15-853:Algorithms in the Real World

ECC I (Overview, Hamming Codes, Linear Codes)
ECC II (Reed-Solomon Codes)
ECC III (LDPC/Expander Codes)
Error Correcting Codes III
– Reed-Solomon Decoding
– Overview of basic Number Theory

Decoding RS codes in polynomial time

REED-SOLOMON DECODING

Reed-Solomon Codes

Irving S. Reed and Gustave Solomon

PDF-417

QR code

Aztec code

DataMatrix code

All 2-dimensional Reed-Solomon bar codes

images: wikipedia

Viewing Messages as Polynomials

A (n, k, n-k+1) code:

Consider the polynomial of degree k-1

$p(x) = a_{k-1} x^{k-1} + \cdots + a_1 x + a_0$

Message: $(a_{k-1}, \ldots, a_1, a_0)$

Codeword: $(p(1), p(2), \ldots, p(n))$

To keep the p(i) fixed size, we use $a_i \in$ finite field of size q^r

To make the i distinct, $n \le q^r$

For simplicity, imagine that $n = q^r$. So we have a $(n, k, n-k+1)_n$ code.

Encoding/Decoding Time

Can choose any n "interpolation points"

E.g., choose n roots of unity

Can then use FFT for encoding, take O(n log n) time.

If there are no errors,

can use FFT to decode the codeword, also O(n log n).

If s errors, not clear what to do.

Naïve Algorithm

Naïve algo: (say s errors)

1. "guess" the n-s uncorrupted locations,
2. find degree-(k-1) poly Q(x) that has

   P(i) = Q(i) for these n-s locations i.

   (if any exist)

Know; if the number of errors $s \le (n-k)/2$

a) we will output the correct polynomial P(x)

b) we will never output any incorrect polynomial.

But "guess" = "enumerate", so time is (n choose s) ~ n^s.

The Berlekamp Welch Algorithm

Say we sent $c_i = P(i)$ for $i = 1..n$

Received c_i' where $c_i = c_i'$ for all but s locations.

Let S be the set of these s error locations.

Suppose we magically know error polynomial $E(x)$

such that $E(x) = 0$ for all x in S.

And $E(x)$ has degree s.

Does such a thing exist?

Sure. $E(x) = \prod_{a \text{ in } S} (x - a)$

The Berlekamp Welch Algorithm

Say we sent $c_i = P(i)$ for $i = 1..n$

Received c_i' where $c_i = c_i'$ for all but s locations.

Let S be the set of these s error locations.

Suppose we magically know error polynomial $E(x)$

such that $E(x) = 0$ for all x in S.

And $E(x)$ has degree s.

Then we know that

$P(i) \cdot E(i) = c_i' \cdot E(i)$ for all $i\ in\ 1..n$

The Berlekamp Welch Algorithm

Know that

$P(i) \cdot E(i) = c_i' \cdot E(i)$ for all $i\ in\ 1..n$

Want to solve for polys $P(x)$ (of deg $k-1$), $E(x)$ of deg s.

How? First, rewrite as:

$\mathrm{R}(i) = c_i' \cdot E(i)$ for all $i\ in\ 1..n$

for polynomials R of degree (k+s-1), E of degree s.

R has k+s "degrees of freedom". E has s+1.

Have n equalities.

So perhaps can get solution if $(k+s)+(s+1) \geq n$.

Return $\frac{R(x)}{E(x)}$.

The current situation

We know that

$R(i) = c_i' \cdot E(i)$ for all $i\ in\ 1..n$

Suppose $\mathrm{R(x)} = \sum_{j=1..k+s-1} r_j\, x^j$

$k+s$ unknowns (the r_i values)

And $E(x) = \sum_{j=0..s} e_j\, x^j$

$s+1$ unknowns (the e_i values)

How to solve for $R(x), E(x)$?

The linear system

Linear equalities

$$r_0 + r_1 \cdot 1 + r_2 \cdot 1^2 + \ ... + r_{k+s-1}1^{k+s-1} = c_1' \cdot (e_0 + e_1 \cdot 1 + \cdots + e_s 1^s)$$

$$r_0 + r_1 \cdot 2 + r_2 \cdot 2^2 + \ ... + r_{k+s-1}2^{k+s-1} = c_1' \cdot (e_0 + e_1 \cdot 2 + \cdots + e_s 2^s)$$

$$\cdots$$

$$r_0 + r_1 \cdot i + r_2 \cdot i^2 + \ ... + r_{k+s-1}i^{k+s-1} = c_1' \cdot (e_0 + e_1 \cdot i + \cdots + e_s i^s)$$

$$\cdots$$

$$r_0 + r_1 \cdot n + r_2 \cdot n^2 + \ ... + r_{k+s-1}n^{k+s-1} = c_1' \cdot (e_0 + e_1 \cdot n + \cdots + e_s n^s)$$

Linearly independent equalities. (Vandermonde matrix.)

