Lecture 18: Applications of Factoring in Coding Theory

Rafael Oliveira

University of Waterloo
Cheriton School of Computer Science

rafael.oliveira.teaching@gmail.com

March 22, 2021

Overview

- Introduction to Coding Theory
- Reed-Solomon Codes
- List Decoding of Reed-Solomon Codes
- Conclusion
- Acknowledgements

Why codes?

- Hamming in 1950:

  We need to deal with *errors* that may occur when *storing digital information* on disk

Why codes?

- Hamming in 1950:

  We need to deal with *errors* that may occur when *storing digital information* on disk

- also

  Errors can occur when *transmitting* a message through *noisy channel*

$$\boxed{\text{duck}} \xrightarrow[\text{channel}]{\text{noisy}} \text{puck}$$

Why codes?

- Hamming in 1950:

  We need to deal with *errors* that may occur when *storing digital information* on disk

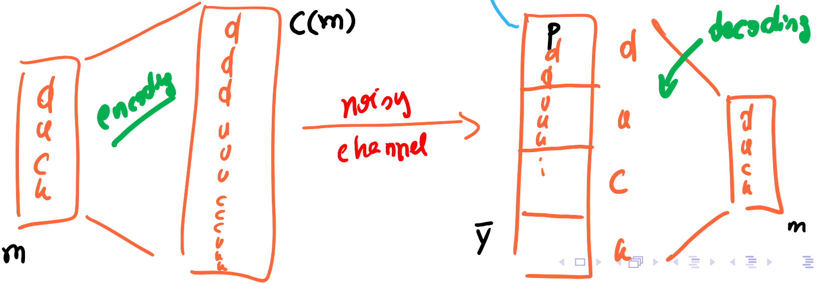
- also

  Errors can occur when *transmitting* a message through *noisy channel*

- Simple solution:

  (corrupted) codeword

  Store each bit three times. If one error occurs, we can still recover!



Why codes?

- Hamming in 1950:

  We need to deal with *errors* that may occur when *storing digital information* on disk

- also

  Errors can occur when *transmitting* a message through *noisy channel*

- Simple solution:

  Store each bit three times. If one error occurs, we can still recover!

- These are repetition codes. Not very efficient

add ~~too~~ much redundancy

Why codes?

- Hamming in 1950:

  We need to deal with *errors* that may occur when *storing digital information* on disk

- also

  Errors can occur when *transmitting* a message through *noisy channel*

- Simple solution:

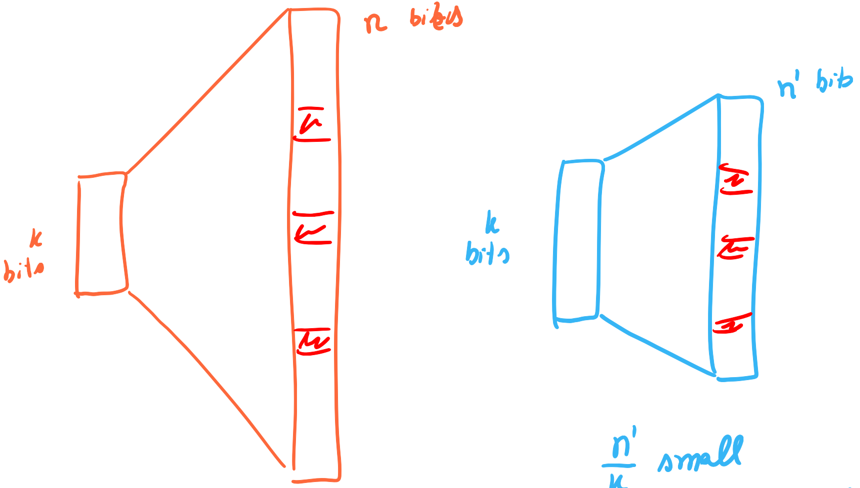
  Store each bit three times. If one error occurs, we can still recover!

- These are repetition codes. Not very efficient
- Can we do better?

  YES!

- What does better mean?

Efficiency in terms of redundancy



$\frac{n}{k}$ large ← too much redundancy

$\frac{n'}{k}$ small (not too much redundancy)

Error-Correcting Codes

- Let $\mathbb{F}$ be a field. Given two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{F}^n$, their *Hamming distance*

$$\Delta(\mathbf{u}, \mathbf{v}) = |\{i \in [n] \mid \mathbf{x}_i \neq \mathbf{y}_i\}|$$

Number of coordinates that they differ.

$u = (1, 2, 3, 1, 1)$ $v = (2, 1, 3, 1, 2)$

$\Delta(u, v) = 1 + 1 + 0 + 0 + 1 = 3$

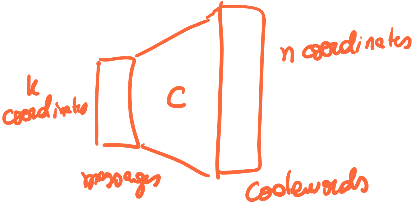
Error-Correcting Codes

- Let $\mathbb{F}$ be a field. Given two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{F}^n$, their *Hamming distance*

$$\Delta(\mathbf{u}, \mathbf{v}) = |\{i \in [n] \mid \mathbf{x}_i \neq \mathbf{y}_i\}|$$

  Number of coordinates that they differ.

- An *Error-Correcting Code* is a map $C : \mathbb{F}^k \to \mathbb{F}^n$



Error-Correcting Codes

- Let $\mathbb{F}$ be a field. Given two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{F}^n$, their *Hamming distance*

$$\Delta(\mathbf{u}, \mathbf{v}) = |\{i \in [n] \mid \mathbf{x}_i \neq \mathbf{y}_i\}|$$

  Number of coordinates that they differ.

