On Decoding Cohen-Haeupler-Schulman Tree Codes

Matthew Weidner[1]*

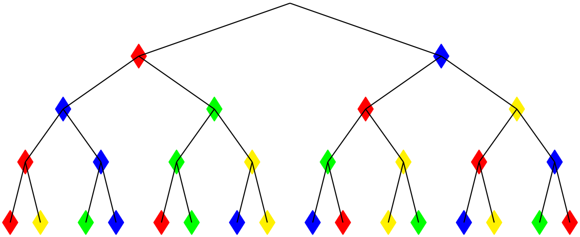
joint with Anand Kumar Narayanan[2]

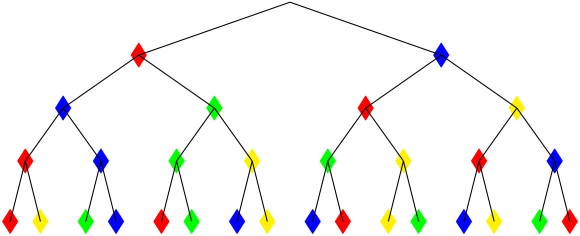
[1]Carnegie Mellon University, Pittsburgh, PA, USA

[2]Laboratoire d'informatique de Paris 6,
Sorbonne Université, Paris, France

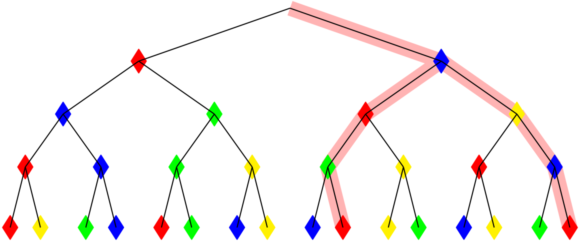
SODA 2020

*Supported by the NSF through a SIAM Travel Award

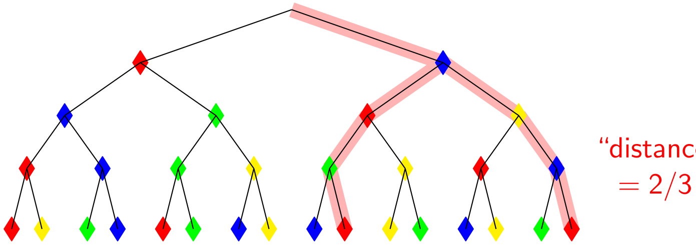




Goal: different branches have very different color sequences (after they diverge).

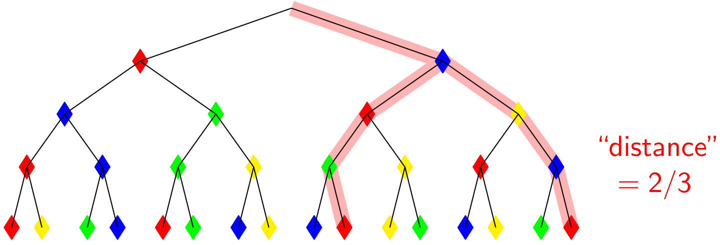


Goal: different branches have very different color sequences (after they diverge).



Goal: different branches have very different color sequences (after they diverge).

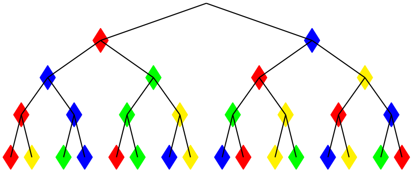
Tree Codes



Goal: different branches have very different color sequences (after they diverge).

Parameters:

- Length (depth)
- Alphabet size (number of colors/labels)
- Distance (minimum distance between two branches)

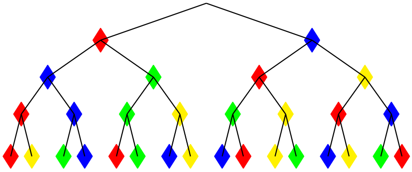


$\approx$

Online function

$\{0,1\}^{\leq n} \to \Sigma^{\leq n}$,

some alphabet Σ.



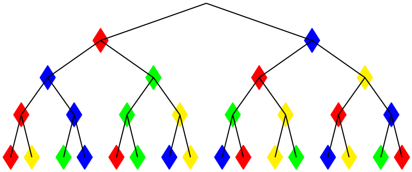
$\approx$

Online function

$\{0,1\}^{\leq n} \to \Sigma^{\leq n}$,

some alphabet Σ.

- Online analog of error-correcting codes



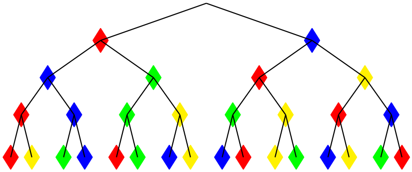
$\approx$

Online function

$\{0,1\}^{\leq n} \to \Sigma^{\leq n}$,

some alphabet Σ.

- Online analog of error-correcting codes
  - Can "decode" input from errored version of output with $< \frac{1}{2}$(distance) errors



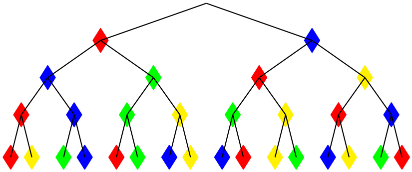
$\approx$

Online function

$\{0,1\}^{\leq n} \to \Sigma^{\leq n}$,

some alphabet Σ.

