AGENDA

1. Finishing up dual RS codes
2. Berlekamp-Welch
3. Berlekamp-Massey [sketch]
4. Recall the definition of RS codes:

**DEF. (REED-SOLOMON CODES)**

Let \( n \geq k, q \geq n \). The REED-SOLOMON CODE of dimension \( k \) over \( \mathbb{F}_q \), with evaluation points \( \tilde{\alpha} = (\alpha_1, ..., \alpha_n) \), is

\[
\text{RS}_q(\tilde{\alpha}, n, k) = \{ f(\alpha_1), f(\alpha_2), ..., f(\alpha_n) : f \in \mathbb{F}_q[X], \deg(f) \leq k - 1 \}
\]

Last time, we saw that they meet the Singleton bound.

[Need to finish up RS duality - see LECTURE 4 notes]

HISTORIC ASIDE. RS codes were invented by Reed + Solomon in 1960.

At the time, they didn’t have any fast decoding algs, so they were sort of neat but not that useful. But in the late 1960’s, Peterson, Berlekamp-Massey developed an \( O(n^2) \)-time alg, which can be made to run in time \( O(n \log(n)) \) with FFT tricks. Then RS codes started to be used all over the place! CDs, satellites, QR codes, ...

In 1986, Welch + Berlekamp came up with another decoding alg — it is a bit slower but it is really pretty, so we’ll start with that.
Problem (Decoding $R_{q}(n, k)$ from $e \leq \left\lfloor \frac{n-k}{2} \right\rfloor$ Errors)

Given $w = (w_1, \ldots, w_n) \in \mathbb{F}_q^n$, find a polynomial $f \in \mathbb{F}_q[x]$ so that:

1. $\deg(f) < k$
2. $f(x_i) \neq w_i$ for at most $e \leq \left\lfloor \frac{n-k}{2} \right\rfloor$ values of $i$,

or else return ⊥ if no such polynomial exists.

Idea: Consider the polynomial $E(X) = \prod_{i: w_i \neq f(x)} (X - \alpha_i)$. This is called the "error locator polynomial." (Notice that we don't know what it is...)

Then $\forall i, w_i \cdot E(\alpha_i) = f(\alpha_i) \cdot E(\alpha_i)$

Call this $Q(\alpha_i)$

Algorithm (Berlekamp-Welch)

1. Find:
   a. a monic degree $e$ polynomial $E(X)$
   b. a degree $\leq e + k - 1$ polynomial $Q(X)$

   so that: $w_i \cdot E(\alpha_i) = Q(\alpha_i) \forall i$ \hspace{1cm} (*)

   If it doesn't exist, RETURN ⊥.

2. Let $\hat{f}(X) = Q(X)/E(X)$

   If $\Delta(f, w) > e$:
       RETURN ⊥
   RETURN $\hat{f}$
Two questions:
1. How do we find such polys?
2. Once we do, why is it correct to return $Q/E$? What if we didn't find the “correct” $Q$ and $E$?

Let's answer question 2 first.

Claim: If there is a degree $\leq k-1$ poly $f$ s.t. $\Delta(f,w) \leq e$, then there exists $E$ and $Q$ satisfying $(*)$.

Proof: Let $E(X) = \prod_{i \neq f(w)} (X-w_i) \cdot X^{e-\Delta(f,w)}$.
Let $Q(X) = E(X) \cdot f(X)$.

Claim: Suppose that $(E_1,Q_1)$, $(E_2,Q_2)$ both satisfy the requirements in step 1. Then:

\[
\frac{Q_1(X)}{E_1(X)} = \frac{Q_2(X)}{E_2(X)}
\]

Proof: Consider $R(X) = \frac{Q_1(X)E_2(X) - Q_2(X)E_1(X)}{deg E + k - 1}$.\[\deg R \leq 2e + k - 1, \text{ and } \forall i \in \mathbb{Z} \implies R(x_i) = 0.

\[
R(x_i) = [w_i \cdot E_1(x_i)] \cdot E_2(x_i) - [w_i \cdot E_2(x_i)] \cdot E_1(x_i) = 0
\]

Hence $R$ has at least $n$ roots. Since $e < \frac{n-k+1}{2}$, $2e + k - 1 < n$.

