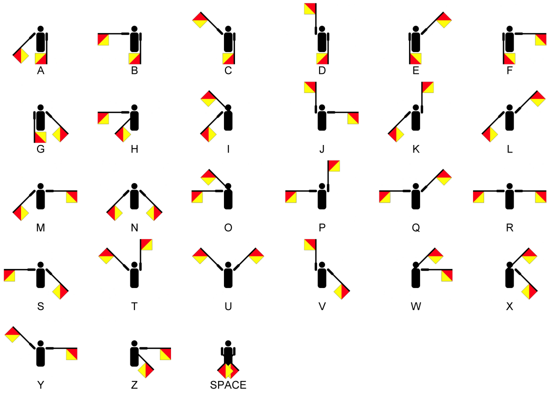
Lecture 10: Error Correcting Codes

WCat Can You 2o Cith A Noisy ChahDel?

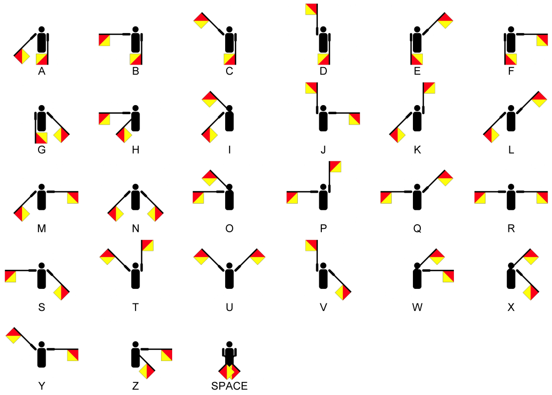
Sema-Five?

Suppose I am trying to communicate via semaphore:



Sema-Five?

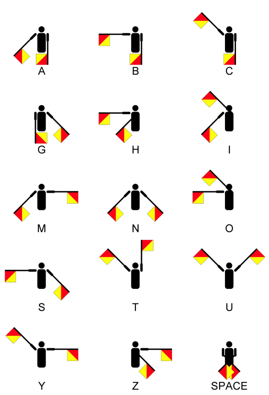
Suppose I am trying to communicate via semaphore:



What if my recipient misses some letters?

Sema-Five?

Suppose I am trying to communicate via semaphore:



What if my recipient misses some letters?

Less silly: deal with dropped internet packets

Problem Statement

Formally: have message in n parts $m_1, \ldots, m_n$
Channel may drop up to k packets sent
How many packets needed to ensure receipt?

Problem Statement

Formally: have message in n parts $m_1, \ldots, m_n$
Channel may drop up to k packets sent
How many packets needed to ensure receipt?

Naïve idea: repetition coding

- Repeat message “enough times”

Problem Statement

Formally: have message in n parts $m_1, \ldots, m_n$
Channel may drop up to k packets sent
How many packets needed to ensure receipt?

Naïve idea: repetition coding

- Repeat message "enough times"

How many reps required to *guarantee* receipt?

Problem Statement

Formally: have message in n parts $m_1, \ldots, m_n$
Channel may drop up to k packets sent
How many packets needed to ensure receipt?

Naïve idea: repetition coding

- Repeat message “enough times”

How many reps required to *guarantee* receipt?
Could drop first packet every time!
Need $k + 1$ repetitions to be safe

Problem Statement

Formally: have message in n parts $m_1, \ldots, m_n$
Channel may drop up to k packets sent
How many packets needed to ensure receipt?

Naïve idea: repetition coding

- Repeat message "enough times"

How many reps required to *guarantee* receipt?
Could drop first packet every time!
Need $k + 1$ repetitions to be safe

For n packet message, send $n(k + 1)$ packets

Problem Statement

Formally: have message in n parts $m_1, \ldots, m_n$
Channel may drop up to k packets sent
How many packets needed to ensure receipt?

Naïve idea: repetition coding

- Repeat message “enough times”

How many reps required to *guarantee* receipt?
Could drop first packet every time!
Need $k + 1$ repetitions to be safe

For n packet message, send $n(k + 1)$ packets
Can we do better?

A Better Encoding

Claim: Can get away with $n + k$ packets

A Better Encoding

Claim: Can get away with $n + k$ packets

How?

A Better Encoding

Claim: Can get away with $n + k$ packets

How? Using polynomials!

