Number Theory: III

Advanced Topics

李昂生

Discrete Mathematics
U CAS
19th April, 2018

Outline

1. Fingerprinting
2. Hashing
3. *Error correcting code
4. *Locally testable code
5. Cryptography
6. *Primality test
7. *Advanced reading

General view

- Applications
- Research projects
- New achievements
- The challenges for the future

Chinese Remainder Theorem Revisit

- For every prime p and a natural number number k, we have finite fields

$$\mathrm{GF}(p) = \mathbb{Z}_p$$

$$\mathrm{GF}(p^k) = \mathbb{Z}_p^k = \mathbb{Z}_p \times \cdots \times \mathbb{Z}_p$$

- For every natural number n, suppose that $n = p_1^{k_1} p_2^{k_2} \cdots p_l^{k_l}$, then

$$\mathbb{Z}_n \cong \mathrm{GF}(p_1^{k_1}) \times \cdots \times \mathrm{GF}(p_l^{k_l}).$$

These provide the universe for computer science.

The mechanism of fingerprinting

Question: Given a universe U, decide whether or not two elements x, y in U are identical.
The fingerprinting mechanism is:
To pick a random mapping R from U to a small set V such that for any $x, y \in U$,
Completeness: If $x = y$, then,

$$R(x) = R(y),$$

Soundness: If $x \neq y$, then with high probability,

$$R(x) \neq R(y).$$

Matrices product

Let $\mathbb{F}$ be a finite field, $\mathbb{Z}_p$ for some prime, p say. Let A, B and C be $n \times n$ matrices over $\mathbb{F}$.
To test whether or not $AB = C$, naive approach is to compute the matrix product and compare - in time complexity $O(n^3)$.
By fingerprinting, we test as follows:
Tester $\mathcal{T}$:

(1) Let r be a vector chosen randomly and uniformly from $\{0, 1\}^n$ (of course could be any other field, $\mathbb{F}^n$ say)

(2) Let $x = Br$, $y = Ax$ and $z = Cr$.
(Time complexity $O(n^2)$.)

(3) If $y = z$, then accepts, and rejects, otherwise.

Proof

If $AB = C$, then $\mathcal{T}$ accepts with probability 1.
Suppose that $AB \neq C$. Let $D = AB - C = (d_{ij})$. Suppose $d_{11} \neq 0$. For the random vector

$$r = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix} \tag{1}$$

If $Dr = 0$, then $d_{11}r_1 + d_{12}r_2 + \cdots + d_{1n}r_n = 0$, giving

$$r_1 = -\frac{d_{12}r_2 + \cdots + d_{1n}r_n}{d_{11}}, \tag{2}$$

which occurs with probability at most $\frac{1}{2}$.
Therefore, the probability that $\mathcal{C}$ accepts is at most $\frac{1}{2}$.

The fingerprints

- x, y and z are the fingerprints that generated by the random vector r.
- If r can be chosen from $\mathbb{F}^n$, then the probability of the error is reduced to

$$\frac{1}{p}.$$

- By repeating k times, the probability that an error occurs is reduced to $\frac{1}{2^k}$.
- Is there tester that uses less time, say $O(n)$, or even $O(\log n)$?
  Research project.

Polynomial identity test

Theorem 1

Let $\mathbb{F}$ be a finite filed, and $Q(x_1, \cdots, x_n) \in \mathbb{F}[x_1, \cdots, x_n]$ be a multivariate polynomial of total degree d over $\mathbb{F}$. Let $S \subset \mathbb{F}$, and let $r_1, \cdots, r_n$ be chosen independently and uniformly at random from S. Then,

$$\Pr[Q(r_1, \cdots, r_n) = 0 \mid Q(x_1, \cdots, x_n \not\equiv 0] \leq \frac{d}{|S|}. \tag{3}$$

Proof

If $Q(x_1, \cdots, x_n) \equiv 0$, then the probability that $Q(r_1, \cdots, r_n) = 0$ is 1.
Suppose that $Q \not\equiv 0$.
By induction on n. $n = 1$, done before. Suppose the theorem holds for all $n' < n$ and $n > 1$.
Let

$$Q(x_1, x_2, \cdots, x_n) = \sum_{i=0}^{k} x_1^i Q_i(x_2, \cdots, x_n), \tag{4}$$

for $k > 0$.
By the choice of k, the coefficient $Q_k(x_2, \cdots, x_n)$ of x_1^k is not identically zero, and the total degree of Q_k is $d - k$.