- An *Error-Correcting Code* is a map $C : \mathbb{F}^k \to \mathbb{F}^n$
- Minimum distance of C:

$$\Delta(C) = \min_{\mathbf{x} \neq \mathbf{y} \in \cancel{C}\ \mathbb{F}^k} \cancel{\Delta(\mathbf{x}, \mathbf{y})}\ \Delta(C(x), C(y))$$

$$C : \mathbb{F}^3 \to \mathbb{F}^9$$

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \to \begin{pmatrix} a \\ a \\ a \\ b \\ b \\ b \\ c \\ c \\ c \end{pmatrix}$$

$$\boxed{\Delta(C) = 3}$$

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \quad \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}$$

$$a \neq \alpha$$

$$\begin{matrix} a & \alpha \\ a & \alpha \\ a & \alpha \end{matrix}$$

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \quad \begin{pmatrix} \alpha \\ b \\ c \end{pmatrix}$$

Error-Correcting Codes

- Let $\mathbb{F}$ be a field. Given two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{F}^n$, their *Hamming distance*

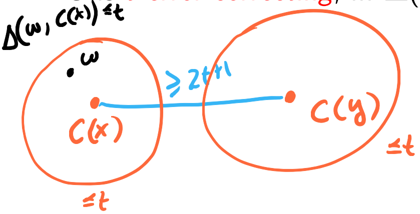
$$\Delta(\mathbf{u}, \mathbf{v}) = |\{i \in [n] \mid \mathbf{x}_i \neq \mathbf{y}_i\}|$$

  Number of coordinates that they differ.

- An *Error-Correcting Code* is a map $C : \mathbb{F}^k \to \mathbb{F}^n$
- Minimum distance of C:

$$\Delta(C) = \min_{\mathbf{x} \neq \mathbf{y} \in C} \Delta(\mathbf{x}, \mathbf{y})$$

- C is *t-error correcting*, iff $\Delta(C) \geq 2t + 1$



if $\omega \in B_t(C(x)) \cap B_t(C(y))$

$C(y) \xrightarrow{t} \omega \xrightarrow{t} C(x)$

$C(y) \xrightarrow{\leq 2t} C(x)$

Error-Correcting Codes

- Let $\mathbb{F}$ be a field. Given two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{F}^n$, their *Hamming distance*

$$\Delta(\mathbf{u}, \mathbf{v}) = |\{i \in [n] \mid \mathbf{x}_i \neq \mathbf{y}_i\}|$$

  Number of coordinates that they differ.
- An *Error-Correcting Code* is a map $C : \mathbb{F}^k \to \mathbb{F}^n$
- Minimum distance of C:

$$\Delta(C) = \min_{\mathbf{x} \neq \mathbf{y} \in C} \Delta(\mathbf{x}, \mathbf{y})$$

- C is *t-error correcting*, iff $\Delta(C) \geq 2t + 1$
- Key parameters:
  1. codeword length: n
  2. relative distance: $\delta := \Delta(C)/n$ fraction of errors
  3. rate: n/k redundancy of your code
  4. alphabet size: $|\mathbb{F}|$

constant

symbols we use per coordinate

Example: Repetition Code

Example: Hamming Code (Linear Code)

- $C : \mathbb{F}_2^4 \to \mathbb{F}_2^7$

$x \mapsto C \cdot x$

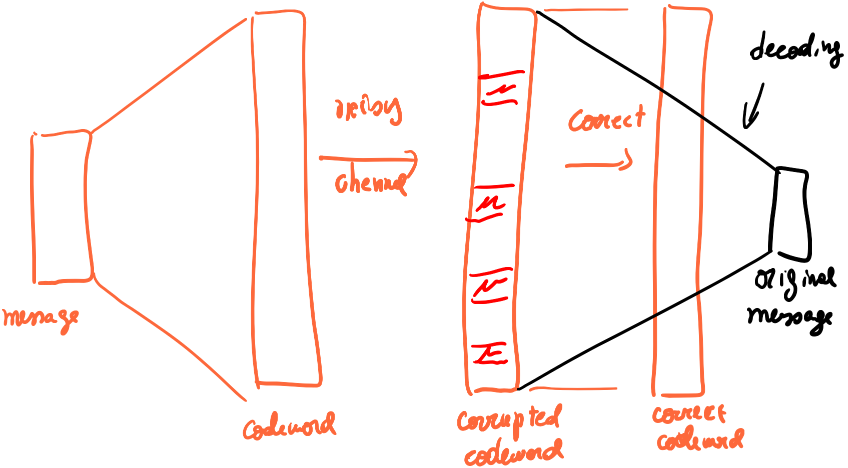
$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

$$C(x) = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_2 + x_3 + x_4 \\ x_1 + x_3 + x_4 \\ x_1 + x_2 + x_4 \end{pmatrix}$$

$\Delta(C) = 3 \Rightarrow C$ is 1-error correcting

C is more efficient than repetition codes!

Correction vs Decoding



Reed-Solomon Codes

- $\mathbb{F}_q$ be a finite field with q elements, $S \subset \mathbb{F}_q$ of size n, say $S = \{\alpha_1, \ldots, \alpha_n\}$ and $d < n$.
- Reed-Solomon Code:

$$RS_{S,d} : \mathbb{F}_q^{d+1} \to \mathbb{F}_q^n$$

given by $(p_0, \ldots, p_d) \leftrightarrow p(x) = p_0 + p_1 x + \cdots + p_d x^d$, then

$$RS_{S,d}(\underbrace{p_0, \ldots, p_d}_{p}) = (\underbrace{p(\alpha_1), \ldots, p(\alpha_n)}_{c(p)})$$

$$\underbrace{\begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_d \end{pmatrix}}_{\text{polynomial of degree } d} \in \mathbb{F}_q^{d+1}$$

$RS_{S,d}$: univariate polynomials of degree d $\rightarrow$ evaluations of the polynomial at S