So $R \equiv 0$ is the all-zero polynomial. (Low degree polynomials don't have too many roots.)
Together, these CLAIMS answer Question 2.

Moving on to Question 1. How do we find \( E, Q \)? POLYNOMIAL INTERPOLATION!

More precisely, we want:

\[
W_i \cdot E(x_i) = Q(x_i) \quad \text{for } i = 1, \ldots, n,
\]

\[
\text{n linear constraints.}
\]

\[
\text{deg}(E) = e, \quad E \text{ monic}
\]

\[
\text{deg}(Q) \leq e + k - 1.
\]

\[
\text{e} + (e+k) = 2e + k \text{ variables, which are the coefficients on these two polynomials.}
\]

We already know (from CLAIM 1) that a solution exists (assuming \( f \) does).

So solve this system of eqs to find \( k \)!

\[
\left[ \text{Notice that } 2e + k < (n-k+1) + k \leq n, \text{ so the system looks like,} \right]
\]

\[
\begin{bmatrix}
\text{2e+k vars} \\
\text{n constants}
\end{bmatrix} = \begin{bmatrix}
\end{bmatrix}
\]

But we don't actually care if the system is over or under-determined; now that we know that a solution exists and that any solution will do.

Running Time of BERLEKAMP-WELCH:

- Step 1 takes time \( O(n^3) \) for polynomial division
- Step 2 takes time \( O(n^3) \) for Gaussian Elimination

\[\Rightarrow O(n^3) \text{ total.}\]
The Berlekamp-Massey algorithm is more efficient than the Berlekamp-Welch alg, especially when the # errors is small. Also, it turns out to be really nice to implement in hardware, although we won't go into that.

Let \( H \) be the parity-check matrix for our RS code. I'm actually going to cheat a bit and add a row of ones on top, so that \( H = G^T \) for some RS generator matrix \( G \), since it makes the exposition a bit nicer. Everything in sight is a generalized RS code, so it doesn't matter too much.

So let

\[
H = \begin{bmatrix}
1 & 1 & 1 & 1 & \cdots & 1 \\
1 & \gamma & \gamma^2 & \gamma^3 & \cdots & \gamma^{n-1} \\
1 & \gamma^2 & \gamma^4 & \gamma^6 & \cdots & \gamma^{2(n-1)} \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \gamma^{n-k-1} & \cdots & \gamma^{(n-k)(n-1)}
\end{bmatrix}
\]

We will do SYNDROME DECODING (like for Hamming codes). That is, suppose \( w = c + e \), and we can compute

\[
H \cdot w = H \cdot c + H \cdot e = H \cdot e, \quad \text{since} \quad H \cdot c = 0 \quad \forall \ c \in C.
\]

Our goal will be to use \( H \cdot e \) (the "SYNDROME") to recover \( E(X) \), the error locator polynomial.

\[
E(X) = \prod_{i : e_i \neq 0} (X - \gamma^i)
\]
We don't have direct access to $e$, but we do have access to $H \cdot e$.

Consider, for some vector $\left( f_0, f_1, \ldots, f_{n-k-1} \right)$,

$$
\begin{bmatrix}
  f_0 & f_1 & \ldots & f_{n-k-1}
\end{bmatrix}
$$

This we can compute, since we know $H \cdot e$.

However, if we remember that $H = G^T$, this is also equal to

$$
\begin{bmatrix}
  f(0) & f(1) & \ldots & f(n-1)
\end{bmatrix}
$$

where $f(x) = \sum_{i=0}^{n-k-1} f_i \cdot x^i$.

So, we can actually compute $\langle f, e \rangle$ for any $f$ with $\deg(f) < n-k$.

Our goal is to use this power to recover $e$.

Actually, we are going to recover $E(X)$, then factor it to learn $e$.