Under-constrained: n equations, (k+s)+(s+1) = n+1 variables.

But that's OK, since scaling E, R by same constant also is a solution.

Math for both coding theory and cryptography

A NUMBER THEORY PRIMER

Number Theory Outline

Groups

- Definitions, Examples, Properties
- Multiplicative group modulo n
- The Euler-phi function

Fields

- Definition, Examples
- Polynomials
- Galois Fields

Number theory is crucial for arithmetic over finite sets.

Groups

A **Group** (G,*,I) is a set G with operator * such that:

1. **Closure**. For all $a,b \in G$, $a * b \in G$
2. **Associativity.** For all $a,b,c \in G$, $a*(b*c) = (a*b)*c$
3. **Identity.** There exists $I \in G$, such that for all $a \in G$, $a*I=I*a=a$
4. **Inverse.** For every $a \in G$, there exist a unique element $b \in G$, such that $a*b=b*a=I$

An **Abelian or Commutative Group** is a Group with the additional condition

5. **Commutativity.** For all $a,b \in G$, $a*b=b*a$

Examples of groups

- Integers, Reals or Rationals with Addition
- The nonzero Reals or Rationals with Multiplication
- Non-singular n x n real matrices with Matrix Multiplication
- Permutations over n elements with composition
  $[0\rightarrow1, 1\rightarrow2, 2\rightarrow0] \circ [0\rightarrow1, 1\rightarrow0, 2\rightarrow2] = [0\rightarrow0, 1\rightarrow2, 2\rightarrow1]$

Often we will be concerned with **finite groups**, I.e., ones with a finite number of elements.

Key properties of finite groups

Notation: $a^j \equiv a * a * a * \ldots$ j times

Theorem (Fermat's little): for any finite group (G,*,I) and $g \in G$, $g^{|G|} = I$

Definition: the **order** of $g \in G$ is the smallest positive integer m such that $g^m = I$

Definition: a group G is **cyclic** if there is a $g \in G$ such that order(g) = |G|

Definition: an element $g \in G$ of order |G| is called a **generator** or **primitive element** of G.

Groups based on modular arithmetic

The group of positive integers modulo a prime *p*

$Z_p^* \equiv \{1, 2, 3, \ldots, p-1\}$

$*_p \equiv$ multiplication modulo p

Denoted as: $(Z_p^*, *_p)$

Required properties

1. Closure. Yes.
2. Associativity. Yes.
3. Identity. 1.
4. Inverse. Yes.

Example: $Z_7^* = \{1,2,3,4,5,6\}$

$1^{-1} = 1, 2^{-1} = 4, 3^{-1} = 5, 6^{-1} = 6$

Other properties

$|Z_p^*| = (p-1)$

By Fermat's little theorem: $a^{(p-1)} = 1 \pmod p$

Example of Z_7^*

x	x^2	x^3	x^4	x^5	x^6
1	1	1	1	1	1
2	4	1	2	4	1
3	**2**	**6**	**4**	**5**	**1**
4	2	1	4	2	1
5	**4**	**6**	**2**	**3**	**1**
6	1	6	1	6	1

Generators

For all p the group is cyclic.

What if n is not a prime?

The group of positive integers modulo a non-prime n

$Z_n \equiv \{1, 2, 3, \ldots, n-1\}$, n not prime

$*_p \equiv$ multiplication modulo n

Required properties?

1. Closure. ?
2. Associativity. ?
3. Identity. ?
4. Inverse. ?

How do we fix this?

Groups based on modular arithmetic

The **multiplicative group modulo n**

$Z_n^* \equiv \{m : 1 \le m < n, \gcd(n,m) = 1\}$

$* \equiv$ multiplication modulo n

Denoted as $(Z_n^*, *_n)$

Required properties:

- Closure. Yes.
- Associativity. Yes.
- Identity. 1.
- Inverse. Yes.

Example: $Z_{15}^* = \{1,2,4,7,8,11,13,14\}$

$1^{-1} = 1,\ 2^{-1} = 8,\ 4^{-1} = 4,\ 7^{-1} = 13,\ 11^{-1} = 11,\ 14^{-1} = 14$

The Euler Phi Function

$$\phi(n) = |Z_n^*| = n \prod_{p|n} (1 - 1/p)$$

If n is a product of two primes p and q, then

$$\phi(n) = pq(1-1/p)(1-1/q) = (p-1)(q-1)$$

Note that by Fermat's Little Theorem:

$$a^{\phi(n)} = 1 \pmod{n} \quad \text{for} \quad a \in Z_n^*$$

Or for n = pq

$$a^{(p-1)(q-1)} = 1 \pmod{n} \quad \text{for} \quad a \in Z_{pq}^*$$

This will be very important in RSA!