Reed-Solomon Codes

- $\mathbb{F}_q$ be a finite field with q elements, $S \subset \mathbb{F}_q$ of size n, say $S = \{\alpha_1, \ldots, \alpha_n\}$ and $d < n$.
- Reed-Solomon Code:

$$RS_{S,d} : \mathbb{F}_q^{d+1} \to \mathbb{F}_q^n$$

given by $(p_0, \ldots, p_d) \leftrightarrow p(x) = p_0 + p_1 x + \cdots + p_d x^d$, then

$$RS_{S,d}(p_0, \ldots, p_d) = (p(\alpha_1), \ldots, p(\alpha_n))$$

- Evaluate the polynomial of degree d corresponding to $(p_0, p_1, \ldots, p_d)$ on the points of S

Reed-Solomon Codes

- $\mathbb{F}_q$ be a finite field with q elements, $S \subset \mathbb{F}_q$ of size n, say $S = \{\alpha_1, \ldots, \alpha_n\}$ and $d < n$.
- Reed-Solomon Code:

$$RS_{S,d} : \mathbb{F}_q^{d+1} \to \mathbb{F}_q^n$$

given by $(p_0, \ldots, p_d) \leftrightarrow p(x) = p_0 + p_1 x + \cdots + p_d x^d$, then

$$RS_{S,d}(p_0, \ldots, p_d) = (p(\alpha_1), \ldots, p(\alpha_n))$$

- Evaluate the polynomial of degree d corresponding to $(p_0, p_1, \ldots, p_d)$ on the points of S
- Properties of Reed-Solomon code:
  1. linear code
  2. alphabet size: q
  3. rate: $n/(d+1)$
  4. distance: $n - d$

Distance of Reed-Solomon Codes

$$\Delta(RS_{S,d}) = \min_{\substack{p,q \in \mathbb{F}[x]_{\leq d} \\ p \neq q}} \underbrace{\Delta(C(p), C(q))}_{\substack{\#\text{ of elements } \alpha_i \in S \\ \text{s.t. } p(\alpha_i) \neq q(\alpha_i)}}$$

If $p \neq q \Rightarrow p-q$ not zero polynomial

$\deg(p-q) \leq d$

$\Rightarrow p-q$ has at most d zeros in $\mathbb{F}_q$ / S

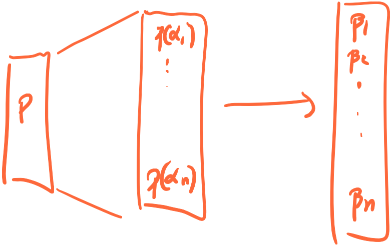
$\Rightarrow p(\alpha_i) = q(\alpha_i)$ in at most d values α_i

$\Rightarrow p(\alpha_j) \neq q(\alpha_j)$ for at least $n-d$ values of α_j

Decoding Reed-Solomon Codes

errors: $\underbrace{e < \frac{n-d}{2}}_{\text{half the distance.}}$

- Locating the errors



received

$(\beta_1, \beta_2, \ldots, \beta_n)$

errors are in set $B \subset [n]$

$|B| = e < \frac{n-d}{2}$

Error polynomial $E(x)$: if $\beta_i \neq p(\alpha_i)$ set $E(\alpha_i) = 0$

$E(x) = \prod_{i \in B} (x - \alpha_i)$

$$\underbrace{\deg(E) = |B| < \frac{n-d}{2}}_{\text{small degree}}$$

$$\underbrace{E(x)}_{< \frac{n-d}{2}} \cdot \underbrace{p(x)}_{\leq d} \quad \text{" low degree "}$$

$$E(\alpha_i)\,p(\alpha_i) = \begin{cases} 0 & \text{if } p(\alpha_i) \neq \beta_i \\ \underbrace{E(\alpha_i)\,p(\alpha_i)}_{E(\alpha_i)\beta_i} & \text{if } p(\alpha_i) = \beta_i \end{cases}$$

if we know $E(x)$ we can definitely decode p

$$\underbrace{E(\alpha_i) \neq 0}_{\geq n - \frac{n-d}{2} = \frac{n+d}{2} > d} \Rightarrow \boxed{p(\alpha_i) = \beta_i}$$

correct value for at least $d+1$ evaluations!

$\Rightarrow$ can interpolate p !

Decoding Reed-Solomon Codes

- Finding error detecting polynomial

$\underbrace{E(x)}_{e} \cdot \underbrace{p(x)}_{d}$

$\boxed{N(x)}$ $\deg(N) \leq d+e$ $\boxed{E(x)}$ $\deg(E) = e$

$\boxed{N(\alpha_i) = \beta_i \cdot E(\alpha_i) \quad \text{for all } i \in [n]}$

linear system of equations!

How can we find N, E ?

$$N(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{d+e} x^{d+e}$$

$$E(x) = b_0 + b_1 x + \cdots + b_e x^e$$

variables: $a_0, \ldots, a_{d+e}, b_0, \ldots, b_e$

$$\boxed{N(\alpha_i) = \beta_i \cdot E(\alpha_i)}$$ ← one linear equation per (α_i, β_i)

have n equations $(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)$

Q: does the system above have non-trivial solution?

if $< \frac{n-d}{2}$ errors $\boxed{N(x) = E(x)\, p(x)}$ non-trivial solution!