**Observation.** $\langle e, X^r \cdot E(X) \rangle = 0 \ \forall r$.

**Proof:** $\langle e, X^r \cdot E(X) \rangle = \sum_{i=0}^{n-1} e_i \cdot y_i^r \cdot E(y_i)$

**Our Plan:** Let's find some poly $f$ s.t. $\deg(f) = t$ and $\langle e, X^r \cdot f(x) \rangle = 0$ for $r = 0, \ldots, t-1$. It's not immediately clear that this is a good plan...
... but in fact it is a good plan:

**Prop.** Suppose that \( \text{wt}(e) = t \), and that \( \langle e, X^r \cdot f(X) \rangle = 0 \) for \( r = 0, \ldots, t-1 \). Then \( E(X) \mid f(X) \).

In particular, if \( \deg(f) \leq t \), \( E(X) = \alpha \cdot f(X) \) for some \( \alpha \in F^* \).

**Proof.** If \( \langle e, X^r \cdot f(X) \rangle = 0 \) \( \forall r = 0, \ldots, t-1 \), then by linearity,

\[
\langle e, g(X) \cdot f(X) \rangle = 0 \quad \forall g \in F \llbracket X \rrbracket \text{ with } \deg(g) \leq t-1.
\]

For any \( k \) s.t. \( e_k = 1 \), let \( g_k(X) = \frac{E(X)}{X - \gamma^k} = \prod_{i \neq 0} (X - \gamma^i) \).

Then \( \deg(g_k) \leq t-1 \), hence

\[
0 = \langle e, g(X) \cdot f(X) \rangle = \sum_{i=0}^{n-1} e_i \cdot g(\gamma^i) \cdot f(\gamma^i) = e_k \cdot g_k(\gamma^k) \cdot f(\gamma^k).
\]

Hence, \( f(\gamma^k) = 0 \). So \( (X - \gamma^k) \mid f(X) \) \( \forall k \) s.t. \( e_k = 1 \), and \( E(X) \mid f(X) \).

OK, so our plan is a good one. Let's try to find \( f \) so that:

- \( \deg(f) \leq t \)
- \( \langle e, X^r \cdot f(X) \rangle = 0 \) \( \forall r = 0, \ldots, t-1 \).

To this end, define:

- \( \text{span}(f) = \) the smallest \( r \) s.t. \( \langle e, X^r \cdot f(X) \rangle \neq 0 \)
- \( \text{disc}(f) = \langle e, X^\text{span}(f) \cdot f(X) \rangle \) is the nonzero value.
**USEFUL LEMMA:** If $\deg(g) \leq \text{span}(f)$, then $\deg(g) + \text{span}(g) \leq \deg(f) + \text{span}(f)$.

**Proof.** First, suppose that $\deg(g) = \text{span}(f)$, say $g(x) = \alpha \cdot X^{\text{span}(f)} + \text{STUFF}$.

Then $\langle e, g(x) \cdot f(x) \rangle = \langle e, (g(x) - \alpha X^{\text{span}(f)}) \cdot f(x) \rangle + \langle e, \alpha \cdot X^{\text{span}(f)} \cdot f(x) \rangle$.

In particular, ONE of the terms that shows up in $f$, say $X^c$ for $c \leq \deg(f)$, has $\langle e, g(x) \cdot X^c \rangle \neq 0$, hence $\text{span}(g) \leq c \leq \deg(f)$.

Then $\deg(g) + \text{span}(g) \leq \text{span}(f) + \deg(f)$.

Next, if $\deg(g) < \text{span}(f)$, apply the above to $X^{\text{span}(f) - \deg(g)} \cdot g(x)$.

**Cor.** If $\text{span}(f) = t$ then $\text{span}(f) = \infty$.

**Proof.** Say $\text{span}(f) \geq t$ but is finite. Then $\deg(E) = t \leq \text{span}(f)$, so by the **USEFUL LEMMA**, $\deg(E) + \text{span}(E) \leq \deg(f) + \text{span}(f)$.