Generators

Example of Z_{10}^*: {1, 3, 7, 9}

Generators

x	x^2	x^3	x^4
1	1	1	1
3	**9**	**7**	**1**
7	**9**	**3**	**1**
9	1	9	1

For n = (2, 4, p^s, $2p^s$), p an odd prime, Z_n^* is cyclic

Operations we will need

Multiplication: a*b (mod n)

- Can be done in $O(\log^2 n)$ bit operations, or better

Power: a^k (mod n)

- The power method O(log n) steps, $O(\log^3 n)$ bit ops

```
fun pow(a,k) =
  if (k = 0) then 1
  else if (k mod 2 = 1)
       then a * (pow(a,k/2))²
       else (pow(a, k/2))²
```

Inverse: a^{-1} (mod n)

- Extended Euclid's algorithm
  - O(log n) steps, $O(\log^3 n)$ bit ops

Euclid's Algorithm

Euclid's Algorithm:

gcd(a,b) = gcd(b,a mod b)

gcd(a,0) = a

"Extended" Euclid's algorithm:

- Find **x** and **y** such that **ax + by = gcd(a,b)**
- Can be calculated as a side-effect of Euclid's algorithm.
- Note that **x** and **y** can be zero or negative.

This allows us to find **<u>a^{-1} mod n</u>**, for **a** $\in Z_n^*$

In particular return **<u>x</u>** in **<u>ax + ny = 1</u>**.

Euclid's Algorithm

```
fun euclid(a,b) =
  if (b = 0) then a
  else euclid(b, a mod b)
                                   gcd      y
fun ext_euclid(a,b) =
  if (b = 0) then (a, 1, 0)
  else                           x
    let (d, x, y) = ext_euclid(b, a mod b)
    in (d, y, x - (a/b) y)
    end
```

The code is in the form of an inductive proof.

<u>Exercise</u>: prove the inductive step

Discrete Logarithms

If g is a generator of Z_n^*, then for all y there is a unique x (mod ϕ(n)) such that

$$y = g^x \bmod n$$

This is called the **<u>discrete logarithm</u>** of y and we use the notation

$$x = \log_g(y)$$

In general finding the discrete logarithm is conjectured to be hard…as hard as factoring.

Fields

A **<u>Field</u>** is a set of elements F with binary operators * and + such that

1. (F, +) is an **<u>abelian group</u>**
2. (F \ I_+, *) is an **<u>abelian group</u>**
   the "multiplicative group"
3. **<u>Distribution</u>**: a*(b+c) = a*b + a*c
4. **<u>Cancellation</u>**: a*I_+ = I_+

The **order** of a field is the number of elements.

A field of finite order is a **finite field**.

The reals and rationals with + and * are fields.

Finite Fields

$\mathbb{Z}_p$ (p prime) with + and * mod p, is a **finite** field.

1. $(\mathbb{Z}_p, +)$ is an **abelian group** (0 is identity)
2. $(\mathbb{Z}_p \setminus 0, \times)$ is an **abelian group** (1 is identity)
3. **Distribution**: a*(b+c) = a*b + a*c
4. **Cancellation**: a*0 = 0

We denote this by $\mathbb{F}_p$ or GF(p)

Are there other finite fields?

What about ones that fit nicely into bits, bytes and words (i.e with 2^k elements)?

Polynomials over $\mathbb{F}_p$

$\mathbb{F}_p[x]$ = polynomials on x with coefficients in $\mathbb{F}_p$.

- Example of $\mathbb{F}_p[x]$: $f(x) = 3x^4 + 1x^3 + 4x^2 + 3$
- $\deg(f(x)) = 4$ (the **degree** of the polynomial)

Operations: (examples over $\mathbb{F}_5[x]$)

- Addition: $(x^3 + 4x^2 + 3) + (3x^2 + 1) = (x^3 + 2x^2 + 4)$
- Multiplication: $(x^3 + 3) * (3x^2 + 1) = 3x^5 + x^3 + 4x^2 + 3$
- $1_+ = 0$, $1_* = 1$
- + and * are associative and commutative
- Multiplication distributes and 0 cancels

Do these polynomials form a field?