A Better Encoding

Claim: Can get away with $n + k$ packets

How? Using polynomials!

Idea: Take prime q st $q > n + k$, $>$ largest message
Encode message as polynomial in $GF(q)$

A Better Encoding

Claim: Can get away with $n + k$ packets

How? Using polynomials!

Idea: Take prime q st $q > n + k$, $>$ largest message
Encode message as polynomial in $GF(q)$

Interpolate poly $p(x)$ st $p(i) = m_i$ for $1 \leq i \leq n$
Send $p(1)$, $p(2)$, ..., $p(n + k)$

Recovery

Claim: With $\leq k$ erasures, recovery always possible

Recovery

Claim: With $\leq k$ erasures, recovery always possible

Proof:

- Suppose receive n points
- Interpolate poly $p'(x)$ through them

Recovery

Claim: With $\leq k$ erasures, recovery always possible

Proof:

- Suppose receive n points
- Interpolate poly $p'(x)$ through them
- $\deg(p) = \deg(p') = n - 1$
- p and p' agree on n points

Recovery

Claim: With $\leq k$ erasures, recovery always possible

Proof:

- Suppose receive n points
- Interpolate poly $p'(x)$ through them
- $\deg(p) = \deg(p') = n - 1$
- p and p' agree on n points
- So $p = p'$

Recovery

Claim: With $\leq k$ erasures, recovery always possible

Proof:

- Suppose receive n points
- Interpolate poly $p'(x)$ through them
- $\deg(p) = \deg(p') = n - 1$
- p and p' agree on n points
- So $p = p'$
- Thus $m_i = p'(i)$ for $1 \leq i \leq n$

Enconding Example

Want to send $m = (4, 0, 5)$, protect for 2 erasures

Enconding Example

Want to send $m = (4, 0, 5)$, protect for 2 erasures

Interpolate polynomial modulo 7:

Enconding Example

Want to send $m = (4, 0, 5)$, protect for 2 erasures

Interpolate polynomial modulo 7:

$$\Delta_1(x) = (x-2)(x-3)[(1-2)(1-3)]^{-1}$$
$$\equiv 4(x^2 - 5x + 6) \equiv 4x^2 + x + 3 \pmod 7$$

Enconding Example

Want to send $m = (4, 0, 5)$, protect for 2 erasures

Interpolate polynomial modulo 7:

$$\Delta_1(x) = (x-2)(x-3)[(1-2)(1-3)]^{-1}$$
$$\equiv 4(x^2 - 5x + 6) \equiv 4x^2 + x + 3 \pmod 7$$

Don't need to calculate $\Delta_2(x)$!

Enconding Example

Want to send $m = (4, 0, 5)$, protect for 2 erasures

Interpolate polynomial modulo 7:

$$\Delta_1(x) = (x-2)(x-3)[(1-2)(1-3)]^{-1}$$
$$\equiv 4(x^2 - 5x + 6) \equiv 4x^2 + x + 3 \pmod 7$$

Don't need to calculate $\Delta_2(x)$!

$$\Delta_3(x) = (x-1)(x-2)[(3-1)(3-2)]^{-1}$$
$$\equiv 4(x^2 - 3x + 2) \equiv 4x^2 + 2x + 1 \pmod 7$$

Enconding Example

Want to send $m = (4, 0, 5)$, protect for 2 erasures

Interpolate polynomial modulo 7:

$$\Delta_1(x) = (x-2)(x-3)[(1-2)(1-3)]^{-1}$$
$$\equiv 4(x^2 - 5x + 6) \equiv 4x^2 + x + 3 \pmod 7$$

Don't need to calculate $\Delta_2(x)$!

$$\Delta_3(x) = (x-1)(x-2)[(3-1)(3-2)]^{-1}$$
$$\equiv 4(x^2 - 3x + 2) \equiv 4x^2 + 2x + 1 \pmod 7$$

$$p(x) = 4\Delta_1(x) + 5\Delta_3(x) \equiv x^2 + 3 \pmod 7$$

Enconding Example

Want to send $m = (4, 0, 5)$, protect for 2 erasures

Interpolate polynomial modulo 7:

$$\Delta_1(x) = (x-2)(x-3)[(1-2)(1-3)]^{-1}$$
$$\equiv 4(x^2 - 5x + 6) \equiv 4x^2 + x + 3 \pmod 7$$