CONTRACTION!
Again, our goal is to come up with some function with large span. The following lemma will tell us how to get this.

**Lemma**

Suppose \( \text{span}(f) = r \), \( \text{disc}(f) = \mu \)

Suppose \( \text{span}(g) = c \), \( \text{disc}(g) = \nu \)

AND say that \( c < r \).

Then \( h(x) = f(x) - \left( \frac{\mu}{\nu} \right) \cdot x^{c-r} \cdot g(x) \) has

\( \text{span}(h) > \text{span}(f) \).

The point of this lemma is that, given \( f \) and \( g \) with reasonably close spans, we can combine them to get \( h \) with a strictly bigger span and degree not too much larger.

**Proof.** Just consider

\[
\langle e, x^i \cdot \left[ f(x) - \left( \frac{\mu}{\nu} \right) X^{c-r} g(x) \right] \rangle \\
= \langle e, x^i f(x) \rangle - \left( \frac{\mu}{\nu} \right) \langle e, x^{c-r+i} g(x) \rangle
\]

If \( i < r \), then both terms are 0 since \( \text{sp}(f) = r \), \( \text{sp}(g) = c \).
If \( i = r \), then we have \( \mu - \left( \frac{\mu}{\nu} \right) \cdot \nu = 0 \).

Hence \( \text{sp}(h) > r \).
**Algorithm (Berlekamp-Massey):**

1. Initialize \( f \leftarrow 1, \quad g \leftarrow 0 \)

2. For \( m = 0, \ldots, 2t - 1 \):
   - \( c \leftarrow \deg(f) - 1 \)
   - \( r \leftarrow m - c - 1 \) \( (= m - \deg(f)) \)
   - \( \mu \leftarrow \langle e_2, \chi^r f(X) \rangle \)

   - If \( \mu = 0 \) or \( r < c \):
     - \( f'(X) \leftarrow f(X) - \mu \cdot \chi^{c-r} g(X) \)
     - \( g'(X) \leftarrow g(X) \)
   - Else:
     - \( f'(X) \leftarrow \chi^{r-c} f(X) - \mu g(X) \)
     - \( g'(X) \leftarrow \frac{1}{\mu} f(X) \)

3. **RETURN** \( f(X) \)

**Claim.** This algorithm maintains:

After iteration \( m \):

- \( f \) is monic and \( \deg(f) + \text{span}(f) > m \)
- EITHER \( g = 0 \)
- OR: \( \text{span}(g) = \deg(f) - 1 \)
  - \( \text{span}(g) + \deg(g) \leq m \)
  - \( \text{disc}(g) = 1 \)
This proof skipped in class.

Proof. The base case (after $m = -1$) is easy.

Let $m \geq -1$ and assume by induction that

1. $f$ is monic and $\deg(f) + \text{span}(f) > m$
2. EITHER $g = 0$ OR:
   - $\text{span}(g) = \deg(f) - 1$
   - $\text{span}(g) + \deg(g) \leq m$
   - $\text{disc}(g) = 1$.

CASE 1. $\mu = 0$. Then $f$ and $g$ are unchanged.

So the stuff (2) about $g$ is good.

Further, since $0 = \mu = \langle e, X^r f(x) \rangle = \langle e, X^{m-\deg(f)} f(x) \rangle,$

we have $\text{span}(f) > m - \deg(f)$, hence $\text{span}(f) + \deg(f) > m$, so (1) holds.

CASE 2. $\mu \neq 0$.

CASE 2A. $r \leq c$.

Then $f' \leftarrow f(x) - \mu \cdot X^{c-r} g(x)$.

By induction,

\[
\text{span}(g) = \deg(f) - 1 = c \quad \text{by our choice of } c.
\]

Hence $\text{span}(X^{c-r} g(x)) = r$.

So both $f, g$ have span = r, $\text{disc}(f) = \mu$, $\text{disc}(g) = 1$, so this update is precisely

the one from the LEMMA and $\text{sp}(f') > r$.