Berlekamp-Welch Algorithm

- **Input:** evaluations $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree d, error parameter $e < \dfrac{n-d-1}{2}$
- **Output:** message p (our polynomial of degree d), or not e-close to any codeword

Berlekamp-Welch Algorithm

- **Input:** evaluations $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree d, error parameter $e < \dfrac{n-d-1}{2}$
- **Output:** message p (our polynomial of degree d), or not e-close to any codeword
- Find non-zero polynomials $N(x), E(x) \in \mathbb{F}_q[x]$ such that
  1. $\deg(N) \leq d + e$
  2. $\deg(E) \leq e$
  3. $N(\alpha_i) = \beta_i E(\alpha_i)$ for $i \in [n]$

Berlekamp-Welch Algorithm

- **Input:** evaluations $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree d, error parameter $e < \dfrac{n-d-1}{2}$
- **Output:** message p (our polynomial of degree d), or <u>not e-close to any codeword</u>
- Find non-zero polynomials $N(x), E(x) \in \mathbb{F}_q[x]$ such that
  1. $\deg(N) \leq d + e$
  2. $\deg(E) \leq e$
  3. $N(\alpha_i) = \beta_i E(\alpha_i)$ for $i \in [n]$
- If E divides N, output

$$p(x) = \frac{N(x)}{E(x)}$$

else, output <u>not e-close to any codeword</u>

Uniqueness of Solution

$N_1(x), E_1(x)$ $\quad$ $N_2(x), E_2(x)$ $\quad$ non-trivial solutions $(E_1, E_2 \neq 0)$

WTS: $\dfrac{N_1}{E_1} = \dfrac{N_2}{E_2}$

Proof: $\left.\begin{array}{l} N_1(\alpha_i) = \beta_i E_1(\alpha_i) \\ \beta_i E_2(\alpha_i) = N_2(\alpha_i) \end{array}\right\} \Rightarrow \beta_i N_1(\alpha_i) E_2(\alpha_i) =$

$= \beta_i E_1(\alpha_i) N_2(\alpha_i)$

$\Rightarrow$ for n values of α_i we have

$N_1(\alpha_i) E_2(\alpha_i) = E_1(\alpha_i) N_2(\alpha_i)$

$\deg(N_1(x)E_2(x)) \leq d + 2e < n > d + 2e \geq \deg(N_2(x)E_1(x))$

$\Rightarrow N_1 E_2 = N_2 E_1$

$\Rightarrow \dfrac{N_1}{E_1} = \dfrac{N_2}{E_2}$

Uniqueness of Solution

- Introduction to Coding Theory
- Reed-Solomon Codes
- List Decoding of Reed-Solomon Codes
- Conclusion
- Acknowledgements

What is list decoding?

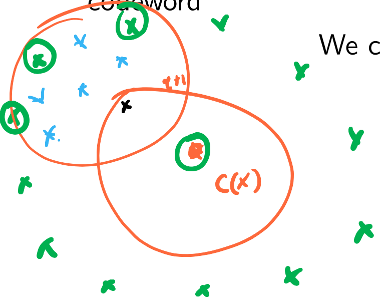
- Given a code $C : \mathbb{F}_q^k \to \mathbb{F}_q^n$ with distance $\Delta(C) = 2t + 1$, we can correct a (corrupted) codeword up to t errors.

What is list decoding?

- Given a code $C : \mathbb{F}_q^k \to \mathbb{F}_q^n$ with distance $\Delta(C) = 2t + 1$, we can correct a (corrupted) codeword up to t errors.
- The above is *worst-case* behaviour

What is list decoding?

- Given a code $C : \mathbb{F}_q^k \to \mathbb{F}_q^n$ with distance $\Delta(C) = 2t + 1$, we can correct a (corrupted) codeword up to t errors.
- The above is *worst-case* behaviour
- What happens if the message has more errors?
- Even in worst-case, if we have a bit more errors than t, often we can prove there exists *small list* of codewords which are close to corrupted codeword



We can show this in *worst-case*!

blue doesn't happen!

What is list decoding?

- Given a code $C : \mathbb{F}_q^k \to \mathbb{F}_q^n$ with distance $\Delta(C) = 2t + 1$, we can correct a (corrupted) codeword up to t errors.
- The above is *worst-case* behaviour
- What happens if the message has more errors?
- Even in worst-case, if we have a bit more errors than t, often we can prove there exists *small list* of codewords which are close to corrupted codeword

We can show this in *worst-case*!

- This is purpose of list decoding.

duck $\longrightarrow$ {duck, puck, tuck}

What is list decoding?

- Given a code $C : \mathbb{F}_q^k \to \mathbb{F}_q^n$ with distance $\Delta(C) = 2t + 1$, we can correct a (corrupted) codeword up to t errors.
- The above is *worst-case* behaviour
- What happens if the message has more errors?
- Even in worst-case, if we have a bit more errors than t, often we can prove there exists *small list* of codewords which are close to corrupted codeword

We can show this in *worst-case*!

- This is purpose of list decoding.
- List decoding allows us to tolerate a lot more errors!

List Decoding of Reed-Solomon Codes

- $\mathbb{F}_q$ be a finite field with q elements, $S \subset \mathbb{F}_q$ of size n, say $S = \{\alpha_1, \ldots, \alpha_n\}$ and $d < n$.
- Reed-Solomon Code:

$$RS_{S,d} : \mathbb{F}_q^{d+1} \to \mathbb{F}_q^n$$

  given by $(p_0, \ldots, p_d) \leftrightarrow p(x) = p_0 + p_1 x + \cdots + p_d x^d$, then

$$RS_{S,d}(p_0, \ldots, p_d) = (p(\alpha_1), \ldots, p(\alpha_n))$$

- Properties of Reed-Solomon code:
  1. linear code
  2. alphabet size: q
  3. rate: $n/(d+1)$
  4. distance: $n - d$

List Decoding of Reed-Solomon Codes

- $\mathbb{F}_q$ be a finite field with q elements, $S \subset \mathbb{F}_q$ of size n, say $S = \{\alpha_1, \ldots, \alpha_n\}$ and $d < n$.
- Reed-Solomon Code:

$$RS_{S,d} : \mathbb{F}_q^{d+1} \to \mathbb{F}_q^n$$

given by $(p_0, \ldots, p_d) \leftrightarrow p(x) = p_0 + p_1 x + \cdots + p_d x^d$, then

$$RS_{S,d}(p_0, \ldots, p_d) = (p(\alpha_1), \ldots, p(\alpha_n))$$

- Properties of Reed-Solomon code:
  1. linear code
  2. alphabet size: q
  3. rate: $n/(d+1)$
  4. distance: $n - d$

unique decoding

- we will now see how to *list decode* more than $\dfrac{n-d}{2}$ errors!