Don't need to calculate $\Delta_2(x)$!

$$\Delta_3(x) = (x-1)(x-2)[(3-1)(3-2)]^{-1}$$
$$\equiv 4(x^2 - 3x + 2) \equiv 4x^2 + 2x + 1 \pmod 7$$

$$p(x) = 4\Delta_1(x) + 5\Delta_3(x) \equiv x^2 + 3 \pmod 7$$

Send $(p(1), p(2), p(3), p(4), p(5)) = (4, 0, 5, 5, 0)$

Recovery Example

Sent: $(4, 0, 5, 5, 0)$; Received: $(-, 0, 5, 5, -)$

Recovery Example

Sent: $(4, 0, 5, 5, 0)$; Received: $(-, 0, 5, 5, -)$
Need to interpolate!

Recovery Example

Sent: $(4, 0, 5, 5, 0)$; Received: $(-, 0, 5, 5, -)$
Need to interpolate!

Don't need $\Delta_2(x)$!

Recovery Example

Sent: $(4, 0, 5, 5, 0)$; Received: $(-, 0, 5, 5, -)$
Need to interpolate!

Don't need $\Delta_2(x)$!

$\Delta_3(x) = (x-2)(x-4)[(3-2)(3-4)]^{-1}$
$\equiv 6(x^2 - 6x + 8) \equiv 6x^2 + 6x + 6$

Recovery Example

Sent: $(4, 0, 5, 5, 0)$; Received: $(-, 0, 5, 5, -)$
Need to interpolate!

Don't need $\Delta_2(x)$!

$\Delta_3(x) = (x-2)(x-4)[(3-2)(3-4)]^{-1}$
$\equiv 6(x^2 - 6x + 8) \equiv 6x^2 + 6x + 6$

$\Delta_4(x) = (x-2)(x-3)[(4-2)(4-3)]^{-1}$
$\equiv 4(x^2 - 5x + 6) \equiv 4x^2 + x + 3$

Recovery Example

Sent: $(4, 0, 5, 5, 0)$; Received: $(-, 0, 5, 5, -)$
Need to interpolate!

Don't need $\Delta_2(x)$!

$$\Delta_3(x) = (x-2)(x-4)[(3-2)(3-4)]^{-1}$$
$$\equiv 6(x^2 - 6x + 8) \equiv 6x^2 + 6x + 6$$

$$\Delta_4(x) = (x-2)(x-3)[(4-2)(4-3)]^{-1}$$
$$\equiv 4(x^2 - 5x + 6) \equiv 4x^2 + x + 3$$

Interpolate $p'(x) = 5\Delta_3(x) + 5\Delta_4(x) \equiv x^2 + 3$

Recovery Example

Sent: $(4, 0, 5, 5, 0)$; Received: $(-, 0, 5, 5, -)$
Need to interpolate!

Don't need $\Delta_2(x)$!

$$\Delta_3(x) = (x-2)(x-4)[(3-2)(3-4)]^{-1}$$
$$\equiv 6(x^2-6x+8) \equiv 6x^2+6x+6$$

$$\Delta_4(x) = (x-2)(x-3)[(4-2)(4-3)]^{-1}$$
$$\equiv 4(x^2-5x+6) \equiv 4x^2+x+3$$

Interpolate $p'(x) = 5\Delta_3(x) + 5\Delta_4(x) \equiv x^2+3$

Evaluate for message: $(p'(1), p'(2), p'(3)) = (4, 0, 5)$

Optimality

Claim: Can't guarantee success w/$< n + k$ packets

Optimality

Claim: Can't guarantee success w/$< n + k$ packets

Proof:

- May send one of two messages:
  - $(m_1, m_2, ..., m_{n-1}, m_n)$ or
  - $(m_1, m_2, ..., m_{n-1}, m'_n)$

Optimality

Claim: Can't guarantee success w/$< n + k$ packets

Proof:

- May send one of two messages:
  - $(m_1, m_2, \ldots, m_{n-1}, m_n)$ or
  - $(m_1, m_2, \ldots, m_{n-1}, m'_n)$
- Channel drops nth packet and all extras

Optimality

Claim: Can't guarantee success w/$< n + k$ packets

Proof:

- May send one of two messages:
  - $(m_1, m_2, \ldots, m_{n-1}, m_n)$ or
  - $(m_1, m_2, \ldots, m_{n-1}, m'_n)$
- Channel drops nth packet and all extras
- Which message was sent?