Proof - continued

By inductive hypothesis,

$$\Pr[Q_k(r_2, \cdots, r_n) = 0] \le \frac{d-k}{|S|}. \tag{5}$$

Assume $Q_k(r_2, \cdots, r_n) \neq 0$. Let

$$q(x_1) = \sum_{i=0}^{k} x_1^i Q_i(r_2, \cdots, r_n).$$

Then

$$\Pr[q(r_1) = 0] \le \frac{k}{|S|}. \tag{6}$$

Therefore,

$$\Pr[Q(r_1, r_2, \cdots, r_n) = 0] \le \frac{d-k}{|S|} + \frac{k}{|S|} = \frac{d}{|S|}.$$

Identity of data

Alice and Bob share the data D initially. During the procedure of processing, the data may be corrupted. So they want to make sure that their data A and B are are same.
However, the data A and B are huge, for which verification of equality is not easy.
By fingerprinting, we may check easily as follows:

1. To transform A and B to $a = (a_1, \cdots, a_n)$ and $b = (b_1, \cdots, b_n)$ of numbers in a universe $\mathbb{F}^n$.
2. For a prime p, define the fingerprint by

$$f_p(x) = x \bmod p. \tag{7}$$

3. Randomly pick a prime p,
   if $f_p(a) = f_p(b)$, then accept, and reject, otherwise.

Arguments

- There are many primes within a number n ($\approx \frac{n}{\ln n}$, prime number theorem)
  Here we need to decide whether or not a given number x is a prime.
- For every n, there is only a small number of prime factors of n ($\log_2 n$, **why?**).
- If $a = b$, the tester accepts with probability 1, and if $a \neq b$, the tester accepts with only a small probability. Using Chinese reminder theorem.

General ideas of fingerprinting

- Characterise the two objects as polynomials A and B
- Randomly and uniformly choose a random number r in $\mathbb{Z}_p$, written $r \in_{\mathrm{R}} \mathbb{Z}_p$.
- The fingerprints is $A(r)$ and $B(r)$ for random r, in a field $\mathbb{Z}_p$ for some prime p
- If $A \equiv B$, then accepts with probability 1, and if $A \not\equiv B$, the probability of acceptance is at most $\frac{k}{p}$.
- The n-bit comparison is reduced to compare only $O(\log n)$ bits.

The idea of hashing

The idea of hash table is again the fingerprinting of the following form:

1) Given n-bit integers a and b
2) Fix a prime $p > 2^n$
3) Pick randomly and uniformly a polynomial P
4) Compute and compare $P(a)$ and $P(b)$ in $\mathbb{Z}_p$.

The questions

1. Given a set of *keys* S, organise S into a data structure that supports efficient processing of finding queries and updating operations,
   Remark: Classically, it is a balanced binary tree, allowing $O(\log n)$ time of operations of query, insertion and deletion etc.
2. To build a data structure dynamically by basic operations of insertion and deletion that supports efficient operations.
3. Classical data structure has optimum complexity $O(\log n)$.
4. Hash tables break the lower bound of $O(\log n)$ to $O(1)$.

The crucial new idea

Random Access Machine (RAM): For a set S of keys,

- Create a table of size $O(|S|)$
- Find a query by random access to the Hash Table T by a hash function h, of time complexity $O(1)$, - just directly query the table
- Create a secondary hash table (or backup hash table) T', when collision occurs
- Collisions occur only $O(1)$ many times.

Hash Table

(i) It is a table T of n cells, indexed by

$$N = \{0, 1, \cdots, n-1\}.$$

(ii) A hash function is a function of the form:

$$h : \ M \to N,$$

where $M = \{0, 1, \cdots, m-1\}$ and $m >> n$.

(iii) Each cell in table T allows to encode an element of M, i.e., with size $\log m$.

(iv) The hash function is a fingerprint function for the keys in a large set M to the small set N of fingerprints (cells)

(v) Fingerprint function h ensures that for distinct keys $x \neq y$, the probability that the cells $h(x)$ equals $h(y)$, i.e., $h(x) = h(y)$, is small, so that collisions occur with a only small probability

Formal description of a hash table

Given a fingerprint function

$$h : M \to N, \tag{8}$$

which is the hash function.
Therefore,

$$\text{fingerprinting function} \;=\; \text{hashing function}. \tag{9}$$

The finding operation proceeds as follows:

1) Store each key $k \in S$ at the location $h(k)$ in T, i.e., $T[h(k)] = k$.
2) To search for a key q, we only need to check if $T[h(q)] = q$.

Resolving collisions

By the same reason as the proofs for fingerprinting, we know that collisions occur only a small number of times. However, nevertheless, collisions are unavoidable.

To resolve this issue, we introduce the *secondary hash table* or *backup hash table*.

We will ensure that, a constant number of backup hash tables are sufficient.

The construction of hash functions

Fix m and n. Choose a prime $p \geq m$. We will work over the field $\mathbb{Z}_p$.

1. Let $g: \mathbb{Z}_p \to \mathbb{N}$ be the function

$$g(x) = x \bmod n, \tag{10}$$

   for some small number n, - the length of the hash table.