Warm-up Problem 1: mixture of two codewords

- Let $p(x), q(x) \in \mathbb{F}_q[x]$ be two polynomials of degree $d < n/2$
- Suppose we are given $(\alpha_i, \{p(\alpha_i), q(\alpha_i)\})$, where $i \in [n]$

Note that we *do not know order*!

- Can we recover p, q?

$$\{\beta_i, \gamma_i\} = \{p(\alpha_i), q(\alpha_i)\}$$

Warm-up Problem 1: mixture of two codewords

- Let $p(x), q(x) \in \mathbb{F}_q[x]$ be two polynomials of degree $d < n/2$
- Suppose we are given $(\alpha_i, \{p(\alpha_i), q(\alpha_i)\})$, where $i \in [n]$

  Note that we *do not know order*!

- Can we recover p, q?
- Idea: we can compute $p(\alpha_i) + q(\alpha_i)$ and $p(\alpha_i) \cdot q(\alpha_i)$, thus can interpolate the polynomials:

$$p(x)q(x), \quad \text{and} \quad p(x) + q(x)$$

Warm-up Problem 1: mixture of two codewords

- Let $p(x), q(x) \in \mathbb{F}_q[x]$ be two polynomials of degree $d < n/2$
- Suppose we are given $(\alpha_i, \{p(\alpha_i), q(\alpha_i)\})$, where $i \in [n]$

  Note that we *do not know order*!

- Can we recover p, q?
- Idea: we can compute $p(\alpha_i) + q(\alpha_i)$ and $p(\alpha_i) + q(\alpha_i)$, thus can interpolate the polynomials:

$$p(x)q(x), \quad \text{and} \quad p(x) + q(x)$$

- Once we find these polynomials, can now factor

$$S(x, y) = y^2 + \underline{(p(x) + q(x))}y + \underline{p(x)q(x)}$$

$$= (y + p(x))(y + q(x))$$

Warm-up Problem 2: mixture of two codewords

- Let $p(x), q(x) \in \mathbb{F}_q[x]$ be two polynomials of degree $d < n/6$
- Suppose for each α_i we are given (α_i, β_i), where $\beta_i \in \{p(\alpha_i), q(\alpha_i)\}$

  Note that for each α_i we know *only one* of $\{p(\alpha_i), q(\alpha_i)\}$!
- Can we recover p, q?

Warm-up Problem 2: mixture of two codewords

- Let $p(x), q(x) \in \mathbb{F}_q[x]$ be two polynomials of degree $d < n/6$
- Suppose for each α_i we are given (α_i, β_i), where $\beta_i \in \{p(\alpha_i), q(\alpha_i)\}$

  Note that for each α_i we know *only one* of $\{p(\alpha_i), q(\alpha_i)\}$!
- Can we recover p, q?
- if only given $p(\alpha_i)$, no way to recover $q(x)$, need extra assumption:

  Both polynomials are *well represented*: that is, at least $n/3$ of the evaluations are from p and at least $n/3$ are from q.

Warm-up Problem 2: mixture of two codewords

- Let $p(x), q(x) \in \mathbb{F}_q[x]$ be two polynomials of degree $d < n/6$
- Suppose for each α_i we are given (α_i, β_i), where $\beta_i \in \{p(\alpha_i), q(\alpha_i)\}$

  Note that for each α_i we know *only one* of $\{p(\alpha_i), q(\alpha_i)\}$!
- Can we recover p, q?
- if only given $p(\alpha_i)$, no way to recover $q(x)$, need extra assumption:

  Both polynomials are *well represented*: that is, at least $n/3$ of the evaluations are from p and at least $n/3$ are from q.
- Idea: let

$$S(x, y) = (y - p(x))(y - q(x))$$

  Then for each (α_i, β_i), we know that $S(\alpha_i, \beta_i) = 0$.

  And we know that S has *low degree*!

Warm-up Problem 2: mixture of two codewords

- Let $p(x), q(x) \in \mathbb{F}_q[x]$ be two polynomials of degree $d < n/6$
- Suppose for each α_i we are given (α_i, β_i), where $\beta_i \in \{p(\alpha_i), q(\alpha_i)\}$

  Note that for each α_i we know *only one* of $\{p(\alpha_i), q(\alpha_i)\}$!
- Can we recover p, q?
- if only given $p(\alpha_i)$, no way to recover $q(x)$, need extra assumption:

  Both polynomials are *well represented*: that is, at least $n/3$ of the evaluations are from p and at least $n/3$ are from q.
- Idea: let

$$S(x, y) = (y - p(x))(y - q(x))$$

  Then for each (α_i, β_i), we know that $S(\alpha_i, \beta_i) = 0$.

  And we know that S has *low degree*!
- Now, find a *bivariate polynomial* $R(x, y)$ which is of the form:

$$R(x, y) = y^2 + R_1(x)y + R_2(x)$$

  satisfying: $\deg(R_1) \leq d$, $\deg(R_2) \leq 2d$ and $R(\alpha_i, \beta_i) = 0$ for $i \in [n]$.

Warm-up Problem 2: mixture of two codewords

- Let $p(x), q(x) \in \mathbb{F}_q[x]$ be two polynomials of degree $d < n/6$
- Suppose for each α_i we are given (α_i, β_i), where $\beta_i \in \{p(\alpha_i), q(\alpha_i)\}$
- Found a *bivariate polynomial* $R(x, y)$ which is of the form:

$$R(x, y) = y^2 + R_1(x)y + R_2(x)$$

satisfying: $\deg(R_1) \leq d$, $\deg(R_2) \leq 2d$ and $R(\alpha_i, \beta_i) = 0$ for $i \in [n]$.

