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 \(y_i, a_i \in \text{GF}(p^r)\)

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:

A code can:
- Detect \(2s\) errors
- Correct \(s\) errors
- Correct \(2s\) erasures

Generally can correct \(\alpha\) erasures and \(\beta\) 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**: \((m_0, m_1, ..., m_{k-1})\)

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**: \((m_0, m_1, ..., m_{k-1}, p(y_k), p(y_{k+1}), ..., p(y_{n-1}))\)

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

\((204,188,17)_{256} : \text{ITU J.83(A)}\)

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

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

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

Decoding rate on 1.8GHz Pentium 4:
- \((255,251,5) = 89\text{Mbps}\)
- \((255,223,33) = 18\text{Mbps}\)

Dozens of companies sell hardware cores that operate 10x faster (or more)
- \((204,188,17) = 320\text{Mbps} \text{ (Altera decoder)}\)
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”).

<table>
<thead>
<tr>
<th>Code</th>
<th>Code Bits</th>
<th>Check Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS ((255, 253, 3)_{256})</td>
<td>2040</td>
<td>16</td>
</tr>
<tr>
<td>Hamming ((2^{11}-1, 2^{11}-11-1, 3)_{2})</td>
<td>2047</td>
<td>11</td>
</tr>
</tbody>
</table>

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:

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

Evaluating polynomial at n points via matrix multiply: 
\( \alpha \) is a primitive n\(^{th} \) root of unity \( (\alpha^n = 1) \) – a generator

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

\[
\begin{pmatrix}
c_0 \\
c_1 \\
c_k \\
c_k \\
c_{n-1}
\end{pmatrix} = T \begin{pmatrix}
m_0 \\
m_1 \\
m_1 \\
m_k \\
m_{k-1}
\end{pmatrix}
\]

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

DFT: \( c = Tm \)

Inverse DFT: \( m = T^{-1}c \)
Galois Field

GF(2^3) with irreducible polynomial: \( x^3 + x + 1 \)
\( \alpha = x \) is a generator

<table>
<thead>
<tr>
<th>( \alpha^0 = 0 )</th>
<th>( \alpha^1 = x )</th>
<th>( \alpha^2 = x^2 )</th>
<th>( \alpha^3 = x + 1 )</th>
<th>( \alpha^4 = x^2 + x )</th>
<th>( \alpha^5 = x^2 + x + 1 )</th>
<th>( \alpha^6 = x^2 + 1 )</th>
<th>( \alpha^7 = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>000</td>
<td>( y_0 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \alpha^1 = x )</td>
<td></td>
<td>010</td>
<td>( y_1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \alpha^2 = x^2 )</td>
<td></td>
<td>100</td>
<td>( y_2 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \alpha^3 = x + 1 )</td>
<td></td>
<td>011</td>
<td>( y_3 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \alpha^4 = x^2 + x )</td>
<td></td>
<td>110</td>
<td>( y_4 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \alpha^5 = x^2 + x + 1 )</td>
<td></td>
<td>111</td>
<td>( y_5 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \alpha^6 = x^2 + 1 )</td>
<td></td>
<td>101</td>
<td>( y_6 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \alpha^7 = 1 )</td>
<td></td>
<td>001</td>
<td>( y_7 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Will use this as an example.
**DFT Example**

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

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

Recall \( \alpha = y_1, \alpha^2 = y_2, \ldots \)

\[
T = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\
1 & \alpha^2 & \alpha^4 & \alpha^6 \\
1 & \alpha^3 & \alpha^6 \\
1 & \alpha^4 & \cdot & \cdot & \cdot \\
1 & \alpha^5 \\
1 & \alpha^6 
\end{bmatrix}
\]

\[
= \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & y_1 & y_1^2 & y_1^3 & y_1^4 & y_1^5 & y_1^6 \\
1 & y_2 & y_2^2 & y_2^3 \\
1 & y_3 & y_3^2 \\
1 & y_4 & \cdot & \cdot \\
1 & y_6 \\
1 & y_7 & y_7^6 
\end{bmatrix}
\]

Should be clear that \( c = T \cdot (m_0, m_1, \ldots, m_{k-1}, 0, \ldots)^T \)

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 \((m_0, m_1, ..., m_{k-1})\) and \((c_0, c_1, ..., c_{n-1})\)

**Generator Matrix:**
A \(k \times n\) matrix \(G\) such that:
\[ C = \{m \cdot G \mid m \in \Sigma^k\} \]
Made from stacking the basis vectors

**Parity Check Matrix:**
A \((n - k) \times n\) matrix \(H\) such that:
\[ C = \{v \in \Sigma^n \mid H \cdot v^T = 0\} \]
Codewords are the nullspace of \(H\)

These **always exist for linear codes**
\[ H \cdot G^T = 0 \]
RS Generator and Parity Check Polynomials