- Online analog of error-correcting codes
  - Can "decode" input from errored version of output with $< \frac{1}{2}$(distance) errors
- (Schulman 90s): Add error tolerance to *interactive communication protocols*



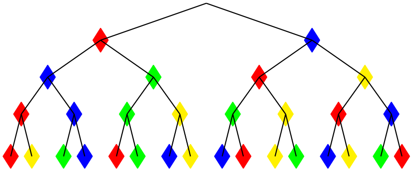
$\approx$

Online function

$\{0,1\}^{\leq n} \to \Sigma^{\leq n}$,

some alphabet Σ.

- Online analog of error-correcting codes
  - Can "decode" input from errored version of output with $< \frac{1}{2}$(distance) errors
- (Schulman 90s): Add error tolerance to *interactive communication protocols*
- Explicit constructions are challenging



$\approx$

Online function

$\{0,1\}^{\leq n} \to \Sigma^{\leq n}$,

some alphabet Σ.

- Online analog of error-correcting codes
  - Can "decode" input from errored version of output with $< \frac{1}{2}$(distance) errors
- (Schulman 90s): Add error tolerance to *interactive communication protocols*
- Explicit constructions are challenging
  - "Good" tree codes exist, but no poly-time construction is known!

(Binary) tree codes $TC : \{0,1\}^{\leq n} \to \Sigma^{\leq n}$, $|\Sigma| = \text{polylog}(n)$, distance $= \frac{1}{2}$.

(Binary) tree codes $TC : \{0,1\}^{\leq n} \to \Sigma^{\leq n}$, $|\Sigma| = \text{polylog}(n)$, distance $= \frac{1}{2}$.

Based on *integer* tree codes $TC_{\mathbb{Z}} : \mathbb{Z}^{\leq n} \to (\mathbb{Z}^2)^{\leq n}$, distance $= \frac{1}{2}$.

(Binary) tree codes $TC : \{0,1\}^{\leq n} \to \Sigma^{\leq n}$, $|\Sigma| = \mathrm{polylog}(n)$, distance $= \frac{1}{2}$.

Based on *integer* tree codes $TC_{\mathbb{Z}} : \mathbb{Z}^{\leq n} \to (\mathbb{Z}^2)^{\leq n}$, distance $= \frac{1}{2}$.

$$(z_0, z_1, \ldots, z_{n-1}) \mapsto ((z_0, a_0), (z_1, a_1), \ldots, (z_{n-1}, a_{n-1}))$$

under the Newton basis transformation

$$z_i = \sum_{j=0}^{n-1} a_j \binom{i}{j}, \qquad \forall i$$

with the inversion formula

$$a_j = \sum_{i=0}^{n-1} (-1)^{j-i} \binom{j}{i} z_i, \qquad \forall j.$$

(Binary) tree codes $TC : \{0,1\}^{\leq n} \to \Sigma^{\leq n}$, $|\Sigma| = \text{polylog}(n)$, distance $= \frac{1}{2}$.

Based on *integer* tree codes $TC_{\mathbb{Z}} : \mathbb{Z}^{\leq n} \to (\mathbb{Z}^2)^{\leq n}$, distance $= \frac{1}{2}$.

$$(z_0, z_1, \ldots, z_{n-1}) \mapsto ((z_0, a_0), (z_1, a_1), \ldots, (z_{n-1}, a_{n-1}))$$

under the Newton basis transformation

$$z_i = \sum_{j=0}^{n-1} a_j \binom{i}{j}, \qquad \forall i$$

$$p(i) = z_i,$$

with the inversion formula

$$p(x) = \sum_{j=0}^{n-1} a_j \binom{x}{j}$$

$$a_j = \sum_{i=0}^{n-1} (-1)^{j-i} \binom{j}{i} z_i, \qquad \forall j.$$

$$(z_0, z_1, \ldots, z_{n-1}) \mapsto ((z_0, a_0), (z_1, a_1), \ldots, (z_{n-1}, a_{n-1})),$$

$$z_i = \sum_{j=0}^{n-1} a_j \binom{i}{j}, \qquad \forall i.$$

$$(z_0, z_1, \ldots, z_{n-1}) \mapsto ((z_0, a_0), (z_1, a_1), \ldots, (z_{n-1}, a_{n-1})),$$

$$z_i = \sum_{j=0}^{n-1} a_j \binom{i}{j}, \qquad \forall i.$$

Theorem

If $z_0 \neq 0$*, then*

$$Sparsity(z_0, z_1, \ldots, z_{n-1}) + Sparsity(a_0, a_1, \ldots, a_{n-1}) \geq n + 1$$

$$\Rightarrow Sparsity((z_0, a_0), (z_1, a_1), \ldots, (z_{n-1}, a_{n-1})) \geq \frac{n}{2}.$$

Thus the CHS code has distance $1/2$.

Restated:

$$(z_0, z_1, \ldots, z_{n-1}) \mapsto ((z_0, a_0), (z_1, a_1), \ldots, (z_{n-1}, a_{n-1})),$$

$$p(x) = \sum_{j=0}^{n-1} a_j \binom{x}{j}, \qquad z_i = p(i), \ \ \forall i.$$

Theorem

If $p(0) \neq 0$, then the number of roots of $p(x)$ in $\mathbb{N}$ is less than its sparsity in the Newton basis $\{\binom{x}{j}\}_{j \geq 0}$.