Warm-up Problem 2: mixture of two codewords

- Let $p(x), q(x) \in \mathbb{F}_q[x]$ be two polynomials of degree $d < n/6$
- Suppose for each α_i we are given (α_i, β_i), where $\beta_i \in \{p(\alpha_i), q(\alpha_i)\}$
- Found a *bivariate polynomial* $R(x, y)$ which is of the form:

$$R(x, y) = y^2 + R_1(x)y + R_2(x)$$

  satisfying: $\deg(R_1) \leq d$, $\deg(R_2) \leq 2d$ and $R(\alpha_i, \beta_i) = 0$ for $i \in [n]$.
- What is it good for?

Warm-up Problem 2: mixture of two codewords

- Let $p(x), q(x) \in \mathbb{F}_q[x]$ be two polynomials of degree $d < n/6$
- Suppose for each α_i we are given (α_i, β_i), where $\beta_i \in \{p(\alpha_i), q(\alpha_i)\}$
- Found a *bivariate polynomial* $R(x, y)$ which is of the form:

$$R(x, y) = y^2 + R_1(x)y + R_2(x)$$

  satisfying: $\deg(R_1) \leq d$, $\deg(R_2) \leq 2d$ and $R(\alpha_i, \beta_i) = 0$ for $i \in [n]$.
- What is it good for?
- If $f(x)$ of degree $< n/6$ is such that $R(\alpha_i, f(\alpha_i)) = 0$ for at least $n/3$ values of α_i, then

$$y - f(x) \text{ divides } R(x, y)$$

Warm-up Problem 2: mixture of two codewords

- Let $p(x), q(x) \in \mathbb{F}_q[x]$ be two polynomials of degree $d < n/6$
- Suppose for each α_i we are given (α_i, β_i), where $\beta_i \in \{p(\alpha_i), q(\alpha_i)\}$
- Found a *bivariate polynomial* $R(x, y)$ which is of the form:

$$R(x, y) = y^2 + R_1(x)y + R_2(x)$$

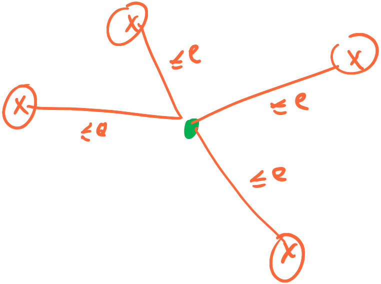
  satisfying: $\deg(R_1) \leq d$, $\deg(R_2) \leq 2d$ and $R(\alpha_i, \beta_i) = 0$ for $i \in [n]$.
- What is it good for?
- If $f(x)$ of degree $< n/6$ is such that $R(\alpha_i, f(\alpha_i)) = 0$ for at least $n/3$ values of α_i, then

$$y - f(x) \text{ divides } R(x, y)$$

- Previous lemma implies that $R(x, y) = (y - p(x))(y - q(x))$!

List Decoding of Reed-Solomon Codes

- **Input:** list of pairs $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree parameter d, and error parameter e for the Reed-Solomon code.
- **Output:** a list of *all* polynomials $p(x) \in \mathbb{F}_q[x]$ of degree $\leq d$ such that $p(\alpha_i) = \beta_i$ for *at least* $\boxed{n - e}$ values of $i \in [n]$



agree in at least
n-e values

List Decoding of Reed-Solomon Codes

- **Input:** list of pairs $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree parameter d, and error parameter e for the Reed-Solomon code.
- **Output:** a list of *all* polynomials $p(x) \in \mathbb{F}_q[x]$ of degree $\leq d$ such that $p(\alpha_i) = \beta_i$ for *at least* $n - e$ values of $i \in [n]$
- Let $t = n - e$ be our *agreement parameter*

List Decoding of Reed-Solomon Codes

- **Input:** list of pairs $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree parameter d, and error parameter e for the Reed-Solomon code.
- **Output:** a list of *all* polynomials $p(x) \in \mathbb{F}_q[x]$ of degree $\leq d$ such that $p(\alpha_i) = \beta_i$ for *at least* $n - e$ values of $i \in [n]$
- Let $t = n - e$ be our *agreement parameter*
- For this generalization, we need a generalized notion of degree:

  A polynomial $S(x, y)$ has *$(1, d)$-degree* D iff for each monomial $x^a y^b$ of $S(x, y)$ we have that $a + db \leq D$.

$x^a y^b \mapsto a + db$

x has degree 1
y has degree d

List Decoding of Reed-Solomon Codes

- **Input:** list of pairs $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree parameter d, and error parameter e for the Reed-Solomon code.
- **Output:** a list of *all* polynomials $p(x) \in \mathbb{F}_q[x]$ of degree $\leq d$ such that $p(\alpha_i) = \beta_i$ for *at least* $n - e$ values of $i \in [n]$
- Let $t = n - e$ be our *agreement parameter*
- For this generalization, we need a generalized notion of degree:

  A polynomial $S(x, y)$ has *$(1, d)$-degree* D iff for each monomial $x^a y^b$ of $S(x, y)$ we have that $a + db \leq D$.

- If $S(x, y)$ has $(1, d)$-degree D and $p(x)$ is such that
  1. $\deg(p) \leq d$
  2. $S(\alpha_i, p(\alpha_i)) = 0$ for at least $D + 1$ values of α_i

  then $y - p(x)$ divides $S(x, y)$

$\deg(S(x, p(x))) \leq D$

univariate

$S(x,y) = (y - p(x))\, Q(x,y) + \underbrace{R(x)}_{= S(x,\, p(x))}$

$R(\alpha_i) = 0$ for $\geq D+1$ α_i's

$\Rightarrow R(x) = 0$

List Decoding of Reed-Solomon Codes

- **Input:** list of pairs $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree parameter d, and error parameter e for the Reed-Solomon code.
- **Output:** a list of *all* polynomials $p(x) \in \mathbb{F}_q[x]$ of degree $\leq d$ such that $p(\alpha_i) = \beta_i$ for *at least* $n - e$ values of $i \in [n]$
- Let $t = n - e$ be our *agreement parameter*
- For this generalization, we need a generalized notion of degree:

  A polynomial $S(x, y)$ has *$(1, d)$-degree* D iff for each monomial $x^a y^b$ of $S(x, y)$ we have that $a + db \leq D$.
- If $S(x, y)$ has $(1, d)$-degree D and $p(x)$ is such that
  1. $\deg(p) \leq d$
  2. $S(\alpha_i, p(\alpha_i)) = 0$ for at least $D + 1$ values of α_i

  then $y - p(x)$ divides $S(x, y)$
- So, if we could interpolate a polynomial $S(x, y)$ of $(1, d)$-degree $< t$ such that $S(\alpha_i, \beta_i) = 0$ we would be done!

