} \equiv a^{\phi(n)-1} \pmod{n}$ *and* $a \equiv a^{\phi(n)+1} \pmod{n}$.

RSA Preliminaries

Proposition

Suppose n is squarefree (the exponents in the prime factorization are all 1). Then for any integer a and positive integer r, $a^{r\phi(n)+1} \equiv a \pmod{n}$.

Now consider an inverse pair $d, e \pmod{\phi(n)}$, and set $c = a^d$. Since $de = r\phi(n) + 1$, we get $c^e \equiv a \pmod{n}$. This return-to-original process forms the basis of the RSA cryptographic system.

RSA

The RSA cryptographic system (named for Rivest, Shamir, and Adleman) works as follows:

- I choose two prime numbers, p and q, secretly from everyone (including you).
- I calculate $n = pq$, $\phi(n) = (p-1)(q-1)$, and a inverse pair d, e mod $\phi(n)$.
- I release n and d publicly (the modulus and the public key), while keeping e private (the private key).
- You now wish to send me a private message. You encode it as a single number a smaller than n.
- You compute $c \equiv m^d \pmod{n}$ and send it to me.
- I compute $c^e \equiv (m^d)^e \equiv a \pmod{n}$ and read your message securely.

RSA Example

Example

Spock is on a mission in the Romulan Neutral Zone. He needs to ask Kirk how many Federation starships are patrolling the border, but he cannot allow the Romulans to read this message. Spock is very quick at mental arithmetic, and

- picks the primes $p = 401$ and $q = 293$,
- calculates $n = 117493$ and $\phi(n) = 116800$,
- chooses the inverse public/private key pair $d \equiv 1701, e \equiv 19501 \pmod{\phi(n)}$,
- and transmits the values n and d to Kirk.

Kirk knows that all 9 of the remaining Constitution-class starships are patrolling the border, so he computes and sends the single value $c \equiv 9^{1701} \equiv 69765 \pmod{n}$ back to Spock. Spock computes $69765^{19501} \equiv 9 \pmod{n}$ and uses this information to continue the mission.

RSA

Why is RSA secure?

- It is computationally difficult to compute $c^{1/d} \equiv a \pmod{n}$ unless a hostile party knows e.
- It is not possible to calculate e, even given d, without knowing $\phi(n)$.
- It is not possible to calculate $\phi(n) = \phi(pq) = (p-1)(q-1)$ without knowing p and q.
- It is computationally difficult to compute p, q only knowing n.

RSA

Why is RSA easy to use?

- It is easy for me to generate two prime numbers p, q and then multiply them. I can use the Miller-Rabin test (discussed last week) to quickly check numbers with hundreds of digits for primality with almost 100% confidence.
- It is easy for you to calculate $a^d \pmod{n}$.

For a physical analogy, you can imagine that I have given you an open box with a keyhole. You put your message inside, and once you close the box, it locks automatically and becomes impossible for even you to open again. You return the box to me and I use my key to open it and read your message.

Attacks on RSA

How could a malicious actor attack RSA? The most obvious method is to successfully factor n, upon which calculating $\phi(n)$ and e are easy.

Example

The Romulans notice that if they try to trial factor 117493, they only need to try dividing by primes up to $\sqrt{117493} \approx 343$. There are only 68 such primes, so they enumerate all of them and eventually hit upon $q = 293$ as a factor. Thus $p = 401$ is the other factor, and they compute $\phi(117493) = 116800$, $e \equiv d^{-1} \equiv 19501 \pmod{116800}$, and finally $69765^{19501} \equiv 9 \pmod{117493}$. They send 20 birds-of-prey to the border to follow the 9 starships and deter the Federation's plans.

Ranges for RSA

Modern RSA uses numbers with hundreds of digits. For example, the open-source OpenSSH library generates RSA "keys" with about 1000 decimal digits by default.

It is out of the range of computers to trial factor numbers this large - one would need a list of the primes up to about 10^{500}, of which there are more than 10^{495}. Nevertheless, mathematicians have succeeded in factoring RSA numbers of 250 digits using other algorithms.

Pollard's Rho Algorithm

- Let n be a composite number, and d a non-trivial factor of n.
- Let c be a random integer and $g(x)$ be the polynomial $x^2 + c$.
- Let x_0 be a random integer and recursively define $x_{i+1} = g(x_i)$.
- Since there are only d congruence classes mod d, this sequence must repeat eventually mod d. Let $j < k$ be the least values such that $x_j \equiv x_k \pmod{d}$.
- The value $\gcd(x_j - x_k, n)$ is divisible by d and thus either n or a non-trivial factor of n.
- The algorithm iteratively generates terms of the sequence mod n, checking specifically $\gcd(x_{2i} - x_i, n)$ for each i. If it finds a factor, it terminates; else it will eventually loop mod n (where $x_{2i} \equiv x_i \pmod{n}$) and terminate at this point.

Pollard's Rho Algorithm Considerations

- The polynomial $x^2 + c$ is thought to be sufficient to make x_i behave like a random sequence.
- Assuming that the x_i's are sufficiently random, it is likely that the first pair x_i, x_j congruent mod d will *not* be congruent mod n, and thus that $\gcd(x_j - x_k, n)$ will be a proper divisor of n.
- It is thought that exceeding $\sqrt{d}$ iterations is generally enough to find a non-trivial factor. If d is taken to be the smallest prime dividing n, then $d \leq \sqrt{n}$ and about $n^{1/4}$ iterations are needed.

Example

Example

Let's use this to factor the 6-th Fermat number, $2^{2^6}+1 = 18446744073709551617$. I set $x_0 = 1$, and after 808 iterations, my Python script discovers the factor 274177. This is the 23974-th prime, and had we used trial factoring, that is how many iterations it would take to find this factor.

Other Primality Tests / Factorization Algorithms

Let's conclude this three-week series on primality testing and factorization with a list of some state-of-the-art algorithms.

- The AKS (Agrawal-Kayal-Saxena) primality test determines whether n is prime by checking a congruence of *polynomials*.
- The Pollard $p-1$ factoring algorithm is especially fast when factoring n such that $n-1$ has many small factors.
- The general number field sieve has been used recently to factor RSA moduli with up to 250 digits.
- If quantum computers with enough qubits can be built, Shor's algorithm would make RSA factoring with 1000 digit numbers feasible.