Algorithms in the Real World

Error Correcting Codes II

- Cyclic Codes
- Reed-Solomon Codes

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: $(p(y_0), p(y_1), \ldots, p(y_{n-1}))$ for distinct $y_0, \ldots, y_{n-1}$

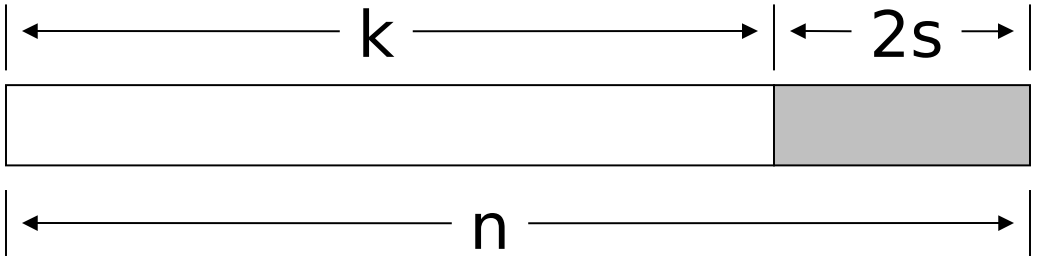
To keep the $p(y_i)$ fixed size, we use $y_i, a_i \in GF(p^r)$

To make the y_i distinct, $n < p^r$

Unisolvence Theorem: Any subset of size k of $(p(y_0), p(y_2), \ldots, p(y_{n-1}))$ is enough to (uniquely) reconstruct p(x) using polynomial interpolation, e.g., LaGrange's Formula.

Polynomial-Based Code

A (n, k, 2s +1) code:



Can **detect** 2s errors

Can **correct** s errors

Can **correct** 2s erasures

Generally can correct α erasures and β errors if
$\alpha + 2\beta \leq 2s$

2s = n-k, so this is an (n,k,n-k+1) code

Correcting Errors

Correcting s errors:

1. Find k + s symbols that agree on a polynomial p(x).
   These must exist since originally k + 2s symbols agreed and only s are in error
2. There are no k + s symbols that agree on the wrong polynomial p’(x)
   - Any subset of k symbols will define p’(x)
   - Since at most s out of the k+s symbols are in error, p’(x) = p(x)

A Systematic Code

Message: $(m_0, m_1, \ldots, m_{k-1})$

Find polynomial $p(x) = a_{k-1} x^{k-1} + \cdots + a_1 x + a_0$ such that $p(y_0) = m_0$, $p(y_2) = m_2$, …, $p(y_{k-1}) = m_0$

Codeword: $(m_0, m_1, \ldots, m_{k-1}, p(y_k), p(y_{k+1}), \ldots, p(y_{n-1}))$

This has the advantage that if we know there are no errors (e.g., all points lie on the same degree-(k-1) polynomial), decoding is trivial.

The version of RS used in practice uses something slightly different.

This will allow us to use the "**Parity Check**" ideas from linear codes (i.e., $Hc^T = 0$?) to quickly test for errors.

Reed-Solomon Codes in the Real World

(204,188,17)$_{256}$: ITU J.83(A)[2]

(128,122,7)$_{256}$: ITU J.83(B)

(255,223,33)$_{256}$: Common in Practice

- Note that they are all byte based (i.e., symbols are from GF(2^8)).

Decoding rate on 1.8GHz Pentium 4:

- (255,251,5) = 89Mbps
- (255,223,33) = 18Mbps

Dozens of companies sell hardware cores that operate 10x faster (or more)

- (204,188,17) = 320Mbps (Altera decoder)

Applications of Reed-Solomon Codes

- **Storage**: CDs, DVDs, “hard drives”,
- **Wireless**: Cell phones, wireless links
- **Sateline and Space**: TV, Mars rover, ...
- **Digital Television**: DVD, MPEG2 layover
- **High Speed Modems**: ADSL, DSL, ..

Good at handling burst errors.

Other codes are better for random errors.

  – e.g., Gallager codes, Turbo codes

RS and “burst” errors

Let’s compare to Hamming Codes (which are “optimal”).

	code bits	check bits
RS $\mathbf{(255, 253, 3)}_{256}$	2040	16
Hamming $\mathbf{(2^{11}\text{-}1, 2^{11}\text{-}11\text{-}1, 3)}_{2}$	2047	11

They can both correct 1 error, but not 2 random errors.