2. Define

$$f_{a,b}(x) = ax + b \bmod p. \tag{11}$$

$$h_{a,b}(x) = g(f_{a,b}(x)). \tag{12}$$

3. Let $H = \{h_{a,b} \mid a, b \in \mathbb{Z}_p, a \neq 0\}$. Then H is a family of hash functions.

The challenges

- Are the classical data structures including the hash tables sufficient for processing big data?
- If yes, prove, if no, what is the theory of big data structure?

Error correcting code

Definition 2

For $x, y \in \{0,1\}^m$, the fractional Hamming distance of x and y, written, $\Delta(x, y)$ is defined ny

$$\frac{1}{m}|\{i : \ x_i \neq y_i\}|.$$

For $\delta \in [0,1]$, $E : \ \{0,1\}^n \to \{0,1\}^m$, E is called an *error correcting code with distance δ*, if for every $x \neq y$,

$$\Delta(E(x), E(y)) \geq \delta. \tag{13}$$

$E(x)$: the *codeword* of x.

Intuition of ECC

Why ECC?

- To increase slightly the dimensionality allows us to amplify errors largely
- To amplify errors is to rectify the errors.
- Increasing errors amplifies hardness.

Existence of ECC

Lemma 3

For every $\delta < \frac{1}{2}$ and large n, there is a function $E: \{0,1\}^n \to \{0,1\}^m$ that is an ECC with distance δ for $m = n/(1 - H(\delta))$, where $H(\delta) = -\delta \log_2 \delta - (1-\delta)\log_2(1-\delta)$, the Shannon entropy of δ.

Proof

Each δ-ball in $\{0,1\}^m$ contains at most $o(1) \cdot 2^{H(\delta)n}$ elements.
$m = n/(1 - H(\delta))$, there are at least 2^n many δ-balls in $\{0,1\}^m$.
Random enumeration of the δ-balls will define an ECC E with distance δ.

High-dimensional geometry

The math principle of ECC is a high-dimensional geometry theorem:
The volume of a ball of radius r in m-dimensional space is approximately

$$\frac{\pi^{m/2}}{(m/2)!} r^m.$$

The volume increases exponentially as the dimensionality increases.

Efficient ECC

We will need explicitly defined ECC that are both efficiently encoded and decoded.

Decoding an ECC:
If $\Delta(E(x), y) < \frac{\delta}{2}$, then efficiently compute x.

Walsh-Hadamard code

The Walsh-Hadamard code of $u = (u_1, u_2, \cdots, u_n)$ is the function of the following form:

$$WH(x_1, x_2, \cdots, x_n) = u_1 x_1 + u_2 x_2 + \cdots + u_n x_n \qquad (14)$$

It is a function from $\{0,1\}^n$ to $\{0,1\}^{2^n}$, written WH.

Lemma 4

WH is an ECC of distance $\frac{1}{2}$.

ECC over Σ

Given alphabet Σ, $x, y \in \Sigma^m$,

$$\Delta(x, y) = \frac{1}{m}|\{i : x_i \neq y_i\}|.$$

A function $E : \Sigma^n \to \Sigma^m$ is an ECC with distance δ over Σ if for $x \neq y$, $\Delta(E(x), E(y)) \geq \delta$.

Reed-Solomon code

Let $\mathbb{F}$ be a field and n, m numbers with $n \leq m \leq |\mathbb{F}|$. The Reed-Solomon code is

$$\begin{aligned} RS: \quad \mathbb{F}^n &\rightarrow \mathbb{F}^m \\ (a_0, a_1, \cdots, a_{n-1}) &\mapsto (z_0, z_1, \cdots, z_{m-1}), \end{aligned}$$

where $z_j = \sum_{i=0}^{n-1} a_i f_j^i$, f_j is the jth element of $\mathbb{F}$.

Let

$$A(x) = \sum_{i=0}^{n-1} a_i x^i. \tag{15}$$

Then $z_j = A(f_j)$.

RS lemma

Lemma 5

The Reed-Solomon code $RS : \mathbb{F}^n \to \mathbb{F}^m$ *has distance* $1 - \frac{n}{m}$.

Lagrange interpolation

For any set of pairs $(a_1, b_1), \cdots, (a_{d+1}, b_{d+1})$, there exists a unique polynomial $g(x)$ of degree at most d such that $g(a_i) = b_i$, for each $i \in \{1, 2, \cdots, d+1\}$.

Proof.

$$g(x) = \sum_{i=1}^{d+1} b_i \frac{\prod_{j \neq i}(x - a_j)}{\prod_{j \neq i}(a_i - a_j)}. \tag{16}$$

□

Unique decoding for Reed-Solomon

Theorem 6

There is a polynomial time algorithm that, given a list $(a_1, b_1), \cdots, (a_m, b_m)$ of pairs of elements of a finite field $\mathbb{F}$ such that there is a unique degree d polynomial $G : \mathbb{F} \to \mathbb{F}$ satisfying $G(a_i) = b_i$ for t of the numbers $i \in [m]$, where $t > \frac{m}{2} + \frac{d}{2}$, recovers G.