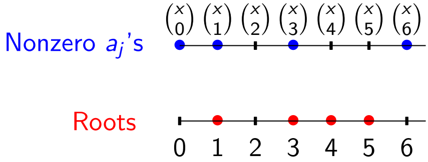
Theorem

If $p(0) \neq 0$, then the number of roots of $p(x) = \sum_{j=0}^{n-1} a_j \binom{x}{j}$ in $\mathbb{N}$ is less than its sparsity in the Newton basis $\{\binom{x}{j}\}_{j \geq 0}$.

Theorem

If $p(0) \neq 0$, then the number of roots of $p(x) = \sum_{j=0}^{n-1} a_j \binom{x}{j}$ in $\mathbb{N}$ is less than its sparsity in the Newton basis $\{\binom{x}{j}\}_{j \geq 0}$.

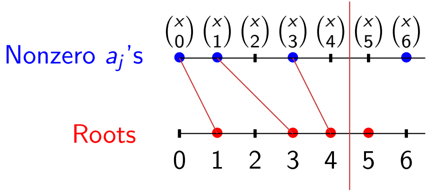
Proof.



Theorem

If $p(0) \neq 0$, then the number of roots of $p(x) = \sum_{j=0}^{n-1} a_j \binom{x}{j}$ in $\mathbb{N}$ is less than its sparsity in the Newton basis $\{\binom{x}{j}\}_{j \geq 0}$.

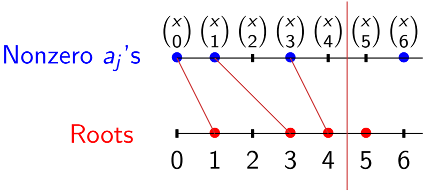
Proof.



Theorem

If $p(0) \neq 0$, then the number of roots of $p(x) = \sum_{j=0}^{n-1} a_j \binom{x}{j}$ in $\mathbb{N}$ is less than its sparsity in the Newton basis $\{\binom{x}{j}\}_{j \geq 0}$.

Proof.



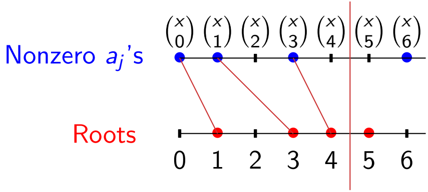
$J = \{\text{nonzero } a_j \text{ indices}\} = \{0, 1, 3\}$, $I = \{\text{roots}\} = \{1, 3, 4\}$.

Sparsity Theorem

Theorem

If $p(0) \neq 0$, then the number of roots of $p(x) = \sum_{j=0}^{n-1} a_j \binom{x}{j}$ in $\mathbb{N}$ is less than its sparsity in the Newton basis $\{\binom{x}{j}\}_{j \geq 0}$.

Proof.



$J = \{\text{nonzero } a_j \text{ indices}\} = \{0, 1, 3\}$, $I = \{\text{roots}\} = \{1, 3, 4\}$.

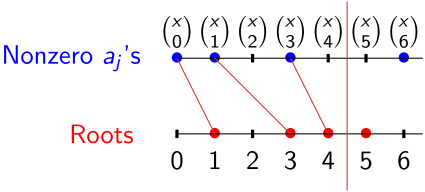
Then $M_{IJ} \cdot \begin{pmatrix} a_{j_1} & \cdots & a_{j_k} \end{pmatrix}^T = \vec{0}$, where M_{IJ} is the I, J minor of

$$M = \left\{ \binom{i}{j} \right\}_{0 \leq i, j \leq n-1}$$

Theorem

If $p(0) \neq 0$, then the number of roots of $p(x) = \sum_{j=0}^{n-1} a_j \binom{x}{j}$ in $\mathbb{N}$ is less than its sparsity in the Newton basis $\{\binom{x}{j}\}_{j \geq 0}$.

Proof.



$J = \{\text{nonzero } a_j \text{ indices}\} = \{0, 1, 3\}$, $I = \{\text{roots}\} = \{1, 3, 4\}$.
Then $M_{IJ} \cdot \begin{pmatrix} a_{j_1} & \cdots & a_{j_k} \end{pmatrix}^T = \vec{0}$, where M_{IJ} is the I, J minor of

$$M = \left\{ \binom{i}{j} \right\}_{0 \leq i,j \leq n-1}$$

But by the **Lindström-Gessel-Viennot Lemma**, $\det M_{IJ} \neq 0$. □

Our Work

Our Work

1. (Partial) decoding algorithm for the CHS tree codes

Our Work

1. (Partial) decoding algorithm for the CHS tree codes
2. Number-theoretic variants of $TC_{\mathbb{Z}}$ with similar parameters

Our Work

1. (Partial) decoding algorithm for the CHS tree codes
2. Number-theoretic variants of $TC_{\mathbb{Z}}$ with similar parameters
3. Speculative framework for decoding via convex optimization

$$(z_0, z_1, \ldots, z_{n-1}) \mapsto ((z_0, a_0), (z_1, a_1), \ldots, (z_{n-1}, a_{n-1})),$$

$$z_i = \sum_{j=0}^{n-1} a_j \binom{i}{j}, \qquad \forall i.$$

$$(z_0, z_1, \ldots, z_{n-1}) \mapsto ((z_0, a_0), (z_1, a_1), \ldots, (z_{n-1}, a_{n-1})),$$
$$z_i = \sum_{j=0}^{n-1} a_j \binom{i}{j}, \qquad \forall i.$$

