Outline

Recall: For **integers**

Euclidean algorithm for finding gcd's
Extended Euclid for finding multiplicative inverses
Extended Euclid for computing Sun-Ze
Test for primitive roots

Now, some analogues for **polynomials** with coefficients in $\mathbf{F}_2 = \mathbf{Z}/2$

Euclidean algorithm for gcd's
Concept of equality mod $M(x)$
Extended Euclid for inverses mod $M(x)$

Looking for good codes

Hamming bound for arbitrary codes
Idea of **linear** codes
Gilbert-Varshamov bound for linear codes

Divisibility for (binary) polynomials

As for integers, use notation

$$f(x)\,\%\,g(x) = \text{remainder dividing } f(x) \text{ by } g(x)$$

Say that $g(x)$ **divides** $f(x)$ if $f(x)\,\%\,g(x) = 0$, and write

$$g(x)|f(x)$$

Write

$$f(x) = g(x) \bmod m(x)$$

if

$$m(x)|[f(x) - g(x)]$$

As with ordinary integers, this relation of **equality modulo** $m(x)$ is an equivalence relation, and is compatible with addition and multiplication.

The **greatest common divisor** $\gcd(f(x), g(x))$ of two polynomials is the largest-degree polynomial dividing both.

Euclid for (binary) polynomials

The Euclidean algorithm for polynomials with coefficients in a **field** (ok, let's say the field is $\mathbf{F}_2 = \mathbf{Z}/2$) is exactly parallel in structure to the Euclidean algorithm for integers.

Each step in the Euclidean algorithm is a division with remainder (now somewhat harder than with integers), and the dividend for the next step is the divisor of the current step, the next divisor is the current remainder, and a new remainder is computed.

That is, to compute the *gcd* of polynomials $f(x)$ and $g(x)$, initialize $F(x) = f(x)$, $G(x) = g(x)$, $R(x) = f(x) \,\%\, g(x)$.

```
  While $R(x) \neq 0$
    replace $F(x)$ by $G(x)$
    replace $G(x)$ by $R(x)$
    recompute $R(x) = F(x) \,\%\, G(x)$.
  When $R(x) = 0$, $G(x) = \gcd(f(x), g(x))$
```

Example: To find $\gcd(x^4+x^2+1, x^2+1)$, do Euclid with these two polynomials as inputs

$$(x^4+x^2+1)-(x^2)\cdot(x^2+1)=1$$

$$(x^2+1)-(x^2+1)\cdot(1)=0$$

Thus, the gcd of x^4+x^2+1 and x^2+1 is 1.

Example: To find $\gcd(x^5+x^4+x+1, x^2+1)$, do Euclid with these two polynomials as inputs

$$(x^5+x^4+x+1)-(x^3+x^2+x+1)\cdot(x^2+1)=0$$

Thus, the gcd of x^5+x^4+x+1 and x^2+1 is x^2+1.

Example:
To find $\gcd(x^5+x^4+x+1, x^4+x^2+1)$, do Euclid with these two polynomials as inputs

$$(x^5+x^4+x+1)-(x+1)\cdot(x^4+x^2+1)=x^3+x^2$$

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

$$(x^3+x^2)-(x^3+x^2)\cdot(1)=0$$

So the gcd of x^5+x^4+x+1 and x^4+x^2+1 is 1.

Example:
To find $\gcd(x^6+x^5+x^3+x+1, x^4+x^2+1)$, do Euclid with these two polynomials as inputs

$$(x^6+x^5+x^3+x+1)-(x^2+x+1)\cdot(x^4+x^2+1)=0$$

Thus, the gcd of $x^6+x^5+x^3+x+1$ and x^4+x^2+1 is x^4+x^2+1. (Not at all obvious that the second poly divides the first!)

Example: To find $\gcd(x^6+x^5+x^3+x+1, x^4+x^3+x+1)$, do Euclid with these two polynomials as inputs

$$(x^6+x^5+x^3+x+1)-x^2\cdot(x^4+x^3+x+1)=x^2+x+1$$

$$(x^4+x^3+x+1)-(x^2+1)\cdot(x^2+x+1)=0$$

Thus, the gcd of $x^6+x^5+x^3+x+1$ and x^4+x^3+x+1 is x^2+x+1.

Remark: These *gcd*'s are indeed less intuitive than *gcd*'s of integers.