Let $t \geq \frac{m}{2} + \frac{d}{2} + 1$, let $L = \frac{m}{2} + \frac{d}{2}$, and $l = \frac{m}{2} - \frac{d}{2}$.
Set

$$C(x) = c_0 + c_1 x + \cdots + c_L x^L$$

$$E(x) = e_0 + e_1 x + \cdots + e_{l-1} x^{l-1} + e_l x^l$$

Proofs

For each $i \in [m]$, set

$$C(a_i) = b_i E(a_i)$$

This is a homogenous system of linear equations with m equations, and $m + 2$ unknowns. It must have nonzero solutions.
Solving the system, get polynomials $C(x)$ and $E(x)$.
Consider $C(x) - G(x)E(x)$.
The degree of the polynomial is $\frac{m}{2} + \frac{d}{2}$. However, for every i, if $G(a_i) = b_i$, then $C(a_i) - G(a_i)E(a_i) = 0$, for which the number of such i's is $t \geq \frac{m}{2} + \frac{d}{2} + 1$.
Therefore $C(x) - G(x)E(x) \equiv 0$. Set $P = \frac{C(x)}{E(x)}$. Then

$$P \equiv G.$$

Decoding and hardness amplification

Decoding: Given a string $x \in \{0,1\}^n$,

$$x \Rightarrow E(x) \Rightarrow \text{ corrupted } E(x) \Rightarrow x \tag{17}$$

Decoding ECC

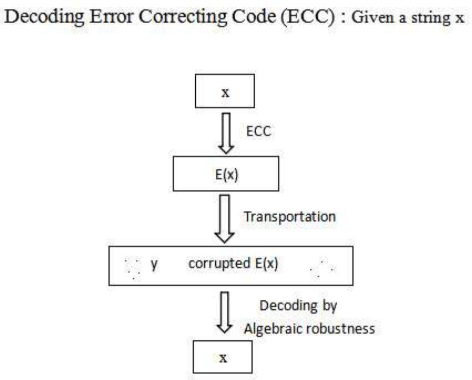

Figure: Decoding error correcting code.

Local decoder

Let $E : \{0,1\}^n \to \{0,1\}^m$ be an ECC and let ρ and q be some numbers. A *local decoder* for E handling ρ errors is an algorithm D, that given random access to a string y this is ρ-close to some codeword $E(x)$ for some unknown x, and an index j, runs for poly log m time and outputs x_j with probability at least $\frac{2}{3}$.

$$x \Rightarrow E(x) \Rightarrow y : \text{corrupted } E(x) \Rightarrow x_j$$

- The decoder D is allowed to randomly read some bits of y only, the corrupted $E(x)$, where x is an unknown.
- The running time of D is poly log m, m is the length of y.

Hardness amplification from local decoder

Worst case hardness to mildly average case hardness: Given a function $f: \{0,1\}^n \rightarrow \{0,1\}$, interpreted as a string,

$$f \Rightarrow E(f) \Rightarrow \text{ computes } f \text{ with prob } 1-\rho \Rightarrow \text{perfectly computes } f \tag{18}$$

Local decoder for WH

Theorem 7

For $\rho < \frac{1}{4}$, there exists a local decoder for the Walsh-Hadamard code handling ρ errors.

Input:
(i) $j \in [n]$,
(ii) random access to $f: \{0,1\}^n \to \{0,1\}$, where

$$\Pr_{y \in_R \{0,1\}^n}[f(y) \neq a \cdot y] \leq \rho \tag{19}$$

for $\rho < \frac{1}{4}$ and some unknown a.
Output: A bit b, that is expected to be a_j.

D for WH

The local decoder D proceeds:

1) Let e^j be the vector that is 1 in the jth bit, and 0 on the all other bits
2) Randomly pick $y \in \{0, 1\}^n$
3) Query f for $f(y)$ and $f(y + e^j)$
4) Output $b = f(y) + f(y + e^j) \pmod 2$.

Then:

$$\begin{aligned} f(y) &= x \cdot y \text{ with prob } 1 - \rho \\ f(y + e^j) &= x \cdot (y + e^j) \text{ with prob } 1 - \rho. \end{aligned}$$

So with prob $1 - 2\rho$, $b = a_j$.
Run several times, then with prob almost 1, $b = a_j$.

Computing the correct $f(x)$ from a corrupted f

Compute $f(x)$ as follows:

1. Randomly pick y
2. Let $b = f(y) + f(y + x) \pmod 2$

Then with prob $1 - 2\rho$, b is the correct value of $f(x)$.
We say that f has the *self-correction property*.