Optimality

Claim: Can't guarantee success w/$< n + k$ packets

Proof:

- May send one of two messages:
  - $(m_1, m_2, \ldots, m_{n-1}, m_n)$ or
  - $(m_1, m_2, \ldots, m_{n-1}, m'_n)$
- Channel drops nth packet and all extras
- Which message was sent?
- Impossible to know!

C0rrupt1on Err0rs

More difficult: what if packets are corrupted?

C0rrupt1on Err0rs

More difficult: what if packets are corrupted?

Don't know which packets are wrong!

C0rrupt1on Err0rs

More difficult: what if packets are corrupted?
Don't know which packets are wrong!

Claim: Previous encoding not good enough

C0rrupt1on Err0rs

More difficult: what if packets are corrupted?
Don't know which packets are wrong!

Claim: Previous encoding not good enough

Proof:

- Again, two possible original messages:
  - $(m_1, ..., m_{n-1}, m_n)$ or
  - $(m_1, ..., m_{n-1}, m'_n)$

C0rrupt1on Err0rs

More difficult: what if packets are corrupted?
Don't know which packets are wrong!

Claim: Previous encoding not good enough

Proof:

- Again, two possible original messages:
  - $(m_1, \ldots, m_{n-1}, m_n)$ or
  - $(m_1, \ldots, m_{n-1}, m_n')$
- First n rec'd match 1st, but next k match 2nd

C0rrupt1on Err0rs

More difficult: what if packets are corrupted?
Don't know which packets are wrong!

Claim: Previous encoding not good enough

Proof:

- Again, two possible original messages:
    - $(m_1, \ldots, m_{n-1}, m_n)$ or
    - $(m_1, \ldots, m_{n-1}, m'_n)$
- First n rec'd match 1st, but next k match 2nd
- Which message was sent?

C0rrupt1on Err0rs

More difficult: what if packets are corrupted?
Don't know which packets are wrong!

Claim: Previous encoding not good enough

Proof:

- Again, two possible original messages:
    - $(m_1, ..., m_{n-1}, m_n)$ or
    - $(m_1, ..., m_{n-1}, m'_n)$
- First n rec'd match 1st, but next k match 2nd
- Which message was sent?
- Impossible to know!

C0rrupt1on Err0rs

More difficult: what if packets are corrupted?
Don't know which packets are wrong!

Claim: Previous encoding not good enough

Proof:

- Again, two possible original messages:
  - $(m_1, ..., m_{n-1}, m_n)$ or
  - $(m_1, ..., m_{n-1}, m'_n)$
- First n rec'd match 1st, but next k match 2nd
- Which message was sent?
- Impossible to know!

Note: works for *any* padding by k packets

NEED MOAR PACKETS

Theorem: For k corruptions, need $\geq n + 2k$ packets

NEED MOAR PACKETS

Theorem: For k corruptions, need $\geq n + 2k$ packets

Suppose only send $2k - 1$ extra packets

NEED MOAR PACKETS

Theorem: For k corruptions, need $\geq n + 2k$ packets

Suppose only send $2k - 1$ extra packets
Consider two possible messages:

$$(m_1, m_2, ..., m_{n-1}, m_n, e_1, ..., e_{k-1}, e_k, ..., e_{2k-1})$$

$$(m_1, m_2, ..., m_{n-1}, m'_n, e'_1, ..., e'_{k-1}, e'_k, ..., e'_{2k-1})$$

NEED MOAR PACKETS

Theorem: For k corruptions, need $\geq n + 2k$ packets

Suppose only send $2k - 1$ extra packets
Consider two possible messages:

$$(m_1, m_2, \dots, m_{n-1}, m_n, e_1, \dots, e_{k-1}, e_k, \dots, e_{2k-1})$$

$$(m_1, m_2, \dots, m_{n-1}, m'_n, e'_1, \dots, e'_{k-1}, e'_k, \dots, e'_{2k-1})$$

$$\downarrow$$

$$(m_1, m_2, \dots, m_{n-1}, m_n, e_1, \dots, e_{k-1}, e'_k, \dots, e'_{2k-1})$$

NEED MOAR PACKETS

Theorem: For k corruptions, need $\geq n + 2k$ packets

Suppose only send $2k - 1$ extra packets
Consider two possible messages:

$$(m_1, m_2, ..., m_{n-1}, m_n, e_1, ..., e_{k-1}, \boxed{e_k, ..., e_{2k-1}})$$

$$(m_1, m_2, ..., m_{n-1}, m'_n, e'_1, ..., e'_{k-1}, e'_k, ..., e'_{2k-1})$$

$$\downarrow$$

$$(m_1, m_2, ..., m_{n-1}, m_n, e_1, ..., e_{k-1}, e'_k, ..., e'_{2k-1})$$

NEED MOAR PACKETS

Theorem: For k corruptions, need $\geq n + 2k$ packets

Suppose only send $2k - 1$ extra packets
Consider two possible messages:

$$(m_1, m_2, \ldots, m_{n-1}, m_n, e_1, \ldots, e_{k-1}, \boxed{e_k, \ldots, e_{2k-1}})$$

$$(m_1, m_2, \ldots, m_{n-1}, \boxed{m'_n, e'_1, \ldots, e'_{k-1}}, e'_k, \ldots, e'_{2k-1})$$

$$\downarrow$$

$$(m_1, m_2, \ldots, m_{n-1}, m_n, e_1, \ldots, e_{k-1}, e'_k, \ldots, e'_{2k-1})$$

NEED MOAR PACKETS

Theorem: For k corruptions, need $\geq n + 2k$ packets

Suppose only send $2k - 1$ extra packets
Consider two possible messages:

$$(m_1, m_2, \ldots, m_{n-1}, m_n, e_1, \ldots, e_{k-1}, \boxed{e_k, \ldots, e_{2k-1}})$$

$$(m_1, m_2, \ldots, m_{n-1}, \boxed{m'_n, e'_1, \ldots, e'_{k-1}}, e'_k, \ldots, e'_{2k-1})$$

$$\Big\downarrow$$

$$(m_1, m_2, \ldots, m_{n-1}, m_n, e_1, \ldots, e_{k-1}, e'_k, \ldots, e'_{2k-1})$$

Don't know which message originally sent!

Relaaaaax

Take a 4 minute break!

Relaaaaax

Take a 4 minute break!

Today's Discussion Question:
What's your strangest family tradition?

Corruption Recovery

Theorem: If use previous encoding with $2k$ extra packets, can recover from k corruptions.

Corruption Recovery

Theorem: If use previous encoding with $2k$ extra packets, can recover from k corruptions.

How?

Corruption Recovery

Theorem: If use previous encoding with $2k$ extra packets, can recover from k corruptions.

How? Find deg $n-1$ poly through $n+k$ points

Corruption Recovery

Theorem: If use previous encoding with $2k$ extra packets, can recover from k corruptions.

How? Find deg $n-1$ poly through $n+k$ points

Claim: Such a poly exists

Corruption Recovery

Theorem: If use previous encoding with $2k$ extra packets, can recover from k corruptions.

How? Find deg $n-1$ poly through $n+k$ points

Claim: Such a poly exists

- Original poly through $n+k$ uncorrupted points

Corruption Recovery

Theorem: If use previous encoding with $2k$ extra packets, can recover from k corruptions.

How? Find deg $n-1$ poly through $n+k$ points

Claim: Such a poly exists

- Original poly through $n+k$ uncorrupted points

Claim: Only one such poly

Corruption Recovery

Theorem: If use previous encoding with $2k$ extra packets, can recover from k corruptions.

How? Find deg $n-1$ poly through $n+k$ points

Claim: Such a poly exists

- Original poly through $n+k$ uncorrupted points

Claim: Only one such poly

- For any $n+k$ points, at least n uncorrupted
- Those n define the original polynomial

Efficiency?

How long does it take to recover?

Efficiency?

How long does it take to recover?

Naïvely, need to try all possible sets of k corruptions

Efficiency?

How long does it take to recover?

Naïvely, need to try all possible sets of k corruptions
$\binom{n+2k}{k} \approx (\frac{n+2k}{k})^k$ possibilities — much too slow

Efficiency?

How long does it take to recover?

Naïvely, need to try all possible sets of k corruptions $\binom{n+2k}{k} \approx (\frac{n+2k}{k})^k$ possibilities — much too slow

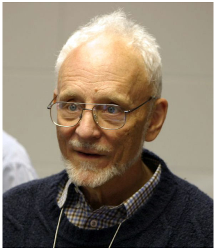
State-of-the-art for over 25 years! (1960 - 1986)

Efficiency?