Given a "received word" $((\widehat{z_0}, \widehat{a_0}), (\widehat{z_1}, \widehat{a_1}), \ldots, (\widehat{z_{n-1}}, \widehat{a_{n-1}}))$ such that $(\widehat{z_i}, \widehat{a_i}) = (z_i, a_i)$ except at $< n/4$ coordinates, output z_0, in time poly(n).

$$(z_0, z_1, \ldots, z_{n-1}) \mapsto ((z_0, a_0), (z_1, a_1), \ldots, (z_{n-1}, a_{n-1})),$$

$$z_i = \sum_{j=0}^{n-1} a_j \binom{i}{j}, \qquad \forall i.$$

Given a "received word" $((\widehat{z_0}, \widehat{a_0}), (\widehat{z_1}, \widehat{a_1}), \ldots, (\widehat{z_{n-1}}, \widehat{a_{n-1}}))$ such that $(\widehat{z_i}, \widehat{a_i}) = (z_i, a_i)$ except at $< n/4$ coordinates, output z_0, in time poly(n).

More generally, given the above and the prefix $z_0, \ldots, z_{k-1}$, if $(\widehat{z_i}, \widehat{a_i}) = (z_i, a_i)$ except at $< (n-k)/4$ coordinates $i \geq k$, output z_k.

$$(z_0, z_1, \ldots, z_{n-1}) \mapsto ((z_0, a_0), (z_1, a_1), \ldots, (z_{n-1}, a_{n-1})),$$

$$z_i = \sum_{j=0}^{n-1} a_j \binom{i}{j}, \qquad \forall i.$$

Given a "received word" $((\widehat{z_0}, \widehat{a_0}), (\widehat{z_1}, \widehat{a_1}), \ldots, (\widehat{z_{n-1}}, \widehat{a_{n-1}}))$ such that $(\widehat{z_i}, \widehat{a_i}) = (z_i, a_i)$ except at $< n/4$ coordinates, output z_0, in time poly(n).

More generally, given the above and the prefix $z_0, \ldots, z_{k-1}$, if $(\widehat{z_i}, \widehat{a_i}) = (z_i, a_i)$ except at $< (n-k)/4$ coordinates $i \geq k$, output z_k.

This yields a decoding algorithm for the binary tree codes, correcting up to $< n/4$ errors.

$$(z_0, z_1, \ldots, z_{n-1}) \mapsto ((z_0, a_0), (z_1, a_1), \ldots, (z_{n-1}, a_{n-1})),$$

$$z_i = \sum_{j=0}^{n-1} a_j \binom{i}{j}, \qquad \forall i.$$

Given a "received word" $((\widehat{z_0}, \widehat{a_0}), (\widehat{z_1}, \widehat{a_1}), \ldots, (\widehat{z_{n-1}}, \widehat{a_{n-1}}))$ such that $(\widehat{z_i}, \widehat{a_i}) = (z_i, a_i)$ except at ~~$< n/4$~~ $\Omega(\sqrt{n/\log(n)})$ coordinates, output z_0, in time poly(n).

More generally, given the above and the prefix $z_0, \ldots, z_{k-1}$, if $(\widehat{z_i}, \widehat{a_i}) = (z_i, a_i)$ except at $< (n-k)/4$ coordinates $i \geq k$, output z_k.

This yields a decoding algorithm for the binary tree codes, correcting up to ~~$< n/4$~~ $\Omega(n^{3/4}/\sqrt{\log(n)})$ *errors.*

Outline

Outline

1. Locate $\alpha(n)/2$ indices $i \in [0, \alpha(n))$ such that $\widehat{z_i} = z_i$, for some $\alpha(n) = \Omega(\sqrt{n \log(n)})$, using a Newton basis analog of the Sudan and Shokrollahi-Wasserman algorithms.

Outline

1. Locate $\alpha(n)/2$ indices $i \in [0, \alpha(n))$ such that $\widehat{z_i} = z_i$, for some $\alpha(n) = \Omega(\sqrt{n \log(n)})$, using a Newton basis analog of the Sudan and Shokrollahi-Wasserman algorithms.
2. Using a duality trick, do the same for the a_j's with the same algorithm.

Outline

1. Locate $\alpha(n)/2$ indices $i \in [0, \alpha(n))$ such that $\widehat{z_i} = z_i$, for some $\alpha(n) = \Omega(\sqrt{n \log(n)})$, using a Newton basis analog of the Sudan and Shokrollahi-Wasserman algorithms.
2. Using a duality trick, do the same for the a_j's with the same algorithm.
3. Among the first $\alpha(n)$ coordinates, we have as many known z_i's as unknown a_j's. Interpolate the remaining unknown a_j's using the Lindström-Gessel-Viennot Lemma, recovering $z_0 = a_0$.

Decoding: Step 1

1. Locate $\alpha(n)/2$ indices $i \in [0, \alpha(n))$ such that $\widehat{z_i} = z_i$, for some $\alpha(n) = \Omega(\sqrt{n \log(n)})$, using a Newton basis analog of the Sudan and Shokrollahi-Wasserman algorithms.

1. Locate $\alpha(n)/2$ indices $i \in [0, \alpha(n))$ such that $\widehat{z_i} = z_i$, for some $\alpha(n) = \Omega(\sqrt{n \log(n)})$, using a Newton basis analog of the Sudan and Shokrollahi-Wasserman algorithms.