- The Hamming code does this with fewer check bits

However, RS can fix 8 contiguous bit errors in one byte

- Much better than lower bound for 8 arbitrary errors

$$\log\left(1+\binom{n}{1}+\cdots+\binom{n}{8}\right) > 8\log(n-7) \approx 88 \text{ check bits}$$

Discrete Fourier Transform (DFT)

Evaluating polynomial at n points via matrix multiply:

α is a primitive n^{th} root of unity ($\alpha^n = 1$) – a generator

$$T = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{n-1} & \alpha^{2(n-1)} & \cdots & \alpha^{(n-1)(n-1)} \end{pmatrix}$$

$$\begin{pmatrix} c_0 \\ \vdots \\ c_{k-1} \\ c_k \\ \vdots \\ c_{n-1} \end{pmatrix} = T \cdot \begin{pmatrix} m_0 \\ \vdots \\ m_{k-1} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Evaluate polynomial $m_{k-1}x^{k-1} + \cdots + m_1x + m_0$ at n distinct roots of unity, 1, α, α^2, α^3, $\cdots$, α^{n-1}

DFT: $c = Tm$

Inverse DFT: $m = T^{-1}c$

FFT is fast way to compute DFT

Galois Field

GF(2^3) with irreducible polynomial: $x^3 + x + 1$

$\alpha = x$ is a generator

0	0	000	y_0
α	x	010	y_1
α^2	x^2	100	y_2
α^3	x + 1	011	y_3
α^4	$x^2 + x$	110	y_4
α^5	$x^2 + x + 1$	111	y_5
α^6	$x^2 + 1$	101	y_6
α^7	1	001	y_7

Will use this as an example.

DFT Example

$\alpha = x$ is 7th root of unity in $GF(2^3)/x^3 + x + 1$

(i.e., multiplicative group, which excludes additive inverse)

Recall $\alpha = y_1$, $\alpha^2 = y_2$, …

$$T = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & & & \\ 1 & \alpha^3 & \alpha^6 & & & & \\ 1 & \alpha^4 & & \ddots & & & \\ 1 & \alpha^5 & & & & & \\ 1 & \alpha^6 & & & & & \end{pmatrix} = \begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{y_1} & \mathbf{y_1^2} & \mathbf{y_1^3} & \mathbf{y_1^4} & \mathbf{y_1^5} & \mathbf{y_1^6} \\ \mathbf{1} & \mathbf{y_2} & \mathbf{y_2^2} & \mathbf{y_2^3} & & & \\ \mathbf{1} & \mathbf{y_3} & \mathbf{y_3^2} & & & & \\ \mathbf{1} & \mathbf{y_4} & & \ddots & & & \\ \mathbf{1} & \mathbf{y_6} & & & & & \\ \mathbf{1} & \mathbf{y_7} & & & & & \mathbf{y_7^6} \end{pmatrix}$$

Should be clear that $c = T \bullet (m_0, m_1, \ldots, m_{k-1}, 0, \ldots)^T$ is the same as evaluating $p(x) = m_0 + m_1 x + \ldots + m_{k-1} x^{k-1}$ at n points $y_0, y_1, \ldots, y_{n-1}$.

Decoding

Why is it hard?

Brute Force: try $\binom{k+2s}{k+s}$ possibilities and solve for each.

Cyclic Codes

A linear code is cyclic if:

$$(c_0, c_1, \ldots, c_{n-1}) \in C \Rightarrow (c_{n-1}, c_0, \ldots, c_{n-2}) \in C$$

Both **Hamming** and **Reed-Solomon** codes are cyclic.

Note: we might have to reorder the columns to make the code “cyclic”.

Motivation: Cyclic codes are easier to decode than general codes.

Linear Code Generator and Parity Check Matrices

View message, codeword as vectors $(m_0, m_1, \ldots, m_{k-1})$ and $(c_0, c_1, \ldots, c_{n-1})$

Generator Matrix:

A k x n matrix **G** such that:

$$C = \{m \bullet \mathbf{G} \mid m \in \sum^k\}$$

Made from stacking the basis vectors

Parity Check Matrix:

A (n – k) x n matrix **H** such that:

$$C = \{v \in \sum^n \mid H \bullet v^T = 0\}$$

Codewords are the nullspace of H

These **always exist for linear codes**

$H \bullet G^T = 0$

RS Generator and Parity Check Polynomials

View message $(m_0, m_1, \ldots, m_{k-1})$ as polynomial $m_0 + m_1x + \ldots + m_{k-1}x^{k-1}$,

codeword $(c_0, c_1, \ldots, c_{n-1})$ as polynomial $c_0 + c_1x + \ldots + c_{n-1}x^{n-1}$