Division and Modulus

Long division on polynomials ($\mathbb{F}_5[x]$):

$$
\begin{array}{r|l}
 & \boxed{1x+4} \\
x^2+1 & x^3+4x^2+0x+3 \\
 & \underline{x^3+0x^2+1x+0} \\
 & 4x^2+4x+3 \\
 & \underline{4x^2+0x+4} \\
 & \boxed{4x+4}
\end{array}
$$

$$(x^3+4x^2+3)/(x^2+1) = (x+4)$$

$$(x^3+4x^2+3) \bmod (x^2+1) = (4x+4)$$

$$(x^2+1)(x+4) + (4x+4) = (x^3+4x^2+3)$$

Polynomials modulo Polynomials

How about making a field of polynomials modulo another polynomial? This is analogous to $\mathbb{F}_p$ (i.e., integers modulo another integer).

e.g. $\mathbb{F}_5[x] \bmod (x^2 + 2x + 1)$

Does this work? E.g., does $(x + 1)$ have an inverse?

Definition: An **irreducible polynomial** is one that is not a product of two other polynomials both of degree greater than 0.

e.g. $(x^2 + 2)$ for $\mathbb{F}_5[x]$

Analogous to a prime number.

Galois Fields

The polynomials

$\mathbb{F}_p[x] \bmod p(x)$

where $p(x) \in \mathbb{F}_p[x]$, p(x) is irreducible,

and deg(p(x)) = n (i.e. n+1 coefficients)

form a finite field. Such a field has p^n elements.

These fields are called **Galois Fields** or **GF(p^n)** or $\mathbb{F}_{p^n}$

The special case n = 1 reduces to the fields $\mathbb{F}_p$.

The special case p = 2 is especially useful for us.

GF(2^n)

$\mathbb{F}_{2^n}$ = set of polynomials in $\mathbb{F}_2[x]$ modulo

irreducible polynomial $p(x) \in \mathbb{F}_2[x]$ of degree n.

Elements are all polynomials in $\mathbb{F}_2[x]$ of degree $\leq n-1$.

Has 2^n elements.

Natural correspondence with bits in $\{0,1\}^n$.

E.g., $x^6 + x^4 + x + 1$ = 01010011

Elements of $\mathbb{F}_{2^8}$ can be represented as **a byte**, one bit for each term.

GF(2^n)

$\mathbb{F}_{2^n}$ = set of polynomials in $\mathbb{F}_2[x]$ modulo

irreducible polynomial $p(x) \in \mathbb{F}_2[x]$ of degree n.

Elements are all polynomials in $\mathbb{F}_2[x]$ of degree $\leq n-1$.

Has 2^n elements.

Natural correspondence with bits in $\{0,1\}^n$.

Addition over $\mathbb{F}_2$ corresponds to xor.

- Just take the xor of the bit-strings (bytes or words in practice). This is dirt cheap

Multiplication over GF(2^n)

If n is small enough can use a table of all combinations.

The size will be 2^n x 2^n (e.g. 64K for $\mathbb{F}_{2^8}$)

Otherwise, use standard shift and add (xor)

Note: dividing through by the irreducible polynomial on an overflow by 1 term is simply a test and an xor.

e.g. 0111 / 1001 = 0111

1011 / 1001 = 1011 xor 1001 = 0010

^ just look at this bit for $\mathbb{F}_{2^3}$

Multiplication over $GF(2^n)$

```
typedef unsigned char uc;

uc mult(uc a, uc b) {
  int p = a;
  uc r = 0;
  while(b) {
    if (b & 1) r = r ^ p;
    b = b >> 1;
    p = p << 1;
    if (p & 0x100) p = p ^ 0x11B;
  }
  return r;
}
```


Finding inverses over $GF(2^n)$

Again, if n is small just store in a table.
- Table size is just 2^n.

For larger n, use Euclid's algorithm.
- This is again easy to do with shift and xors.

Polynomials with coefficients in $GF(p^n)$

Recall that $\mathbb{F}_{p^n}$ was defined in terms of coefficients that were themselves fields (*i.e.*, $\mathbb{F}_p$).

We can apply this **recursively** and define:

$\mathbb{F}_{p^n}[x]$ = polynomials on x with coefficients in $\mathbb{F}_{p^n}$.
- Example of $\mathbb{F}_{2^3}[x]$:
  - $f(x) = 001x^2 + 101x + 010$
    Where 101 is shorthand for x^2+1.

Polynomials with coefficients in $GF(p^n)$

We can make a finite field by using an irreducible polynomial M(x) selected from $\mathbb{F}_{p^n}[x]$.

For an order m polynomial and by abuse of notation we write: **$GF(GF(p^n)^m)$**, which has p^{nm} elements.

Note: all finite fields are isomorphic to $GF(p^n)$ for some p,n so $GF(GF(2^8)^4)$ is just another representation of $GF(2^{32})$.

This representation, however, has practical advantages.

The operations are more modular, easier to implement.