WLOG all $\widehat{a_j} = 0$ (subtract evaluations of $\sum_{j=0}^{n-1} \widehat{a_j} \binom{x}{j}$ from $\widehat{z_i}$'s).

1. Locate $\alpha(n)/2$ indices $i \in [0, \alpha(n))$ such that $\widehat{z_i} = z_i$, for some $\alpha(n) = \Omega(\sqrt{n \log(n)})$, using a Newton basis analog of the Sudan and Shokrollahi-Wasserman algorithms.

WLOG all $\widehat{a_j} = 0$ (subtract evaluations of $\sum_{j=0}^{n-1} \widehat{a_j} \binom{x}{j}$ from $\widehat{z_i}$'s).

Algorithm

Let $\alpha = \alpha(n) = r\sqrt{n \log(n)}$, some $r \in \mathbb{R}$.

1. Locate $\alpha(n)/2$ indices $i \in [0, \alpha(n))$ such that $\widehat{z_i} = z_i$, for some $\alpha(n) = \Omega(\sqrt{n \log(n)})$, using a Newton basis analog of the Sudan and Shokrollahi-Wasserman algorithms.

WLOG all $\widehat{a_j} = 0$ (subtract evaluations of $\sum_{j=0}^{n-1} \widehat{a_j} \binom{x}{j}$ from $\widehat{z_i}$'s).

Algorithm

Let $\alpha = \alpha(n) = r\sqrt{n \log(n)}$, some $r \in \mathbb{R}$.

- Randomly choose indices $R \subset [\alpha, n)$, $|R| = n - 2\alpha + 1$.

1. Locate $\alpha(n)/2$ indices $i \in [0, \alpha(n))$ such that $\widehat{z_i} = z_i$, for some $\alpha(n) = \Omega(\sqrt{n \log(n)})$, using a Newton basis analog of the Sudan and Shokrollahi-Wasserman algorithms.

WLOG all $\widehat{a_j} = 0$ (subtract evaluations of $\sum_{j=0}^{n-1} \widehat{a_j}\binom{x}{j}$ from $\widehat{z_i}$'s).

Algorithm

Let $\alpha = \alpha(n) = r\sqrt{n \log(n)}$, some $r \in \mathbb{R}$.

- Randomly choose indices $R \subset [\alpha, n)$, $|R| = n - 2\alpha + 1$.
- Find polynomials

$$b(x) = \sum_{j \in [0,\alpha) \cup R} b_j \binom{x}{j}, \qquad c(x) = \sum_{j=0}^{\alpha-1} c_j x^j$$

$$\forall i \in [0, n),\ b(i) + c(i)\widehat{z_i} = 0.$$

1. Locate $\alpha(n)/2$ indices $i \in [0, \alpha(n))$ such that $\widehat{z_i} = z_i$, for some $\alpha(n) = \Omega(\sqrt{n \log(n)})$, using a Newton basis analog of the Sudan and Shokrollahi-Wasserman algorithms.

WLOG all $\widehat{a_j} = 0$ (subtract evaluations of $\sum_{j=0}^{n-1} \widehat{a_j} \binom{x}{j}$ from $\widehat{z_i}$'s).

Algorithm

Let $\alpha = \alpha(n) = r\sqrt{n \log(n)}$, some $r \in \mathbb{R}$.

- Randomly choose indices $R \subset [\alpha, n)$, $|R| = n - 2\alpha + 1$.
- Find polynomials

$$b(x) = \sum_{j \in [0,\alpha) \cup R} b_j \binom{x}{j}, \qquad c(x) = \sum_{j=0}^{\alpha-1} c_j x^j$$

$$\forall i \in [0, n),\ b(i) + c(i)\widehat{z_i} = 0.$$

- Output smallest $i_0 \in [0, \alpha)$ s.t. $c(i_0) \neq 0$.

Algorithm

Let $\alpha = \alpha(n) = r\sqrt{n \log(n)}$, some $r \in \mathbb{R}$.

- Randomly choose indices $R \subset [\alpha, n)$, $|R| = n - 2\alpha + 1$.
- Find polynomials

$$b(x) = \sum_{j \in [0,\alpha) \cup R} b_j \binom{x}{j}, \qquad c(x) = \sum_{j=0}^{\alpha-1} c_j x^j$$

$$\forall i \in [0, n),\ b(i) + c(i)\widehat{z_i} = 0.$$

- Output smallest $i_0 \in [0, \alpha)$ s.t. $c(i_0) \neq 0$.

Claim: $\widehat{\mathbf{z_{i_0}}} = \mathbf{z_{i_0}}$.

Algorithm

Let $\alpha = \alpha(n) = r\sqrt{n\log(n)}$, some $r \in \mathbb{R}$.

- Randomly choose indices $R \subset [\alpha, n)$, $|R| = n - 2\alpha + 1$.
- Find polynomials

$$b(x) = \sum_{j \in [0,\alpha) \cup R} b_j \binom{x}{j}, \qquad c(x) = \sum_{j=0}^{\alpha-1} c_j x^j$$

$$\forall i \in [0, n),\; b(i) + c(i)\widehat{z_i} = 0.$$

- Output smallest $i_0 \in [0, \alpha)$ s.t. $c(i_0) \neq 0$.

Claim: $\widehat{\mathbf{z}_{\mathbf{i_0}}} = \mathbf{z}_{\mathbf{i_0}}$.

Let $p(x) = \sum_{j=0}^{n-1} a_j \binom{x}{j}$, the correct polynomial. So $p(i) = z_i$.