Generator Polynomial:

A degree (n-k) polynomial $g(x) = g_0 + g_1x + \ldots + g_{n-k}x^{n-k}$

such that $\mathbf{g} \mid x^n - 1$

$$C = \{m \bullet \mathbf{g} \mid m \in m_0 + m_1x + \ldots + m_{k-1}x^{k-1}\}$$

Parity Check Polynomial:

A degree k polynomial $h(x) = h_0 + h_1x + \ldots + h_kx^k$

such that $\mathbf{h} \mid x^n - 1$

$$C = \{v \in \sum^n[x] \mid h \bullet v = 0 \pmod{x^n - 1}\}$$

These **always exist for linear cyclic codes**

$h \bullet g = x^n - 1$

Poly multiplication via matrix multiplication

If $g(x) = g_0 + g_1x + \ldots + g_{n-k}x^{n-k}$

We can put this generator in matrix form (k x n):

$$G = \begin{pmatrix} \overbrace{g_0 & g_1 & \cdots & g_{k-1} & \cdots & g_{n-k}}^{\text{n-k+1}} & \overbrace{0 & 0 & 0}^{\text{k-1}} \\ 0 & g_0 & \cdots & \cdots & \cdots & g_{n-k-1} & g_{n-k} & & 0 \\ \vdots & & \ddots & & \cdots & & & \ddots & \vdots \\ \underbrace{0 & 0 & \cdots}_{\text{k-1}} & g_0 & \cdots & g_{n-2k-1} & \cdots & \cdots & g_{n-k} \end{pmatrix}$$

Write $m = m_0 + m_1x + \ldots + m_{k-1}x^{k-1}$ as $(m_0, m_1, \ldots, m_{k-1})$

Then c = mG

g generates cyclic codes

$$G = \begin{pmatrix} g_0 & g_1 & \cdots & g_{k-1} & \cdots & g_{n-k} & 0 & 0 & 0 \\ 0 & g_0 & \cdots & \cdots & \cdots & g_{n-k} & g_{n-k} & & 0 \\ \vdots & & \ddots & & \cdots & & & \ddots & \vdots \\ 0 & 0 & \cdots & g_0 & \cdots & g_{n-2k-1} & \cdots & \cdots & g_{n-k} \end{pmatrix} = \begin{pmatrix} g \\ xg \\ \vdots \\ x^{k-1}g \end{pmatrix}$$

Codes consist of all linear combinations of the rows:

$c = (c_0,c_1,\ldots,c_{n-1}) = m_0g + m_1xg + m_2x^2g + \ldots + m_{k-1}x^{k-1}g$

Claim: $c' = (c_{n-1},c_0,c_1,\ldots,c_{n-2})$ is also a codeword.

Right shift of every row but last is next row.

Right shift of last row is $x^kg \bmod (x^n - 1) = g_{n-k},0,\ldots,g_0,g_1,\ldots,g_{n-k-1}$

Will show $x^kg \bmod (x^n - 1)$ is a linear combination of other rows.