Private key

Suppose that we encode a text by

$$f(x) = x + k \pmod{p}, \tag{20}$$

for some prime p and some $k \in \mathbb{Z}_p$.
The decoding of f is simply

$$f^{-1}(y) = y - k \pmod{p}. \tag{21}$$

In this case, a text x is encoded and decoded by the following form:

$$x \Rightarrow x + k \bmod \; \Rightarrow x \bmod p.$$

Here k is the private key.

RSA

Suppose that $n = pq$ for some primes p, q and $p \neq q$.
Suppose that d and e are numbers satisfying:

$$de = 1 + k(p-1)(q-1), \tag{22}$$

for some integer k.
Then the encode is

$$E : M \rightarrow C = M^e \mod n. \tag{23}$$

The decoding - 1

The decoding is:

$$D : C^d \bmod n. \tag{24}$$

We prove that $C^d \equiv M \pmod{n}$.

$$\begin{aligned} C^d &= M^{ed} \\ &= M^{1+k(p-1)(q-1)} \\ &= M \cdot (M^{(p-1)})^{k(q-1)} \pmod{p} \\ &= M \pmod{p}. \end{aligned}$$

The decoding - 2

$$
\begin{aligned}
C^d &= M^{ed} \\
&= M^{1+k(p-1)(q-1)} \\
&= M \cdot (M^{(q-1)})^{k(p-1)} \pmod{q} \\
&= M \pmod{q}.
\end{aligned}
$$

Since $(p, q) = 1$, by the Chinese Remainder Theorem,

$$C^d \equiv M \pmod{n}. \tag{25}$$

Public key

- n can be public
- one of the e and d can be public
- Both p, q are kept for privacy.
- One of e and d is kept for privacy.

The Assumption

1) Finding one of d (or e) from the public e (or d) is hard, without given p and q,
2) Finding the prime factors p, q for n is hard.

Key Agreement Protocol

(1) Alice and Bob agreed a prime p and its primitive root a.

(2) Alice chooses a secret number k_1 and sends $a^{k_1} \mod p$ to Bob.

(3) Bob chooses his own key k_2 and sends a^{k_2} to Alice

(4) Alice computes

$$(a^{k_2})^{k_1} \equiv a^{k_1 k_2} \mod p.$$

(5) Bob computes

$$(a^{k_1})^{k_2} \equiv a^{k_2 k_1} \mod p.$$

(6) Alice and Bob Achieved their shared key:

$$a^{k_1 k_2} \mod p.$$

M. O. Rabin, Probabilistic algorithm for testing primality. J. Number Theory, 12, pp 128 -138, 1980.

This is the first nontrivial randomized algorithm. Rabin was awarded Turing award due to this work.

The algorithm is currently used for primality test in practice.

Overview

Case 1 p prime
$\mathbb{Z}_p$ is a field
Case 2 n is a composite
$\mathbb{Z}_n$ is not a field
Can we use this difference to decide whether or not a given number n is prime?

Fermat Test

Let n be a natural number.
Case 1 If n is prime, then for all non-zero $a \in \mathbb{Z}_n$),
$a^{n-1} = 1 \mod n$
Case 2 If n is a composite, then there are “many" non-zero $a \in \mathbb{Z}_n$,
$a^{n-1} \neq 1 \mod n$
If in case 2, there is half or $\frac{1}{3}$ of the residues a such that $a^{n-1} \neq 1 \mod n$, then this gives a simple algorithm to test whether a natural number n is prime or not.
Unfortunately, this is not the case.

Square roots modulo p

p is always prime.
Considering

$$x^2 \equiv a \pmod{p}, \tag{26}$$

we have

1) it has at most two roots, (The Algebraic Fundamental Theorem)
2) If r is a root, so are $\pm r$.

Therefore, $x^2 \equiv a \pmod{p}$ either has no solution, or has two solutions.

$x^2 \equiv a \pmod{p}$

Lemma 8

If $a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$, then $x^2 = a$ has two solutions in $\mathbb{Z}_p$, and if $a^{\frac{p-1}{2}} \not\equiv 1 \pmod{p}$, then $x^2 = a$ has no solution in $\mathbb{Z}_p$.

Proof.

In $\mathbb{Z}_p$, $a^{p-1} = 1$, so $a^{\frac{p-1}{2}} = \pm 1$. If $a^{\frac{p-1}{2}} = -1$, and $x^2 = a$, then $-1 = a^{\frac{p-1}{2}} = (x^2)^{\frac{p-1}{2}} = x^{(p-1)}$, absurd.
Let r be a primitive root modulo p, and $a = r^i$ for some $i \le p - 1$.
Case 1 $i = 2j$.
In this case,

- $a^{\frac{p-1}{2}} = 1$
- $(r^j)^2 = a$
- $(r^{j+\frac{p-1}{2}})^2 = a$

all hold in $\mathbb{Z}_p$. Both r^j and $r^{j+\frac{p-1}{2}}$ are the roots of $x^2 = a$.