Algorithm

Let $\alpha = \alpha(n) = r\sqrt{n\log(n)}$, some $r \in \mathbb{R}$.

- Randomly choose indices $R \subset [\alpha, n)$, $|R| = n - 2\alpha + 1$.
- Find polynomials

$$b(x) = \sum_{j\in[0,\alpha)\cup R} b_j \binom{x}{j}, \qquad c(x) = \sum_{j=0}^{\alpha-1} c_j x^j$$

$$\forall i \in [0, n),\ b(i) + c(i)\widehat{z_i} = 0.$$

- Output smallest $i_0 \in [0, \alpha)$ s.t. $c(i_0) \neq 0$.

Claim: $\widehat{\mathbf{z_{i_0}}} = \mathbf{z_{i_0}}$.

Let $p(x) = \sum_{j=0}^{n-1} a_j \binom{x}{j}$, the correct polynomial. So $p(i) = z_i$.
Bound on # errors $\implies p(x)$ is $O(\sqrt{n/\log(n)})$-sparse, and the $\widehat{z_i}$ are the outputs of $p(x)$ except with $O(\sqrt{n/\log(n)})$ errors.

Algorithm

Let $\alpha = \alpha(n) = r\sqrt{n \log(n)}$, some $r \in \mathbb{R}$.

- Randomly choose indices $R \subset [\alpha, n)$, $|R| = n - 2\alpha + 1$.
- Find polynomials

$$b(x) = \sum_{j \in [0,\alpha) \cup R} b_j \binom{x}{j}, \qquad c(x) = \sum_{j=0}^{\alpha-1} c_j x^j$$

$$\forall i \in [0, n),\ b(i) + c(i)\widehat{z_i} = 0.$$

- Output smallest $i_0 \in [0, \alpha)$ s.t. $c(i_0) \neq 0$.

Claim: $\widehat{\mathbf{z_{i_0}}} = \mathbf{z_{i_0}}$.

Let $p(x) = \sum_{j=0}^{n-1} a_j \binom{x}{j}$, the correct polynomial. So $p(i) = z_i$.
Bound on # errors $\Longrightarrow$ $p(x)$ is $O(\sqrt{n/\log(n)})$-sparse, and the $\widehat{z_i}$ are the outputs of $p(x)$ except with $O(\sqrt{n/\log(n)})$ errors.

Plan: Show $b(x) + p(x)c(x)$ is sparse but also has many roots, hence is ≈ 0. (Specifically, $b(i_0) + p(i_0)c(i_0) = 0$.)

Algorithm

Let $\alpha = \alpha(n) = r\sqrt{n\log(n)}$, some $r \in \mathbb{R}$.

- Randomly choose indices $R \subset [\alpha, n)$, $|R| = n - 2\alpha + 1$.
- Find polynomials

$$b(x) = \sum_{j\in[0,\alpha)\cup R} b_j \binom{x}{j}, \qquad c(x) = \sum_{j=0}^{\alpha-1} c_j x^j$$

$$\forall i \in [0, n),\ b(i) + c(i)\widehat{z_i} = 0.$$

- Output smallest $i_0 \in [0, \alpha)$ s.t. $c(i_0) \neq 0$.

Claim: $\widehat{\mathbf{z_{i_0}}} = \mathbf{z_{i_0}}$.

Let $p(x) = \sum_{j=0}^{n-1} a_j \binom{x}{j}$, the correct polynomial. So $p(i) = z_i$.
Bound on # errors $\Longrightarrow$ $p(x)$ is $O(\sqrt{n/\log(n)})$-sparse, and the $\widehat{z_i}$ are the outputs of $p(x)$ except with $O(\sqrt{n/\log(n)})$ errors.

Plan: Show $b(x) + p(x)c(x)$ is sparse but also has many roots, hence is ≈ 0. (Specifically, $b(i_0) + p(i_0)c(i_0) = 0$.)

Then $c(i_0) \neq 0 \Rightarrow \widehat{z_{i_0}} = p(i_0) = z_{i_0}$, as desired.

Decoding: Step 1

$$\alpha = O(\sqrt{n\log(n)}), \quad |R| = n - 2\alpha + 1, \quad b(x) = \sum_{j\in[0,\alpha)\cup R} b_j \binom{x}{j}, \quad c(x) = \sum_{j=0}^{\alpha-1} c_j x^j$$

$$\forall i \in [0,n),\ b(i) + c(i)\widehat{z_i} = 0$$

$$p(x) = \textstyle\sum_{j=0}^{n-1} a_j \binom{x}{j},\ p(i) = z_i$$

sparsity$(p(x)) = O(\sqrt{n/\log(n)})$, $z_i = \widehat{z_i}$ except for $O(\sqrt{n/\log(n)})$ errors.

Decoding: Step 1

$$\alpha = O(\sqrt{n\log(n)}), \quad |R| = n - 2\alpha + 1, \quad b(x) = \sum_{j\in[0,\alpha)\cup R} b_j\binom{x}{j}, \quad c(x) = \sum_{j=0}^{\alpha-1} c_j x^j$$

$$\forall i \in [0,n),\ b(i) + c(i)\widehat{z_i} = 0$$

$$p(x) = \textstyle\sum_{j=0}^{n-1} a_j\binom{x}{j},\ p(i) = z_i$$

sparsity$(p(x)) = O(\sqrt{n/\log(n)})$, $z_i = \widehat{z_i}$ except for $O(\sqrt{n/\log(n)})$ errors.