How long does it take to recover?

Naïvely, need to try all possible sets of k corruptions
$\binom{n+2k}{k} \approx (\frac{n+2k}{k})^k$ possibilities — much too slow

State-of-the-art for over 25 years! (1960 - 1986)


Elwyn Berlekamp

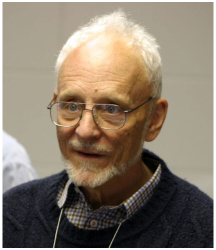
Lloyd Welch

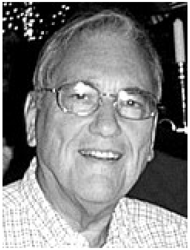
Efficiency?

How long does it take to recover?

Naïvely, need to try all possible sets of k corruptions $\binom{n+2k}{k} \approx (\frac{n+2k}{k})^k$ possibilities — much too slow

State-of-the-art for over 25 years! (1960 - 1986)


Elwyn Berlekamp


Lloyd Welch

Berlekamp-Welch Recovery

Main idea: have (unknown) error-location poly

$e(x) = (x - e_1)(x - e_2)...(x - e_k)$

Berlekamp-Welch Recovery

Main idea: have (unknown) error-location poly

$e(x) = (x - e_1)(x - e_2)...(x - e_k)$

If can find this poly, can fix corruptions!

Berlekamp-Welch Recovery

Main idea: have (unknown) error-location poly

$e(x) = (x - e_1)(x - e_2)...(x - e_k)$

If can find this poly, can fix corruptions!

Define (unknown) $q(x) = p(x)e(x)$ to help solve

Berlekamp-Welch Recovery

Main idea: have (unknown) error-location poly

$e(x) = (x - e_1)(x - e_2)\ldots(x - e_k)$

If can find this poly, can fix corruptions!

Define (unknown) $q(x) = p(x)e(x)$ to help solve
Claim: $q(i) = r_i e(i)$ for all i

Berlekamp-Welch Recovery

Main idea: have (unknown) error-location poly

$e(x) = (x - e_1)(x - e_2)...(x - e_k)$

If can find this poly, can fix corruptions!

Define (unknown) $q(x) = p(x)e(x)$ to help solve
Claim: $q(i) = r_i e(i)$ for all i

- If i error, both sides zero
- Otherwise $r_i = p(i)$, so true by definition

Berlekamp-Welch Recovery

Main idea: have (unknown) error-location poly

$e(x) = (x - e_1)(x - e_2)...(x - e_k)$

If can find this poly, can fix corruptions!

Define (unknown) $q(x) = p(x)e(x)$ to help solve
Claim: $q(i) = r_i e(i)$ for all i

- If i error, both sides zero
- Otherwise $r_i = p(i)$, so true by definition

Gives $n + 2k$ equations known to be true!
Unknowns are coefficients for $q(x)$ and $e(x)$

Berlekamp-Welch: A Closer Look

What does $q(x)$ look like?

Berlekamp-Welch: A Closer Look

What does $q(x)$ look like?
$\deg(p) = n - 1$, $\deg(e) = k$, so $\deg(q) = n + k - 1$

Berlekamp-Welch: A Closer Look

What does $q(x)$ look like?
$\deg(p) = n - 1$, $\deg(e) = k$, so $\deg(q) = n + k - 1$
$q(x) = a_{n+k-1}x^{n+k-1} + \ldots + a_1 x + a_0$

Berlekamp-Welch: A Closer Look

What does $q(x)$ look like?
$\deg(p) = n - 1$, $\deg(e) = k$, so $\deg(q) = n + k - 1$
$q(x) = a_{n+k-1}x^{n+k-1} + ... + a_1 x + a_0$

What does $e(x)$ look like?