Consider $h = h_0 + h_1x + \ldots + h_kx^k$ $(gh = x^n - 1)$

$h_0g + (h_1x)g + \ldots + (h_{k-1}x^{k-1})g + (h_kx^k)g = x^n - 1$

$(h_kx^k)g = (x^n - 1) - (h_0g + (h_1x)g + \ldots + (h_{k-1}x^{k-1})g)$

$(h_kx^k)g \bmod (x^n - 1) = -(h_0g + h_1(xg) + \ldots + h_{k-1}(x^{k-1}g))$

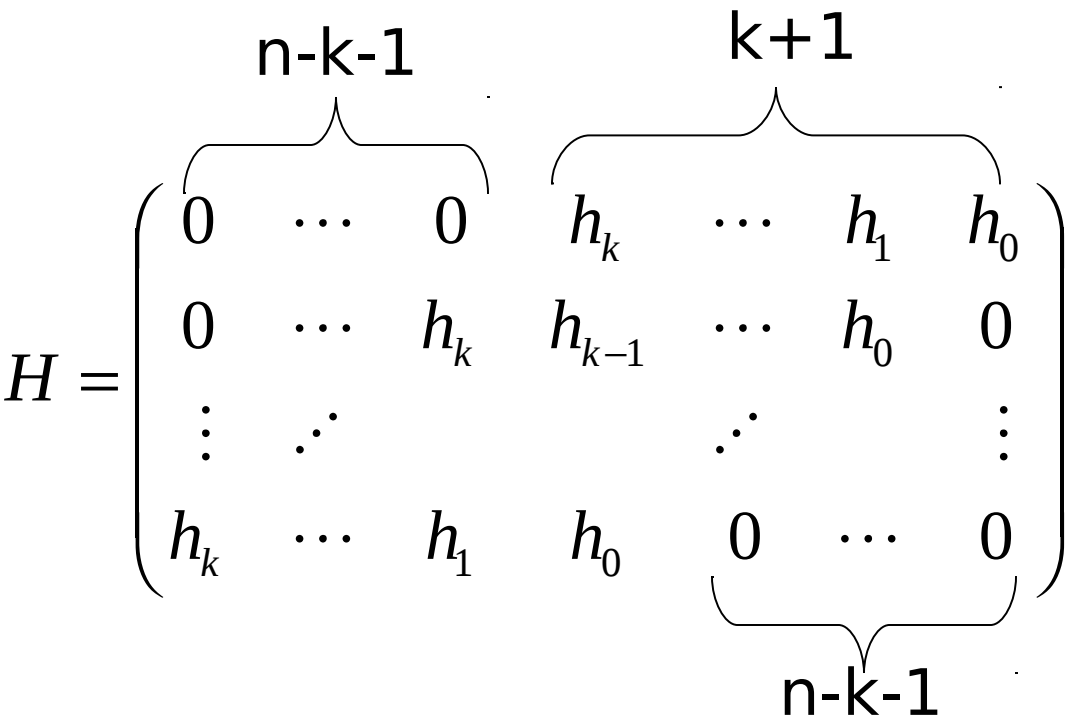
$x^kg \bmod (x^n - 1) = -h_k^{-1}(h_0g + h_1(xg) + \ldots + h_{k-1}(x^{k-1}g))$

Therefore right cyclic shift of every row is a linear combination of other rows.

Viewing h as a matrix

If $h = h_0 + h_1x + \ldots + h_kx^k$

we can put this parity check poly. in matrix form ((n-k) x n) :



$Hc^T = 0$ (syndrome gives coefficients of x^{n-1} through x^k in $h \bullet c$, which are the same as in $h \bullet c \bmod x^n-1$)

Hamming Codes Revisited

The Hamming $(7,4,3)_2$ code.

$g = 1 + x + x^3$ $\qquad$ $h = x^4 + x^2 + x + 1$

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \qquad H = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

$gh = x^7 - 1, \quad GH^T = 0$

The columns are not identical to the previous example Hamming code. (And note that coefficients are from GF(2), but n > 2.)

Factors of $x^n - 1$

Intentionally left blank

Another way to write g

Let α be a **generator** of GF(p^r).

Let $n = p^r - 1$ (the size of the multiplicative group)

Then we can write a generator polynomial as

$g(x) = (x-\alpha)(x-\alpha^2) \ldots (x - \alpha^{n-k})$, $h = (x- \alpha^{n-k+1})\ldots(x-\alpha^n)$

Lemma: $g \mid x^n - 1$, $h \mid x^n - 1$, $gh = x^n - 1$

($a \mid b$ means a divides b)

Proof:

- $\alpha^n = 1$ (because of the size of the group)
  $\Rightarrow \alpha^n - 1 = 0$
  $\Rightarrow \alpha$ root of $x^n - 1$
  $\Rightarrow (x - \alpha) \mid x^n - 1$
- similarly for $\alpha^2, \alpha^3, \ldots, \alpha^n$
- therefore $x^n - 1$ is divisible by $(x - \alpha)(x - \alpha^2) \ldots$

Back to Reed-Solomon

Consider a generator polynomial $g \in GF(p^r)[x]$, s.t. $g \mid (x^n - 1)$

Recall that $n - k = 2s$ (the degree of g is n-k, n-k+1 coefficients)

Encode (trick to make code systematic):

- $m' = m\, x^{2s}$ (basically shift by 2s)
- $b = m' \pmod g$, $m' = qg + b$ for some q
- $c = m' - b = (m_{k-1}, \ldots, m_0, -b_{2s-1}, \ldots, -b_0)$
- Note that c is a **cyclic code** based on g
  - $c = m' - b = qg$

  (i.e., given m we found another message q such that qg is "systematic" for m)

Parity check:

- $h\, c = 0$?

Example

Lets consider the $(7,3,5)_8$ Reed-Solomon code.