Peculiar characterization of gcd's

Our polynomials could have coefficients in any **field**, such as $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{C}$, $\mathbf{Z}/p$ with p prime, or any other finite field, but we'll focus on *binary* ones, meaning with coefficients in $\mathbf{F}_2 = \mathbf{Z}/2$. Just as with integers, and with the same proof, we have

Theorem: The gcd of $g(x)$ of two polynomials $A(x)$ and $B(x)$ is the polynomial of lowest degree expressible as

$$g(x) = R(x) \cdot A(x) + S(x) \cdot B(x)$$

for some polynomials $R(x)$ and $S(x)$.

///

Multiplicative inverses mod $M(x)$

A polynomial $i(x)$ is a **multiplicative inverse** of $f(x)$ modulo $M(x)$ if

$$[f(x) \cdot i(x)] \,\%\, M(x) = 1$$

or, equivalently, if

$$f(x) \cdot i(x) = 1 \bmod M(x)$$

As a corollary of the peculiar characterization of the *gcd*, for $\gcd(f(x), M(x)) = 1$, there are $r(x)$ and $s(x)$ such that

$$1 = \gcd(f(x), M(x)) = r(x) \cdot f(x) + s(x) \cdot M(x)$$

Considering the equation

$$r(x) \cdot f(x) + s(x) \cdot M(x) = 1$$

modulo $M(x)$, we have

$$r(x) \cdot f(x) = 1 \bmod M(x)$$

so $r(x)$ is a multiplicative inverse of $f(x)$ mod $M(x)$.

(And, symmetrically, $s(x)$ is a multiplicative inverse of $M(x)$ mod $f(x)$.)

As with ordinary integers, use the (extended) Euclidean algorithm to find polynomials $r(x)$ and $s(x)$ such that

$$\gcd(f, g) = r \cdot f + s \cdot g$$

Example: To find a multiplicative inverse of x mod $x^2 + x + 1$, use extended Euclid with inputs these two polynomials:

$$\begin{array}{ccc} x^2 + x + 1 - (x+1)(x) & = & 1 \\ x - (x)(1) & = & 0 \end{array}$$

$$1 \quad = \quad (1)(x^2 + x + 1) - (x+1)(x)$$

Since in general

$$1 = r \cdot f + s \cdot g$$

implies that r is a multiplicative inverse of f mod g we see that $x + 1$ is a multiplicative inverse of x mod $x^2 + x + 1$.

Example: To find a multiplicative inverse of x^2+1 mod x^3+x^2+1, use extended Euclid with inputs these two polynomials:

$$\begin{array}{rcl} x^3+x^2+1-(x+1)(x^2+1) & = & x \\ x^2+1-(x)(x) & = & 1 \\ x-(x)(1) & = & 0 \end{array}$$

$$\begin{array}{rcl} 1 & = & (1)(x^2+1)+(x)(x) \\ & = & (x^2+1)+(x)((x^3+x^2+1)+(x+1)(x^2+1)) \\ & = & (x)(x^3+x^2+1)+(x^2+x+1)(x^2+1) \end{array}$$

Since in general

$$1 = r \cdot f + s \cdot g$$

implies that r is a multiplicative inverse of f mod g, x^2+x+1 is a multiplicative inverse of x^2+1 mod x^3+x^2+1.

Back to codes

Review:

Memoryless, binary, discrete channels
We always do minimum-distance decoding
This is the same as max likelihood decoding

This *includes* ‘error correction’: If a received word is closer to the sent word than to any other codeword, the ‘correction’ is correct.

If by mischance there are so many bit errors that the received word is closer to a *different* codeword than the one sent, then the ‘correction’ is *wrong*.

... *but we have no way of knowing this.*

As a default, we imagine that any pattern of errors with more bit errors than half the minimum distance between codewords will not be correctly corrected.

If the minimum distance is $d = 2e + 1$, then any e bit errors *can* be (correctly) corrected.

Shannon's theorem assures that good codes exist...

Our goal is to find/make good codes.

This has proven surprisingly difficult, by contrast to relative success in *compression* (source) coding.

This entails trying to make a code meet two *conflicting* conditions:

Have a high **information rate**
Have a high **minimum distance**

These two conditions are in conflict because, for a fixed length, the higher the minimum distance the less *room* (intuitively!?) there is for codewords, which pushes the information rate down.