Moreover, $\deg(f') = \deg(f)$, hence $\deg(f') + \text{sp}(f') > \deg(f) + \text{sp}(f) > m$

$\Rightarrow \deg(f') + \text{sp}(f') > m + 1$.

To see this, notice that $\deg(g) = (\text{sp}(g) + \deg(g)) - \text{sp}(g) \leq m - \text{sp}(g) = m - c$.

So $\deg(X^{c-r} g(x)) \leq (m - c) + (c - r) = m - r < \deg(f)$.

Thus, the update $-\mu \cdot X^{c-r} g(x)$ affects neither the degree, nor the monicness of $f$.

This arg. says also that $f'$ is monic, so (1) holds for $m + 1$.

(2) holds since in this case we did not update $g$.

CASE 2B. $c > r$ is similar. [FUN EXERCISE].
COR. Suppose $\text{wt}(e) = t$.
If $m \geq 2t - 1$, then after iteration $m$, $f(X) = E(X)$.

proof:

First notice that $\deg(f(X)) \leq t$.
Indeed, we've been maintaining $\text{span}(g) = \deg(f) - 1$, so if $\deg(f) > t$ then $g(X) \not\equiv t$.
By the USEFUL LEMMA (or rather, its COR), we have $\text{span}(g) = \infty$.
But we were also maintaining $\text{span}(g) + \deg(g) \leq m$, so that's a $\checkmark$.

Now, $\deg(f) + \text{span}(f) > m$
$\Rightarrow \text{span}(f) > m - \deg(f)$
$= (2t - 1) - t$
$= t - 1$

So $\text{span}(f) > t$.

But this is what we wanted:

$\text{sp}(f) \geq t \Rightarrow E(X) \mid f(X)$
$\deg(f) \leq t \Rightarrow E(X) = \alpha f(X)$ for some $\alpha \in \mathbb{F}_q$
$f$ monic $\Rightarrow E(X) = f(X)$.

Finally, recall that $d = n - k + 1$, and that the algorithm stops working (we stop being able to query $\langle e, X^t f(X) \rangle = \mu$) when $m = n - k$, so we need $2t - 1 \leq n - k - 1$
aka $t \leq \frac{n - k}{2} = \frac{d - 1}{2}$ which is where the algorithm should stop working.

HOWEVER! Notice that if $t$ happens to be smaller, we can actually stop earlier, with only $O(t)$ rounds.
The polys we are working with all have $\deg \leq m = O(t)$, and so we can do everything in $\text{poly}(t)$ computations over $\mathbb{F}_q$. That's sublinear time!

[See, “Syndrome Encoding and Decoding of BCH Codes in Sublinear Time” by Dodis, Ostrovsky, Reyzin, Smith for details about making this real fast.]
All this just finds $E(X)$. We still need to find the roots of $E(X)$, and then figure out how to fix the errors.

- If you get fancy, you can factor $E(X)$ in time $O(1.839 \cdot \log(n))$.
  \[\text{[Subquadratic-time factoring of polynomials over finite fields]}
  \quad \text{Kaltofen + Shoup 1995}\]

- To actually recover the message, we can't hope for sublinear time (since the message has length $k = Rn$), but we can how do that in time $O(n \log(n))$ via linear algebra. [The $n \log(n)$ is bc Vandermonde matrices admit a nice FFT-like alg.]

That finishes the Berlekamp-Massey algorithm.

This algorithm can actually be implemented nicely in hardware. [the update step can be done with a shift register] and so this is the alg. that’s often used in practice for RS codes. (Or, optimized versions of this).

The Berlekamp-Welch alg is certainly easier to understand, though!

**QUESTIONS TO PONDER**

1. Fill in the details for the Berlekamp-Massey alg.
   [there is one FUN EXERCISE in the notes and I anticipate we skipped SOME proofs/indexes]

2. Can you think of any other alg s for RS codes?

3. How would you adapt RS codes / these algorithms to come up with BINARY codes?