Proof that $b(i_0) + p(i_0)c(i_0) = 0$.

$$\alpha = O(\sqrt{n\log(n)}), \quad |R| = n - 2\alpha + 1, \quad b(x) = \sum_{j\in[0,\alpha)\cup R} b_j\binom{x}{j}, \quad c(x) = \sum_{j=0}^{\alpha-1} c_j x^j$$

$$\forall i \in [0,n),\ b(i) + c(i)\widehat{z_i} = 0$$

$$p(x) = \textstyle\sum_{j=0}^{n-1} a_j\binom{x}{j},\ p(i) = z_i$$

sparsity$(p(x)) = O(\sqrt{n/\log(n)})$, $z_i = \widehat{z_i}$ except for $O(\sqrt{n/\log(n)})$ errors.

Proof that $b(i_0) + p(i_0)c(i_0) = 0$.

1. $b(x) + p(x)c(x)$ has $n - \Omega(\sqrt{n/\log(n)})$ roots (each i s.t. $\widehat{z_i} = z_i$).

$$\alpha = O(\sqrt{n \log(n)}), \quad |R| = n - 2\alpha + 1, \quad b(x) = \sum_{j \in [0,\alpha) \cup R} b_j \binom{x}{j}, \quad c(x) = \sum_{j=0}^{\alpha-1} c_j x^j$$

$$\forall i \in [0, n),\ b(i) + c(i)\widehat{z_i} = 0$$

$$p(x) = \textstyle\sum_{j=0}^{n-1} a_j \binom{x}{j},\ p(i) = z_i$$

sparsity$(p(x)) = O(\sqrt{n/\log(n)})$, $z_i = \widehat{z_i}$ except for $O(\sqrt{n/\log(n)})$ errors.

Proof that $b(i_0) + p(i_0)c(i_0) = 0$.

1. $b(x) + p(x)c(x)$ has $n - \Omega(\sqrt{n/\log(n)})$ roots (each i s.t. $\widehat{z_i} = z_i$).
2. $b(x) + p(x)c(x)$ is $n - \Omega(\sqrt{n/\log(n)})$ sparse w.h.p.:

$$\alpha = O(\sqrt{n \log(n)}), \quad |R| = n - 2\alpha + 1, \quad b(x) = \sum_{j \in [0,\alpha) \cup R} b_j \binom{x}{j}, \quad c(x) = \sum_{j=0}^{\alpha-1} c_j x^j$$

$$\forall i \in [0, n),\ b(i) + c(i)\widehat{z_i} = 0$$

$$p(x) = \textstyle\sum_{j=0}^{n-1} a_j \binom{x}{j},\ p(i) = z_i$$

sparsity$(p(x)) = O(\sqrt{n/\log(n)})$, $z_i = \widehat{z_i}$ except for $O(\sqrt{n/\log(n)})$ errors.

Proof that $b(i_0) + p(i_0)c(i_0) = 0$.

1. $b(x) + p(x)c(x)$ has $n - \Omega(\sqrt{n/\log(n)})$ roots (each i s.t. $\widehat{z_i} = z_i$).
2. $b(x) + p(x)c(x)$ is $n - \Omega(\sqrt{n/\log(n)})$ sparse w.h.p.:

- sparsity$(p(x)c(x)) \leq$
  sparsity$(p(x)) \times \deg(c(x)) \leq O(n)$.

$$\alpha = O(\sqrt{n\log(n)}), \quad |R| = n - 2\alpha + 1, \quad b(x) = \sum_{j\in[0,\alpha)\cup R} b_j \binom{x}{j}, \quad c(x) = \sum_{j=0}^{\alpha-1} c_j x^j$$

$$\forall i \in [0,n),\ b(i) + c(i)\widehat{z_i} = 0$$

$$p(x) = \textstyle\sum_{j=0}^{n-1} a_j \binom{x}{j},\ p(i) = z_i$$

$$\text{sparsity}(p(x)) = O(\sqrt{n/\log(n)}),\ z_i = \widehat{z_i} \text{ except for } O(\sqrt{n/\log(n)}) \text{ errors.}$$

Proof that $b(i_0) + p(i_0)c(i_0) = 0$.

1. $b(x) + p(x)c(x)$ has $n - \Omega(\sqrt{n/\log(n)})$ roots (each i s.t. $\widehat{z_i} = z_i$).
2. $b(x) + p(x)c(x)$ is $n - \Omega(\sqrt{n/\log(n)})$ sparse w.h.p.:

- sparsity($p(x)c(x)$) $\leq$
  sparsity($p(x)$) $\times$ deg($c(x)$) $\leq O(n)$.

Decoding: Step 1

$$\alpha = O(\sqrt{n \log(n)}), \quad |R| = n - 2\alpha + 1, \quad b(x) = \sum_{j \in [0,\alpha) \cup R} b_j \binom{x}{j}, \quad c(x) = \sum_{j=0}^{\alpha-1} c_j x^j$$

$$\forall i \in [0, n),\ b(i) + c(i)\widehat{z_i} = 0$$

$$p(x) = \textstyle\sum_{j=0}^{n-1} a_j \binom{x}{j},\ p(i) = z_i$$

sparsity$(p(x)) = O(\sqrt{n/\log(n)})$, $z_i = \widehat{z_i}$ except for $O(\sqrt{n/\log(n)})$ errors.