List Decoding of Reed-Solomon Codes

- **Input:** list of pairs $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree parameter d, and error parameter e for the Reed-Solomon code.
- **Output:** a list of *all* polynomials $p(x) \in \mathbb{F}_q[x]$ of degree $\leq d$ such that $p(\alpha_i) = \beta_i$ for *at least* $n - e$ values of $i \in [n]$

List Decoding of Reed-Solomon Codes

- **Input:** list of pairs $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree parameter d, and error parameter e for the Reed-Solomon code.
- **Output:** a list of *all* polynomials $p(x) \in \mathbb{F}_q[x]$ of degree $\leq d$ such that $p(\alpha_i) = \beta_i$ for *at least* $n - e$ values of $i \in [n]$
- Let $t = n - e$ be our *agreement parameter*

List Decoding of Reed-Solomon Codes

- **Input:** list of pairs $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree parameter d, and error parameter e for the Reed-Solomon code.
- **Output:** a list of *all* polynomials $p(x) \in \mathbb{F}_q[x]$ of degree $\leq d$ such that $p(\alpha_i) = \beta_i$ for *at least* $n - e$ values of $i \in [n]$
- Let $t = n - e$ be our *agreement parameter*
- For this generalization, we need a generalized notion of degree:

  A polynomial $S(x, y)$ has *$(1, d)$-degree* D iff for each monomial $x^a y^b$ of $S(x, y)$ we have that $a + db \leq D$.

List Decoding of Reed-Solomon Codes

- **Input:** list of pairs $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree parameter d, and error parameter e for the Reed-Solomon code.
- **Output:** a list of *all* polynomials $p(x) \in \mathbb{F}_q[x]$ of degree $\leq d$ such that $p(\alpha_i) = \beta_i$ for *at least* $n - e$ values of $i \in [n]$
- Let $t = n - e$ be our *agreement parameter*
- For this generalization, we need a generalized notion of degree:

  A polynomial $S(x, y)$ has *$(1, d)$-degree* D iff for each monomial $x^a y^b$ of $S(x, y)$ we have that $a + db \leq D$.
- If $S(x, y)$ has $(1, d)$-degree D and $p(x)$ is such that
  1. $\deg(p) \leq d$
  2. $S(\alpha_i, p(\alpha_i)) = 0$ for at least $D + 1$ values of α_i

  then $y - p(x)$ divides $S(x, y)$

List Decoding of Reed-Solomon Codes

- **Input:** list of pairs $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree parameter d, and error parameter e for the Reed-Solomon code.
- **Output:** a list of *all* polynomials $p(x) \in \mathbb{F}_q[x]$ of degree $\leq d$ such that $p(\alpha_i) = \beta_i$ for *at least* $n - e$ values of $i \in [n]$
- Let $t = n - e$ be our *agreement parameter*
- For this generalization, we need a generalized notion of degree:

  A polynomial $S(x, y)$ has *$(1, d)$-degree* D iff for each monomial $x^a y^b$ of $S(x, y)$ we have that $a + db \leq D$.
- If $S(x, y)$ has $(1, d)$-degree D and $p(x)$ is such that
  1. $\deg(p) \leq d$
  2. $S(\alpha_i, p(\alpha_i)) = 0$ for at least $D + 1$ values of α_i

  then $y - p(x)$ divides $S(x, y)$
- So, if we could interpolate a polynomial $S(x, y)$ of $(1, d)$-degree $< t$ such that $S(\alpha_i, \beta_i) = 0$ we would be done!

List Decoding of Reed-Solomon Codes

- **Input:** list of pairs $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree parameter d, and error parameter e for the Reed-Solomon code.
- **Output:** a list of *all* polynomials $p(x) \in \mathbb{F}_q[x]$ of degree $\leq d$ such that $p(\alpha_i) = \beta_i$ for *at least* $n - e$ values of $i \in [n]$
- Let $t = n - e$ be our *agreement parameter*
- If we could interpolate a polynomial $S(x, y)$ of $(1, d)$-degree $< t$ such that $S(\alpha_i, \beta_i) = 0$ we would be done!

List Decoding of Reed-Solomon Codes

- **Input:** list of pairs $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree parameter d, and error parameter e for the Reed-Solomon code.
- **Output:** a list of *all* polynomials $p(x) \in \mathbb{F}_q[x]$ of degree $\leq d$ such that $p(\alpha_i) = \beta_i$ for *at least* $n - e$ values of $i \in [n]$
- Let $t = n - e$ be our *agreement parameter*
- If we could interpolate a polynomial $S(x, y)$ of $(1, d)$-degree $< t$ such that $S(\alpha_i, \beta_i) = 0$ we would be done!
- For which values of t can we do this?

List Decoding of Reed-Solomon Codes

- **Input:** list of pairs $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree parameter d, and error parameter e for the Reed-Solomon code.
- **Output:** a list of *all* polynomials $p(x) \in \mathbb{F}_q[x]$ of degree $\leq d$ such that $p(\alpha_i) = \beta_i$ for *at least* $n - e$ values of $i \in [n]$
- Let $t = n - e$ be our *agreement parameter*
- If we could interpolate a polynomial $S(x, y)$ of $(1, d)$-degree $< t$ such that $S(\alpha_i, \beta_i) = 0$ we would be done!
- For which values of t can we do this?
- $S(\alpha_i, \beta_i) = 0$ for all $i \in [n]$ is a *homogeneous linear system* of equations!