Proof - continued

Case 2 $i = 2j + 1$.

$$a^{\frac{p-1}{2}} = r^{\frac{p-1}{2}} \neq 1, \text{ hence } = -1$$

due to the fact that r is a primitive root.
$x^2 = a$ has no solution.

Legendre symbol

Therefore, in $\mathbb{Z}_p$,

$$\exists x[x^2 = a] \iff a^{\frac{p-1}{2}} = 1. \tag{27}$$

Because $(a^{\frac{p-1}{2}})^2 = a^{p-1} = 1$, $a^{\frac{p-1}{2}} = \pm 1$.
Therefore, $a^{\frac{p-1}{2}}$ indicates whether or not a is a perfect square modulo p. The *Legendre symbol* of a and p is defined by

$$(a|p) = a^{\frac{p-1}{2}} \mod p. \tag{28}$$

$$(ab|p) = (a|p)(b|p). \tag{29}$$

Gauss's Lemma

Lemma 9

Let p, q be odd primes. Then:

$$(q|p) = (-1)^m, \tag{30}$$

where m is the number of residues in the set

$$R = \{q \bmod p, 2q \bmod p, \cdots, \frac{p-1}{2}q \bmod p\}, \tag{31}$$

that are greater than $\frac{p-1}{2}$.

Proof of Gauss's lemma -I

(1) All residues in R are distinct.

Proof.

Let $b \leq a \leq \frac{p-1}{2}$.
If $aq - bq \equiv 0 \pmod{p}$, then

$$(a-b)q \equiv 0 \pmod{p}.$$

Therefore,

$$p|(a-b), \text{since } (p,q) = 1$$

This shows that $a = b$. $\square$

Proof of Gauss's lemma -II

(2) There are no two residues in R that add up to p.

Proof.

Towards a contradiction, let

$$aq \bmod p + bq \bmod p = p.$$

Then $(a+b)q \equiv 0 \pmod{p}$.
This gives $p|(a+b)$. However, $2 \leq a+b \leq p-1$. A contradiction. $\square$

Proof of Gauss's lemma -III

Let $X = \{x \in R \mid x \le \frac{p-1}{2}\}$, $Y = \{x \in R \mid x > \frac{p-1}{2}\}$, and $\widehat{Y} = \{p - y \mid y \in Y\}$.
Then $R = X \cup Y$, $\widehat{Y} = \{-y \mid y \in Y\}$ and

$$X \cup \widehat{Y} = \{1, 2, \cdots, \frac{p-1}{2}\}. \tag{32}$$

Multiplying the elements of the two equal sets above, we have

$$\frac{p-1}{2}!q^{\frac{p-1}{2}} = \frac{p-1}{2}!(-1)^m \pmod p,\ m = |Y|. \tag{33}$$

Therefore,

$$q^{\frac{p-1}{2}} \bmod p = (-1)^m,\ \text{giving } (a|p) = (-1)^m.$$

Legendre's law

Lemma 10

Let p, q are distinct odd primes. Then,

$$(q|p) \cdot (p|q) = (-1)^{\frac{p-1}{2} \cdot \frac{q-1}{2}}. \tag{34}$$

Consider $(q|p)$.
Let R, X, Y and $\widehat{Y}$ be the same as that in the proof of the Gauss's lemma.
As before, we have $X \cup Y = R$, and $X \cup \widehat{Y} = \{1, 2, \cdots, \frac{p-1}{2}\}$.

Proof of Legendre's law - I

Considering $R = X \cup Y$, we have

$$\sum_{i=1}^{\frac{p-1}{2}} iq = \sum_{i=1}^{\frac{p-1}{2}} \lfloor \frac{iq}{p} \rfloor \cdot p + \sum_{x \in X} x + \sum_{y \in Y} y. \tag{35}$$

By $X \cup \widehat{Y} = \{1, 2, \cdots, \frac{p-1}{2}\}$, we have

$$\sum_{x \in X} x + mp - \sum_{y \in Y} y = \sum_{x \in X} x + \sum_{y \in Y} (p - y) = \sum_{i=1}^{\frac{p-1}{2}} i, \tag{36}$$

