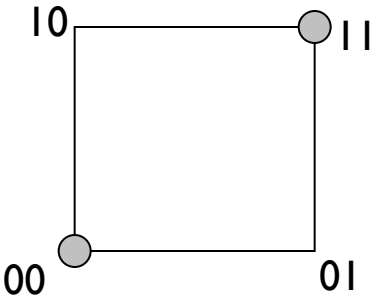
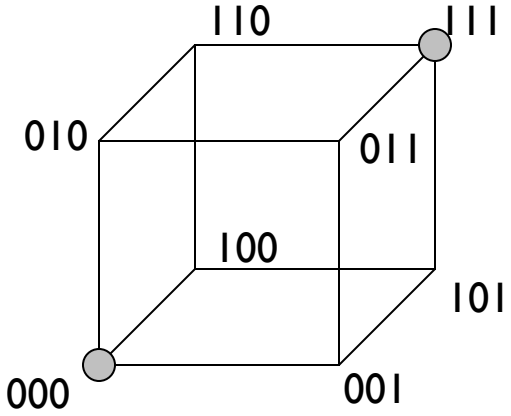
Error Correcting One Bit Messages

How many bits do we need to correct a one bit error on a one bit message?



2 bits
0 -> 00, 1-> 11
(n=2,k=1,d=2)



3 bits
0 -> 000, 1-> 111
(n=3,k=1,d=3)

In general need $d \geq 3$ to correct one error. Why?

Role of Minimum Distance

Theorem:

A code C with minimum distance "d" can:

1. detect any (d-1) errors
2. recover any (d-1) erasures
3. correct any $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors

Intuition: <board>

Stated another way:

For s-bit error detection $d \geq s + 1$

For s-bit error correction $d \geq 2s + 1$

Block Codes

Code described as: $(n, k, d)_q$ ←

- Σ = alphabet
- $q = |\Sigma|$ = alphabet size
- $C \subseteq \Sigma^n$ (codewords)

Question:

What alphabet did we use so far?

(binary)

(Slight detour into number theory)

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

Q: Examples?

- 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
  $[0\to1, 1\to2, 2\to0] \circ [0\to1, 1\to0, 2\to2] = [0\to0, 1\to2, 2\to1]$

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

Groups based on modular arithmetic

The group of positive integers modulo a prime *p*

$Z_p^* \equiv \{1, 2, 3, \ldots, p\text{-}1\}$ $\quad *_p \equiv$ multiplication modulo p

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

Required properties

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

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

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

Fields

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

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

Example: The reals and rationals with + and * are fields.

The **order (or size)** of a field is the number of elements.

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

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. $(\mathbb{Z}_p \setminus 0, *)$ 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)?

GF(2^n)

Another notation: $\mathbb{F}_{2^n}$

Has 2^n elements

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

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

Desired Properties

We look for codes with the following properties:

1. Good rate: k/n should be high (low overhead)
2. Good distance: d should be large (good error correction)
3. Fast encoding and decoding
4. Smaller alphabet (lower finite field size)
5. Small block size k (helps with latency)
6. Others (application specific): want to handle bursty/random errors, local decodability, ...

Q:

If no structure in the code, how would one perform encoding?

Gigantic lookup table!

If no structure in the code, encoding is highly inefficient.

A common kind of structure added is **linearity**

Linear Codes

If $\mathbb{F}$ is a finite field, then $\mathbb{F}^n$ is a vector space

Definition: C is a linear code if it is a linear subspace of $\mathbb{F}^n$ of dimension k.

This means that there is a set of k independent vectors $v_i \in \mathbb{F}^n$ $(1 \le i \le k)$ that span the subspace.

i.e. every codeword can be written as:

$$c = a_1 v_1 + a_2 v_2 + \ldots + a_k v_k \quad \text{where } a_i \in \mathbb{F}$$

"Basis (or spanning) Vectors"

Some Properties of Linear Codes

1. Linear combination of two codewords is a codeword.

2. Minimum distance (d) = weight of least weight (non-zero) codewords

(Weight of a vector refers to the Hamming weight of a vector, which is equal to the number of non-zero symbols in the vector)

$$d = \min_{\substack{c_i, c_j \in C \\ i \neq j}} |c_i - c_j|$$

$$= \min_{\substack{c \in C \\ c \neq 0}} |c|$$

Generator Matrix of a Linear Code

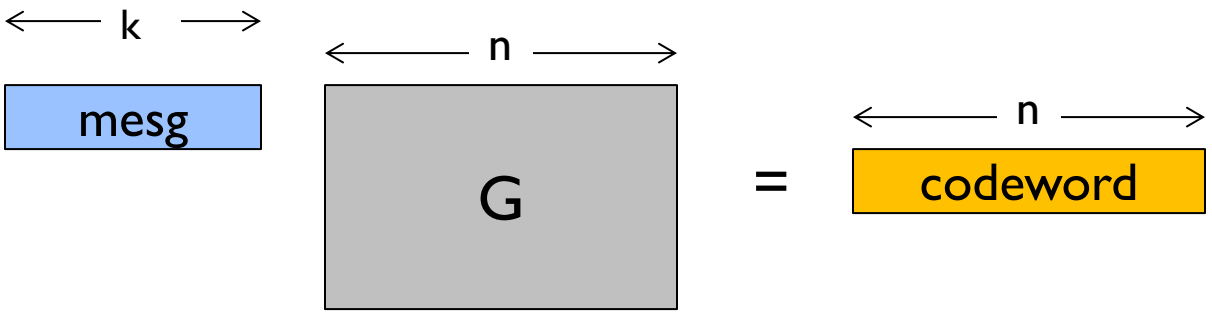
3. Every linear code has a matrix associated with it called the generator matrix.