  Thus, if $S(x, y)$ has more than n monomials, we will have a non-trivial solution to homogeneous system!

$$S(x,y) = \sum_{\substack{i,j \\ i+dj<t}} a_{ij} x^i y^j$$

variables

$$\#\{(i,j) \mid i+dj<t\} > n$$

we are good!

List Decoding of Reed-Solomon Codes

- **Input:** list of pairs $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree parameter d, and error parameter e for the Reed-Solomon code.
- **Output:** a list of *all* polynomials $p(x) \in \mathbb{F}_q[x]$ of degree $\leq d$ such that $p(\alpha_i) = \beta_i$ for *at least* $n - e$ values of $i \in [n]$
- Let $t = n - e$ be our *agreement parameter*
- If we could interpolate a polynomial $S(x, y)$ of $(1, d)$-degree $< t$ such that $S(\alpha_i, \beta_i) = 0$ we would be done!
- For which values of t can we do this?
- $S(\alpha_i, \beta_i) = 0$ for all $i \in [n]$ is a *homogeneous linear system* of equations!
  Thus, if $S(x, y)$ has more than n monomials, we will have a non-trivial solution to homogeneous system!
- Let D be the $(1, d)$-degree of $S(x, y)$. So long as $\binom{D+2}{2} > dn$ we can find a non-zero solution! $\boxed{D = t-1}$

List Decoding of Reed-Solomon Codes

- $S(\alpha_i, \beta_i) = 0$ for all $i \in [n]$ is a *homogeneous linear system* of equations!
  Thus, if $S(x, y)$ has more than n monomials, we will have a non-trivial solution to homogeneous system!
- Let D be the $(1, d)$-degree of $S(x, y)$. So long as $\binom{D+2}{2} > dn$ we can find a non-zero solution!
- We need to prove that for such D, $S(x, y)$ will have $> n$ monomials.

Sudan's Algorithm

$\binom{t+2}{2} > n$

- **Input:** evaluations $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree d, agreement parameter $t > \sqrt{2dn}$
- **Output:** a list of $\boxed{\textit{all}}$ polynomials $p(x) \in \mathbb{F}_q[x]$ of degree $\leq d$ such that $p(\alpha_i) = \beta_i$ for *at least* t values of $i \in [n]$.

Sudan's Algorithm

- **Input:** evaluations $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree d, agreement parameter $t > \sqrt{2dn}$
- **Output:** a list of *all* polynomials $p(x) \in \mathbb{F}_q[x]$ of degree $\leq d$ such that $p(\alpha_i) = \beta_i$ for *at least* t values of $i \in [n]$.
- Find non-zero polynomial $S(x, y) \in \mathbb{F}_q[x, y]$ such that
  1. $(1, d)$-degree of $S(x, y)$ is $\sqrt{2dn}$
  2. $S(\alpha_i, \beta_i) = 0$ for $i \in [n]$

Sudan's Algorithm

- **Input:** evaluations $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree d, agreement parameter $t > \sqrt{2dn}$
- **Output:** a list of *all* polynomials $p(x) \in \mathbb{F}_q[x]$ of degree $\leq d$ such that $p(\alpha_i) = \beta_i$ for *at least* t values of $i \in [n]$.
- Find non-zero polynomial $S(x, y) \in \mathbb{F}_q[x, y]$ such that
  1. $(1, d)$-degree of $S(x, y)$ is $\sqrt{2dn}$
  2. $S(\alpha_i, \beta_i) = 0$ for $i \in [n]$
- Factor $S(x, y)$, and for each factor of the form $\boxed{y - p(x)}$ where $\deg(p) \leq d$, add $p(x)$ to our list.

Sudan's Algorithm

S can have at most $\sqrt{\frac{2n}{d}}$ such factors

- **Input:** evaluations $\{(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)\} \subset \mathbb{F}_q^2$, degree d, agreement parameter $t > \sqrt{2dn}$
- **Output:** a list of *all* polynomials $p(x) \in \mathbb{F}_q[x]$ of degree $\leq d$ such that $p(\alpha_i) = \beta_i$ for *at least* t values of $i \in [n]$.
- Find non-zero polynomial $S(x, y) \in \mathbb{F}_q[x, y]$ such that
  1. $(1, d)$-degree of $S(x, y)$ is $\sqrt{2dn}$
  2. $S(\alpha_i, \beta_i) = 0$ for $i \in [n]$

  $\deg_y = 1$
- Factor $S(x, y)$, and for each factor of the form $\boxed{y - p(x)}$ where $\deg(p) \leq d$, add $p(x)$ to our list.
- Return list

S $(1,d)$-degree $\leq \sqrt{2dn}$

$x^a y^b$ $a + db \leq \sqrt{2dn} \Rightarrow \boxed{b \leq \sqrt{\frac{2n}{d}}}$

Unique Decoding vs List Decoding of Reed-Solomon codes

- Unique decoding of Reed-Solomon codes:

$$\text{Error parameter: } \boxed{e < \frac{n-d}{2}}$$

- List decoding of Reed-Solomon codes:

$$\text{Error parameter: } e < n - \sqrt{2dn}$$

$$\text{List size: } \ell \leq \sqrt{2n/d}$$

Conclusion

- Today we learned about coding theory and how symbolic computation is key in decoding of Reed-Solomon codes (one of the most widely used codes in real life)

Acknowledgement

- Lecture based largely on:
  - Madhu's notes - lectures 7 and 8

    `http://people.csail.mit.edu/madhu/FT98/`
  - Madhu's 6.897 notes
  - Chapter 4 of Venkat's survey

    Algorithmic results in List Decoding