Proof of Legendre's law - II

Summing up the two equations, and taking the modulo 2,

$$\sum_{i=1}^{\frac{p-1}{2}} iq + \sum_{x\in X} x + mp - \sum_{y\in Y} y = \sum_{i=1}^{\frac{p-1}{2}} \lfloor \frac{iq}{p} \rfloor \cdot p + \sum_{x\in X} x + \sum_{y\in Y} y + \sum_{i=1}^{\frac{p-1}{2}} i,$$

$$\Rightarrow q \sum_{i=1}^{\frac{p-1}{2}} i + mp = p \sum_{i=1}^{\frac{p-1}{2}} \lfloor \frac{iq}{p} \rfloor + \sum_{i=1}^{\frac{p-1}{2}} i$$

$$\Rightarrow mp = p \sum_{i=1}^{\frac{p-1}{2}} \lfloor \frac{iq}{p} \rfloor - (q-1) \sum_{i=1}^{\frac{p-1}{2}} i$$

$$\Rightarrow mp \equiv p \sum_{i=1}^{\frac{p-1}{2}} \lfloor \frac{iq}{p} \rfloor \Rightarrow m \equiv \sum_{i=1}^{\frac{p-1}{2}} \lfloor \frac{iq}{p} \rfloor \mod 2.$$

Proof of Legendre's law - III

Let $y = \frac{q}{p}x$ be a linear equation.
Since $(q, p) = 1$, there is no integer solution of the equation.
$\sum_{i=1}^{\frac{p-1}{2}} \lfloor \frac{iq}{p} \rfloor$ is the number of pairs (x, y) of positive integers x, y that are below the line $y = \frac{q}{p}x$, restricted to $x \leq \frac{p-1}{2}$.
By the same proof,
$(p|q) = (-1)^n$, where n is the number of pairs (x, y) of positive integers that are above the line $y = \frac{q}{P}x$ restricted to $y \leq \frac{q-1}{2}$.
Therefore,

$$(q|p) \cdot (p|q) = (-1)^{\frac{p-1}{2} \cdot \frac{q-1}{2}}.$$

Jacobi symbol

Definition 11

For $N = q_1 \cdots q_n$ for primes $q_1, \cdots, q_n$, define the *Jacobi symbol* by

$$(M|N) = \prod_{i=1}^{n} (M|q_i). \tag{37}$$

Jacobi rules

Lemma 12

For natural numbers M, N, M_1, M_2,

(1)

$$(M_1 M_2 | N) = (M_1 | N)(M_2 | N).$$

(2)

$$(M + N | N) = (M | N).$$

(3) *If M, N are odd numbers, then*

$$(M | N)(N | M) = (-1)^{\frac{M-1}{2} \cdot \frac{N-1}{2}}.$$

Proof is easy.

$(2|N)$

Lemma 13

For natural number N, if N is odd, then

$$(2|N) = (-1)^{\frac{N^2-1}{8}}. \tag{38}$$

Proof.

For p prime, consider

$$R = \{1 \cdot 2 \bmod p, 2 \cdot 2 \bmod p, \cdots, \frac{p-1}{2} \cdot 2 \bmod p\}.$$

For $N = mn$, by definition.

$\square$

$(2|N)$ - Proof - 1

Proof.

For p odd prime, consider

$$R = \{1 \cdot 2 \bmod p, 2 \cdot 2 \bmod p, \cdots, \frac{p-1}{2} \cdot 2 \bmod p\}.$$

Define

$$X = \{x \in R \mid x \leq \frac{p-1}{2}\}$$

$$Y = \{y \in R \mid y > \frac{p-1}{2}\}.$$

$$\widehat{Y} = \{p - y \mid y \in Y\}.$$

Then

$$X \cup \widehat{Y} = \{1, 2, \cdots, \frac{p-1}{2}\}$$

$(2|N)$ - Proof - 2

Proof.

Let $|Y| = m$.

$$2^{\frac{p-1}{2}} = (-1)^m \bmod p$$

According to $X \cup Y = R$, we have:

$$\sum_{x \in X} x + \sum_{y \in Y} y = 2(1 + 2 + \cdots + \frac{p-1}{2}) = 0 \bmod 2$$

This implies that

$$\sum_{x \in X} x = \sum_{y \in Y} y \bmod 2$$

□

$(2|N)$ - Proof - 3

Proof.

According to $X \cup \widehat{Y} = \{1, 2, \cdots, \frac{p-1}{2}\}$,

$$\sum_{x \in X} x + \sum_{y \in Y} (p - y) = 1 + 2 + \cdots + \frac{p-1}{2} \mod 2$$

Therefore,

$$\begin{aligned} m &= mp \mod 2 \\ &= -\sum_{x \in X} x + \sum_{y \in Y} y + (1 + 2 + \cdots + \frac{p-1}{2}) \mod 2 \\ &= 1 + 2 + \cdots + \frac{p-1}{2} \mod 2 \\ &= \frac{p^2 - 1}{8} \mod 2 \end{aligned}$$

Algorithm for computing $(M|N)$

Lemma 14

There exists an algorithm, that computes $(M|N)$ for natural numbers M, N, in time $O(l^3)$, where $l = \log M + \log N$.

By the Jacobi rules, and the Euclidean algorithm.