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

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

**Generator Polynomial:**
A degree \((n-k)\) polynomial \(g(x) = g_0 + g_1x + ... + g_{n-k}x^{n-k}\)
such that \(g \mid x^n - 1\)
\[
C = \{m \cdot g \mid m \in m_0 + m_1x + ... + m_{k-1}x^{k-1}\}
\]

**Parity Check Polynomial:**
A degree \(k\) polynomial \(h(x) = h_0 + h_1x + ... + h_kx^k\)
such that \(h \mid x^n - 1\)
\[
C = \{v \in \Sigma^n[x] \mid h \cdot v = 0 \pmod{x^n - 1}\}
\]

These **always exist for linear cyclic codes**

\(h \cdot g = x^n - 1\)
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 \((k \times n)\):

\[
G = \begin{pmatrix}
g_0 & g_1 & \cdots & g_{k-1} & \cdots & g_{n-k} & 0 & 0 & 0 \\
0 & g_0 & \cdots & \cdots & \cdots & g_{n-k-1} & g_{n-k} & 0 \\
\vdots & \vdots & \ddots & \cdots & \cdots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & g_0 & \cdots & g_{n-2k-1} & \cdots & \cdots & g_{n-k}
\end{pmatrix}
\]

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

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

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 \(x^kg \mod (x^n - 1) = g_{n-k},0,\ldots,g_0,g_1,\ldots,g_{n-k-1}\)

Will show \(x^kg \mod (x^n - 1)\) is a linear combination of other rows.
Consider \(h = h_0 + h_1x + \ldots + h_kx^k\) \((gh = x^n - 1)\)
\[
h_0g + (h_1x)g + \ldots + (h_{k-1}x^{k-1})g + (h_kx^k)g = x^n - 1
\]
\[
(h_kx^k)g = (x^n - 1) - (h_0g + (h_1x)g + \ldots + (h_{k-1}x^{k-1})g)
\]
\[
(h_kx^k)g \mod (x^n - 1) = -(h_0g + h_1(xg) + \ldots + h_{k-1}(x^{k-1}g))
\]
\[
x^kg \mod (x^n - 1) = -h^{-1}_k(h_0g + h_1(xg) + \ldots + h_{k-1}(x^{k-1}g))
\]
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 $((n-k) \times n)$:

$$H = \begin{pmatrix}
0 & \cdots & 0 & h_k & \cdots & h_1 & h_0 \\
0 & \cdots & h_k & h_{k-1} & \cdots & h_0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
h_k & \cdots & h_1 & h_0 & 0 & \cdots & 0
\end{pmatrix}$$

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

The Hamming \((7,4,3)\) code.

\[
g = 1 + x + x^3 \\
G = \begin{pmatrix}
1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 \\
\end{pmatrix} \\
\]

\[
h = x^4 + x^2 + x + 1 \\
H = \begin{pmatrix}
0 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 \\
\end{pmatrix} \\
\]

\[
gh = x^7 - 1, \quad GH^\top = 0
\]

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

**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) 
  \[ \Rightarrow \alpha^n - 1 = 0 \]
  \[ \Rightarrow \alpha \text{ root of } x^n - 1 \]
  \[ \Rightarrow (x - \alpha) \mid x^n - 1 \]

- 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 \( g \in \text{GF}(p^r)[x] \), 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 \( q \)
- \( c = m' - b = (m_{k-1}, ..., m_0, -b_{2s-1}, ..., -b_0) \)
- 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

Let's consider the \((7,3,5)_8\) Reed-Solomon code. We use \(\text{GF}(2^3)/x^3 + x + 1\)

<table>
<thead>
<tr>
<th>(\alpha)</th>
<th>(x)</th>
<th>010</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\alpha^2)</td>
<td>(x^2)</td>
<td>100</td>
</tr>
<tr>
<td>(\alpha^3)</td>
<td>(x + 1)</td>
<td>011</td>
</tr>
<tr>
<td>(\alpha^4)</td>
<td>(x^2 + x)</td>
<td>110</td>
</tr>
<tr>
<td>(\alpha^5)</td>
<td>(x^2 + x + 1)</td>
<td>111</td>
</tr>
<tr>
<td>(\alpha^6)</td>
<td>(x^2 + 1)</td>
<td>101</td>
</tr>
<tr>
<td>(\alpha^7)</td>
<td>1</td>
<td>001</td>
</tr>
</tbody>
</table>
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^3x^3 + x^2 + \alpha x + \alpha^3

h = (x - \alpha^5)(x - \alpha^6)(x - \alpha^7)
= x^3 + \alpha^3x^2 + \alpha^2x + \alpha^4

gh = x^7 - 1

Consider the message: 110 000 110

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

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

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

= 110 000 110 011 000 101 101

ch = 0 (mod x^7 - 1)
A useful theorem

**Theorem:** For any $\beta$, 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.
**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

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

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

$$\vdots$$

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

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.)