Generator Matrix:

A k x n matrix **G** such that: C = { mG | m $\in \mathbb{F}^k$ }

(Note: Here vectors are "row vectors".)

Made from stacking the spanning vectors



Encoding is efficient (vector-matrix multiply)

Example and "Standard Form"

"Standard form" of G for systematic codes: $[I_k \; A]$.

$$G = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]$$

(7,4,3) Hamming code

Singleton bound

Theorem: For every $(n, k, d)_q$ code, $n \geq (k + d - 1)$

Another way to look at this: $d \leq (n - k + 1)$

(We will not go into the proof of this theorem in this course due to limited time on this topic.)

Codes that meet Singleton bound with equality are called

Maximum Distance Separable (MDS)

Maximum Distance Separable (MDS)

Only two binary MDS codes!

Q: What are they?

1. Repetition codes (k = 1)
2. Single-parity check codes (n-k = 1)

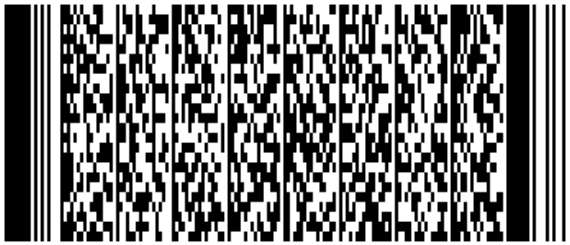
Need to go beyond the binary alphabet.

Finite fields!

Reed-Solmon (RS) codes

One of the most widely codes

- Storage systems, communication systems
- Bar codes (2-dimensional Reed-Solomon bar codes)



PDF-417

QR code

Aztec code

DataMatrix code

images: wikipedia

RS code: Polynomials viewpoint

Message: $[m_0, m_1, \ldots, m_{k-1}]$ where $m_i \in GF(q)$

Consider the polynomial of degree k-1

$$P(x) = m_{k-1} x^{k-1} + \cdots + m_1 x + m_0$$

RS code: Codeword: $[P(\alpha_1), P(\alpha_2), \ldots, P(\alpha_n)]$

(distinct α_i's)

To make the α_i's in $P(\alpha_i)$ distinct, need field size $q \geq n$

That is, need sufficiently large field size for desired codeword length.

Polynomials and their degrees

Fundamental theorem of Algebra: Any non-zero polynomial of degree m has at most m roots (over any field).

Corollary 1: If two degree-m polynomials P, Q agree on m+1 points (i.e., if $P(x_i) = Q(x_i)$ for $x_0, x_1, \ldots, x_m$), then P = Q.

Corollary 2: Given any m+1 points (x_i, y_i), there is at most one degree-m polynomial that has $P(x_i) = y_i$ for all these i.

Theorem: Given any m+1 points (x_i, y_i), there is exactly one degree-m polynomial that has $P(x_i) = y_i$ for all these i.

Proof: e.g., use Lagrange interpolation.

In our case, m=k-1

Minimum distance of an (n, k) RS code

Theorem: RS codes have minimum distance $\mathrm{d} = (n - k + 1)$

Proof: Any ideas?

Hint: Is it a linear code?

1. *RS is a linear code:* if we add two codewords corresponding to P(x) and Q(x), we get a codeword corresponding to the polynomial P(x) + Q(x). Similarly any linear combination..
2. *So look at the least weight codeword*. It is the evaluation of a polynomial of degree k-1 at some n points. So it can be zero on only k-1 points. Hence non-zero on at most (n-(k-1)) points. This means distance at least n-k+1

Apply Singleton bound

Meets Singleton bound: RS codes are MDS

Decoding: Recovering Erasures

Recovering from at most (d-1) erasures:

Received codeword:

[P(α_1), *, P(α_2), …,*, P(α_n)]: at most (d-1) symbols erased

(where * = erased)

Ideas?

1. At most n-k symbols erased
2. So have P(α_i) for at least k evaluations
3. Interpolation to recover the polynomial

Matrix viewpoint: Solving system of linear equations

Decoding: Correcting Errors

Correcting s errors: (d ≥ 2s+1 $\Rightarrow n \geq k + 2s$)

Naïve algo:

- Find k+s symbols that agree on a degree (k-1) poly P(x).
  - There must exist one: since originally k + 2s symbols agreed and at most s are in error
    (i.e., "guess" the n-s uncorrupted locations)

- Can we go wrong?
  Are there k+s symbols that agree on the wrong degree (k-1) polynomial P'(x)? No.
  - 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)

Decoding: Correcting Errors

Correcting s errors: (d ≥ 2s+1)

Naïve algo:

- Find k+s symbols that agree on a degree (k-1) poly P(x).
  - There must exist one: since originally k + 2s symbols agreed and at most s are in error
    (i.e., "guess" the n-s uncorrupted locations)

This suggests a brute-force approach, very inefficient.
"guess" = "enumerate", so time is (n choose s) ~ n^s.

More efficient algorithms exist: <polynomial>
"The Berlekamp Welch Algorithm" (results in solving a system of n linear equations; uses "error" polynomials)
$O(n^2)$ algorithms exist