Characterisation Theorem

Theorem 15

(1) *If n is a prime number, then for every $m \in \Phi(n)$,*

$$(m|n) = m^{\frac{n-1}{2}} \bmod n.$$

(2) *If n is a composite, then there is an $m \in \Phi(n)$ such that*

$$(m|n) \neq m^{\frac{n-1}{2}} \bmod n.$$

(3) *If n is composite, there are at least half of $m \in \Phi(n)$,*

$$(m|n) \neq m^{\frac{n-1}{2}} \bmod n.$$

Proof of the characterisation theorem - I

(1) is by definition.
For (2). Let $n = p_1^{k_1} p_2^{k_2} \cdots p_l^{k_l}$ for distinct primes $p_1, \cdots, p_l$, where $k_1, \cdots, k_l$ are all greater than or equal to 1, and $l \geq 2$.
Case 1 There exists an i such that $k_i = 1$.
Suppose $k_1 = 1$.
Let $r \in \Phi(p_1)$ such that $(r|p_1) = -1 \bmod p_1$.
By the Chinese Remainder Theorem, there is an $m \in \Phi(n)$ such that

$$\begin{cases} m \equiv r \pmod{p_1} \\ m \equiv 1 \pmod{p_2^{k_2}} \\ \cdots \\ m \equiv 1 \pmod{p_l^{k_l}}. \end{cases} \tag{39}$$

Proof of the characterisation theorem - II

Then

$$
\begin{aligned}
(m|n) &= \prod_{i=1}^{l}(m|p_i^{k_i}) \\
&= (r|p_1)(1|p_2^{k_2})\cdots(1|p_l^{k_l}) = -1. \qquad (40)
\end{aligned}
$$

Suppose that $m^{\frac{n-1}{2}} = (m|n) = -1 \bmod n$. Then there is a t such that

$$m^{\frac{n-1}{2}} + 1 = nt,$$

giving

$$1 + 1 = 0 \pmod{p_2^{k_2}},$$

which is impossible since $p_2 > 2$.

Proof of the characterisation theorem - III

Case 2. Let $n = p^\alpha n_1$ for some odd prime p with $p \not| n_1$. Let $r \in \Phi(p^\alpha)$ be a primitive root of modulo p^α.

Then $r^{\frac{n-1}{2}} \neq \pm 1 \bmod n$. Otherwise, $r^{n-1} = 1 \bmod n$, and hence $r^{n-1} = 1 \pmod{p^\alpha}$.

This means that $\phi(p^\alpha)|(n-1)$, so that $p|n$ and $p|(n-1)$ both hold, absurd.

By the Chinese Remainder Theorem, there is an m such that $m \in \Phi(n)$ such that

$$m \equiv r \pmod{p^\alpha}$$

and

$$m \equiv 1 \pmod{n_1}.$$

By the choice of m,

$$(m|n) \neq m^{\frac{n-1}{2}} \pmod{n}.$$

Proof of the characterisation theorem - V

For (3). Let a be such that

$$a \in \Phi(n) \text{ and } (a|n) \neq a^{\frac{n-1}{2}} \pmod{n}. \tag{41}$$

Let B be the set of all $b \in \Phi(n)$ satisfying $(b|n) = b^{\frac{n-1}{2}} \pmod{n}$. Then, for $aB = \{ab \mid b \in B\}$,

(i) $|aB| = |B|$,

(ii) $aB \cap B = \emptyset$,

(iii) For every $x \in aB$,
- $x \in \Phi(n)$, and
- $(x|n) \neq x^{\frac{n-1}{2}} \pmod{n}$.

The Tester $\mathcal{T}$

The tester $\mathcal{T}$ proceeds as follows: For a given natural number n,

1. Randomly and uniformly pick a number m such that $1 < m < n$.
2. If $(m, n) \neq 1$, then n is a composite
3. Otherwise and if $(m|n) \neq m^{\frac{n-1}{2}} \pmod{n}$, then n is composite
4. Otherwise, then with probability at least $\frac{1}{2}$, n is a prime.

Proof of $\mathcal{T}$

- $\mathcal{T}$ runs in time $\log^3 n$
- We may run $\mathcal{T}$ k times, in this case, the probability that
  - n is not a prime, but
  - $\mathcal{T}$ claims n as a prime number

  is

$$\leq \frac{1}{2^k}.$$

By the Prime Number Theorem, for any given n, there are approximately $\frac{n}{\ln n}$ primes within n. Using this, the tester $\mathcal{T}$ is easy to find a large "prime number", that is a true prime with probability ≈ 1, by using relatively large k, $k = \log n$ say.

Advanced reading

1. Quantum machine for factoring
2. Primality is in P

Research directions

- Structures and algorithms for big data
- Error correcting code, coding, and information theory
- Algorithms for factoring and cryptography

谢谢！