On the other hand, if we push the information rate *up* by *adding* codewords, in general this decreases the minimum distance, so decreases the number of bit errors that can be (easily, nicely) corrected.

How to arrange codewords cleverly?

Hamming bound

Using the physical analogy that Hamming distance is really like *distance*:

the set of all length n codewords with an alphabet with q letters (maybe $q = 2$) is like a *container*

codewords with specified minimum distance $d = 2e + 1$ between them are like *balls of radius* e

and the question of how many codewords of length n (alphabet size q) with minimum distance $d = 2e + 1$ can be chosen is analogous to asking

How many balls of a fixed radius can be packed into a box with a specified volume?

This is a hard question, but an easier version is definitive:

$$\boxed{\text{the total volume of the balls packed cannot be greater than the volume of the container.}}$$

(Duh!)

Here *total volume* is the number of length n words on a q-character alphabet, namely q^n.

The volume of a ball of radius e centered at a word w is the number of length n words that differ from w at $\leq e$ positions:

$$\begin{array}{lcc}
\text{no. differing at 0 positions} & = & 1 \\
\text{no. differing at 1 positions} & = & \binom{n}{1} \cdot (q-1) \\
\text{no. differing at 2 positions} & = & \binom{n}{2} \cdot (q-1)^2 \\
\text{no. differing at 3 positions} & = & \binom{n}{3} \cdot (q-1)^3 \\
 & \cdots & \\
\text{no. differing at } e \text{ positions} & = & \binom{n}{e} \cdot (q-1)^e
\end{array}$$

So the volume is

$$\boxed{\begin{array}{c}\text{volume of ball radius } e \text{ of dimension } n \\ = 1+\binom{n}{1}(q-1)+\binom{n}{2}(q-1)^2+\ldots+\binom{n}{e}(q-1)^e\end{array}}$$

Thus, with ℓ codewords of length n, alphabet with q characters, with minimum distance $d = 2e + 1$, the constraint that *the sum of the volumes of the balls cannot be greater than the volume of the whole container in which they're packed* is

$$q^n \geq \ell \cdot \left[1 + \binom{n}{1}(q-1) + \ldots + \binom{n}{e}(q-1)^e\right]$$

This is the **Hamming bound**.

If a code *exists* with ℓ codewords, of length n, and minimum distance $d = 2e+1$, this inequality must hold.

The *contrapositive* assertion is that, given ℓ, n, q, and $d = 2e + 1$ if the inequality *fails* then there *cannot exist* any such code.

Remark: Even when the equality holds, there is no assurance that a code exists. Failure to meet the Hamming bound can prove non-existence, but *meeting* the bound cannot prove existence.

Example: Is there a binary code of length 5 with minimum distance 3 and 7 codewords?

Since the code is binary, the parameter q telling the size of the alphabet in the Hamming bound is just $q = 2$. The minimum distance $d = 2e + 1$ in the statement of the Hamming bound is 3, so the parameter e would be 1. The length is $n = 5$, and the alleged number of codewords is $\ell = 7$. Thus, the assertion

$$q^n \geq \ell \cdot \left[1 + \binom{n}{1}(q-1) + \ldots + \binom{n}{e}(q-1)^e\right]$$

of the Hamming bound in this case would be

$$2^5 \geq 7 \cdot \left[1 + \binom{5}{1}\right]$$

or

$$32 \geq 42$$

which is *false*. Thus, there is no such code.

Remark: When the Hamming bound is violated we are *asking for far too much* from the alleged code.

Example: Is there a binary code of length 6 with minimum distance 3 and 7 codewords?

Since the code is binary, the parameter q telling the size of the alphabet in the Hamming bound is $q = 2$. The minimum distance $d = 2e + 1$ in the statement of the Hamming bound is 3, so the parameter e would be 1. The length is $n = 6$, and the alleged number of codewords is $\ell = 7$. Thus, the assertion

$$q^n \geq \ell \cdot \left[1 + \binom{n}{1}(q-1) + \ldots + \binom{n}{e}(q-1)^e\right]$$

of the Hamming bound in this case would be

$$2^6 \geq 7 \cdot \left[1 + \binom{6}{1}\right]$$

or

$$64 \geq 49$$

which is *true*.

$\boxed{\textit{We reach } \textbf{no} \textit{ conclusion.}}$

Remark: When the Hamming bound is satisfied we *cannot* say whether there does or does not exist any such code.