Berlekamp-Welch: A Closer Look

What does $q(x)$ look like?
$\deg(p) = n - 1$, $\deg(e) = k$, so $\deg(q) = n + k - 1$
$q(x) = a_{n+k-1}x^{n+k-1} + ... + a_1x + a_0$

What does $e(x)$ look like?
$e(x) = (x - e_1)(x - e_2)...(x - e_k)$, so degree k

Berlekamp-Welch: A Closer Look

What does $q(x)$ look like?
$\deg(p) = n - 1$, $\deg(e) = k$, so $\deg(q) = n + k - 1$
$q(x) = a_{n+k-1}x^{n+k-1} + ... + a_1x + a_0$

What does $e(x)$ look like?
$e(x) = (x - e_1)(x - e_2)...(x - e_k)$, so degree k
$e(x) = b_kx^k + ... + b_1x + b_0$

Berlekamp-Welch: A Closer Look

What does $q(x)$ look like?
$\deg(p) = n - 1$, $\deg(e) = k$, so $\deg(q) = n + k - 1$
$q(x) = a_{n+k-1}x^{n+k-1} + \ldots + a_1 x + a_0$

What does $e(x)$ look like?
$e(x) = (x - e_1)(x - e_2)\ldots(x - e_k)$, so degree k
$e(x) = b_k x^k + \ldots + b_1 x + b_0$

But wait! $b_k = 1$ for any $e_1, \ldots, e_k$!
So $e(x) = x^k + b_{k-1}x^{k-1} + \ldots + b_1 x + b_0$

Berlekamp-Welch: A Closer Look

What does $q(x)$ look like?
$\deg(p) = n - 1$, $\deg(e) = k$, so $\deg(q) = n + k - 1$
$q(x) = a_{n+k-1}x^{n+k-1} + \dots + a_1x + a_0$

What does $e(x)$ look like?
$e(x) = (x - e_1)(x - e_2)\dots(x - e_k)$, so degree k
$e(x) = b_kx^k + \dots + b_1x + b_0$

But wait! $b_k = 1$ for any $e_1, \dots, e_k$!
So $e(x) = x^k + b_{k-1}x^{k-1} + \dots + b_1x + b_0$

Have $n + k$ unknowns from q, k from e
Matches $n + 2k$ linear eqns of the form $q(i) = r_ie(i)$

Berlekamp-Welch: A Closer Look

What does $q(x)$ look like?
$\deg(p) = n - 1$, $\deg(e) = k$, so $\deg(q) = n + k - 1$
$q(x) = a_{n+k-1}x^{n+k-1} + ... + a_1 x + a_0$

What does $e(x)$ look like?
$e(x) = (x - e_1)(x - e_2)...(x - e_k)$, so degree k
$e(x) = b_k x^k + ... + b_1 x + b_0$

But wait! $b_k = 1$ for any $e_1, ..., e_k$!
So $e(x) = x^k + b_{k-1}x^{k-1} + ... + b_1 x + b_0$

Have $n + k$ unknowns from q, k from e
Matches $n + 2k$ linear eqns of the form $q(i) = r_i e(i)$

Linear Algebra: can find q, e, so have $p(x) = \frac{q(x)}{e(x)}$

Berlekamp-Welch: Example

Want to send length 2 message, have 1 corruption
Receive messages $(1,3)$, $(2,1)$, $(3,4)$, $(4,0)$ mod 7

Berlekamp-Welch: Example

Want to send length 2 message, have 1 corruption
Receive messages $(1,3)$, $(2,1)$, $(3,4)$, $(4,0)$ mod 7

$q(x) = a_2x^2 + a_1x + a_0$, $e(x) = x + b_0$

Berlekamp-Welch: Example

Want to send length 2 message, have 1 corruption
Receive messages $(1,3)$, $(2,1)$, $(3,4)$, $(4,0)$ mod 7

$q(x) = a_2x^2 + a_1x + a_0$, $e(x) = x + b_0$

Eq 1: $q(1) = r_1e(1)$, so $a_2 + a_1 + a_0 = 3(1 + b_0)$

Berlekamp-Welch: Example

Want to send length 2 message, have 1 corruption
Receive messages $(1, 3)$, $(2, 1)$, $(3, 4)$, $(4, 0)$ mod 7

$q(x) = a_2x^2 + a_1x + a_0$, $e(x) = x + b_0$

Eq 1: $q(1) = r_1e(1)$, so $a_2 + a_1 + a_0 = 3(1 + b_0)$

Eq 2: $q(2) = r_2e(2)$, so $4a_2 + 2a_1 + a_0 = 1(2 + b_0)$

Berlekamp-Welch: Example

Want to send length 2 message, have 1 corruption
Receive messages $(1,3)$, $(2,1)$, $(3,4)$, $(4,0)$ mod 7