We use $GF(2^3)/x^3 + x + 1$

α	x	010
α^2	x^2	100
α^3	$x + 1$	011
α^4	$x^2 + x$	110
α^5	$x^2 + x + 1$	111
α^6	$x^2 + 1$	101
α^7	1	001

Example RS $(7,3,5)_8$

$n = 7, k = 3, n-k = 2s = 4, d = 2s+1 = 5$

$g = (x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4)$

$= x^4 + \alpha^3 x^3 + x^2 + \alpha x + \alpha^3$

$h = (x - \alpha^5)(x - \alpha^6)(x - \alpha^7)$

$= x^3 + \alpha^3 x^2 + \alpha^2 x + \alpha^4$

$gh = x^7 - 1$

Consider the message: 110 000 110

$m = (\alpha^4, 0, \alpha^4) = \alpha^4 x^2 + \alpha^4$

$m' = x^4 m = \alpha^4 x^6 + \alpha^4 x^4$

$= (\alpha^4 x^2 + x + \alpha^3)g + (\alpha^3 x^3 + \alpha^6 x + \alpha^6)$

$c = (\alpha^4, 0, \alpha^4, \alpha^3, 0, \alpha^6, \alpha^6)$

$= 110\ 000\ 110\ 011\ 000\ 101\ 101$

α	010
α^2	100
α^3	011
α^4	110
α^5	111
α^6	101
α^7	001

$ch = 0 \pmod{x^7 - 1}$

A useful theorem

Theorem: For any β, if $g(\beta) = 0$ then $\beta^{2s}m(\beta) = b(\beta)$

Proof:

$x^{2s}m(x) = m'(x) = g(x)q(x) + b(x)$

$\beta^{2s}m(\beta) = g(\beta)q(\beta) + b(\beta) = b(\beta)$

Corollary: $\beta^{2s}m(\beta) = b(\beta)$ for $\beta \in \{\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{2s=n-k}\}$

Proof:

$\{\alpha, \alpha^2, \ldots, \alpha^{2s}\}$ are the roots of g by definition.

Fixing erasures

Theorem: Any k symbols from c can reconstruct c and hence m

Proof:

We can write 2s equations involving m (c_{n-1}, …, c_{2s}) and b (c_{2s-1}, …, c_0). These are

$\alpha^{2s}\, m(\alpha) = b(\alpha)$

$\alpha^{4s}\, m(\alpha^2) = b(\alpha^2)$

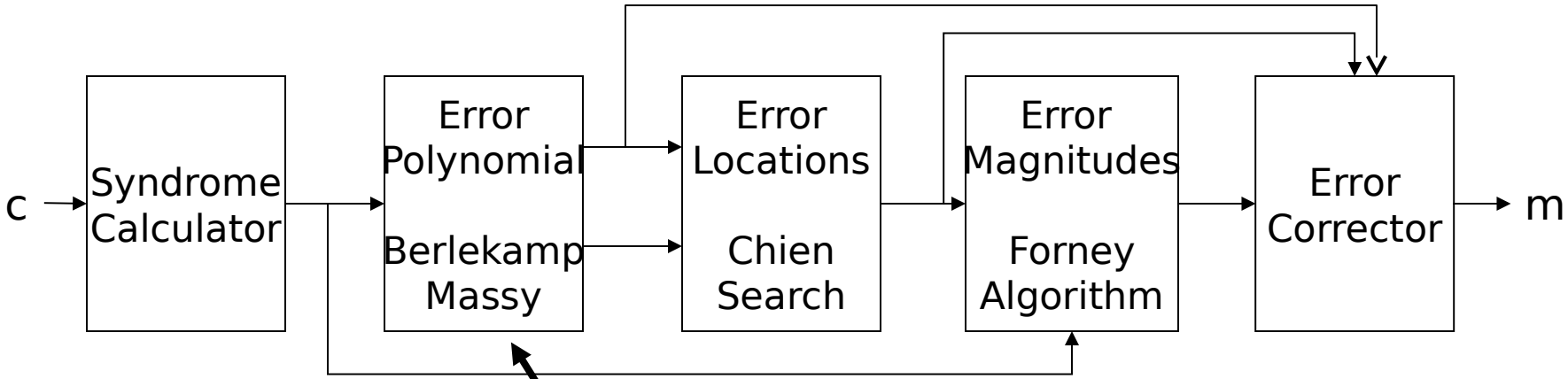
…

$\alpha^{2s(2s)}\, m(\alpha^{2s}) = b(\alpha^{2s})$

We have at most 2s unknowns (erasures), so we can solve for them. (I'm skipping showing that the equations are linearly independent).

Efficient Decoding

I don’t plan to go into the Reed-Solomon decoding algorithm, other than to mention the steps.



This is the hard part. CD players use this algorithm. (Can also use Euclid’s algorithm.)