$q(x) = a_2x^2 + a_1x + a_0$, $e(x) = x + b_0$

Eq 1: $q(1) = r_1e(1)$, so $a_2 + a_1 + a_0 = 3(1 + b_0)$

Eq 2: $q(2) = r_2e(2)$, so $4a_2 + 2a_1 + a_0 = 1(2 + b_0)$

Eq 3: $q(3) = r_3e(3)$, so $9a_2 + 3a_1 + a_0 = 4(3 + b_0)$

Berlekamp-Welch: Example

Want to send length 2 message, have 1 corruption
Receive messages $(1,3)$, $(2,1)$, $(3,4)$, $(4,0)$ mod 7

$q(x) = a_2x^2 + a_1x + a_0$, $e(x) = x + b_0$

Eq 1: $q(1) = r_1e(1)$, so $a_2 + a_1 + a_0 = 3(1 + b_0)$

Eq 2: $q(2) = r_2e(2)$, so $4a_2 + 2a_1 + a_0 = 1(2 + b_0)$

Eq 3: $q(3) = r_3e(3)$, so $9a_2 + 3a_1 + a_0 = 4(3 + b_0)$

Eq 4: $q(4) = r_4e(4)$, so $16a_2 + 4a_1 + a_0 = 0(4 + b_0)$

Berlekamp-Welch: Example

Want to send length 2 message, have 1 corruption
Receive messages $(1,3)$, $(2,1)$, $(3,4)$, $(4,0)$ mod 7

$q(x) = a_2x^2 + a_1x + a_0$, $e(x) = x + b_0$

Eq 1: $q(1) = r_1e(1)$, so $a_2 + a_1 + a_0 = 3(1 + b_0)$

Eq 2: $q(2) = r_2e(2)$, so $4a_2 + 2a_1 + a_0 = 1(2 + b_0)$

Eq 3: $q(3) = r_3e(3)$, so $9a_2 + 3a_1 + a_0 = 4(3 + b_0)$

Eq 4: $q(4) = r_4e(4)$, so $16a_2 + 4a_1 + a_0 = 0(4 + b_0)$

Note: all eqns modulo 7, so can shrink some nums

(Berlekamp-Welch: Example): Continued

Simplify equations mod 7, move all variables to left:

$a_2 + a_1 + a_0 - 3b_0 = 3$

$4a_2 + 2a_1 + a_0 - b_0 = 2$

$2a_2 + 3a_1 + a_0 - 4b_0 = 5$

$2a_2 + 4a_1 + a_0 = 0$

(Berlekamp-Welch: Example): Continued

Simplify equations mod 7, move all variables to left:

$a_2 + a_1 + a_0 - 3b_0 = 3$
$4a_2 + 2a_1 + a_0 - b_0 = 2$
$2a_2 + 3a_1 + a_0 - 4b_0 = 5$
$2a_2 + 4a_1 + a_0 = 0$

Can use Gaussian Elimination (mod 7) to solve

(Berlekamp-Welch: Example): Continued

Simplify equations mod 7, move all variables to left:

$a_2 + a_1 + a_0 - 3b_0 = 3$
$4a_2 + 2a_1 + a_0 - b_0 = 2$
$2a_2 + 3a_1 + a_0 - 4b_0 = 5$
$2a_2 + 4a_1 + a_0 = 0$

Can use Gaussian Elimination (mod 7) to solve

Here, $a_2 = 3$, $a_1 = 6$, $a_0 = 5$, $b_0 = 6$
So $q(x) = 3x^2 + 6x + 5$, $e(x) = x + 6$

(Berlekamp-Welch: Example): Continued

Simplify equations mod 7, move all variables to left:

$a_2 + a_1 + a_0 - 3b_0 = 3$
$4a_2 + 2a_1 + a_0 - b_0 = 2$
$2a_2 + 3a_1 + a_0 - 4b_0 = 5$
$2a_2 + 4a_1 + a_0 = 0$

Can use Gaussian Elimination (mod 7) to solve

Here, $a_2 = 3$, $a_1 = 6$, $a_0 = 5$, $b_0 = 6$
So $q(x) = 3x^2 + 6x + 5$, $e(x) = x + 6$

Do poly long division mod 7 to get $p(x) = 3x + 2$
Original messages: $p(1) = 5$, $p(2) = 1$

Fin

Next time: countability!