Proof that $b(i_0) + p(i_0)c(i_0) = 0$.

1. $b(x) + p(x)c(x)$ has $n - \Omega(\sqrt{n/\log(n)})$ roots (each i s.t. $\widehat{z_i} = z_i$).
2. $b(x) + p(x)c(x)$ is $n - \Omega(\sqrt{n/\log(n)})$ sparse w.h.p.:

- sparsity$(p(x)c(x)) \le$
  sparsity$(p(x)) \times \deg(c(x)) \ \le O(n)$.

- $+b(x)$: union with $[0, \alpha) \cup R$ of size
  $n - \Omega(\sqrt{n \log(n)})$.

Decoding: Step 1

$$\alpha = O(\sqrt{n\log(n)}), \quad |R| = n - 2\alpha + 1, \quad b(x) = \sum_{j\in[0,\alpha)\cup R} b_j \binom{x}{j}, \quad c(x) = \sum_{j=0}^{\alpha-1} c_j x^j$$

$$\forall i \in [0,n),\ b(i) + c(i)\widehat{z_i} = 0$$

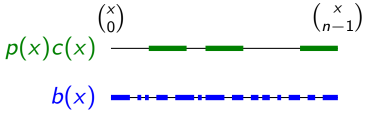
$$p(x) = \textstyle\sum_{j=0}^{n-1} a_j \binom{x}{j},\ p(i) = z_i$$

sparsity$(p(x)) = O(\sqrt{n/\log(n)})$, $z_i = \widehat{z_i}$ except for $O(\sqrt{n/\log(n)})$ errors.

Proof that $b(i_0) + p(i_0)c(i_0) = 0$.

1. $b(x) + p(x)c(x)$ has $n - \Omega(\sqrt{n/\log(n)})$ roots (each i s.t. $\widehat{z_i} = z_i$).
2. $b(x) + p(x)c(x)$ is $n - \Omega(\sqrt{n/\log(n)})$ sparse w.h.p.:

- sparsity$(p(x)c(x)) \leq$
  sparsity$(p(x)) \times \deg(c(x)) \ \leq O(n)$.
- $+b(x)$: union with $[0,\alpha) \cup R$ of size $n - \Omega(\sqrt{n\log(n)})$.



$$\alpha = O(\sqrt{n\log(n)}), \quad |R| = n - 2\alpha + 1, \quad b(x) = \sum_{j\in[0,\alpha)\cup R} b_j \binom{x}{j}, \quad c(x) = \sum_{j=0}^{\alpha-1} c_j x^j$$

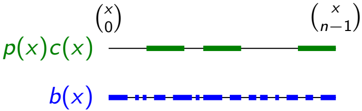
$$\forall i \in [0,n),\ b(i) + c(i)\widehat{z_i} = 0$$

$$p(x) = \textstyle\sum_{j=0}^{n-1} a_j \binom{x}{j},\ p(i) = z_i$$

sparsity$(p(x)) = O(\sqrt{n/\log(n)})$, $z_i = \widehat{z_i}$ except for $O(\sqrt{n/\log(n)})$ errors.

Proof that $b(i_0) + p(i_0)c(i_0) = 0$.

1. $b(x) + p(x)c(x)$ has $n - \Omega(\sqrt{n/\log(n)})$ roots (each i s.t. $\widehat{z_i} = z_i$).
2. $b(x) + p(x)c(x)$ is $n - \Omega(\sqrt{n/\log(n)})$ sparse w.h.p.:
   - sparsity$(p(x)c(x)) \leq$ sparsity$(p(x)) \times \deg(c(x)) \leq O(n)$.
   - $+b(x)$: union with $[0,\alpha)\cup R$ of size $n - \Omega(\sqrt{n\log(n)})$.



3. $\#$ roots $\geq$ sparsity $\Longrightarrow b(0) + p(0)c(0) = 0$.

$$\alpha = O(\sqrt{n\log(n)}), \quad |R| = n - 2\alpha + 1, \quad b(x) = \sum_{j\in[0,\alpha)\cup R} b_j\binom{x}{j}, \quad c(x) = \sum_{j=0}^{\alpha-1} c_j x^j$$

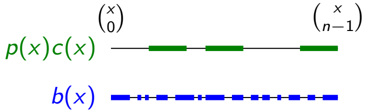
$$\forall i \in [0,n),\ b(i) + c(i)\widehat{z_i} = 0$$

$$p(x) = \textstyle\sum_{j=0}^{n-1} a_j\binom{x}{j},\ p(i) = z_i$$

sparsity$(p(x)) = O(\sqrt{n/\log(n)})$, $z_i = \widehat{z_i}$ except for $O(\sqrt{n/\log(n)})$ errors.

Proof that $b(i_0) + p(i_0)c(i_0) = 0$.

1. $b(x) + p(x)c(x)$ has $n - \Omega(\sqrt{n/\log(n)})$ roots (each i s.t. $\widehat{z_i} = z_i$).
2. $b(x) + p(x)c(x)$ is $n - \Omega(\sqrt{n/\log(n)})$ sparse w.h.p.:
   - sparsity$(p(x)c(x)) \le$ sparsity$(p(x)) \times \deg(c(x)) \le O(n)$.
   - $+b(x)$: union with $[0,\alpha) \cup R$ of size $n - \Omega(\sqrt{n\log(n)})$.



3. $\#$ roots $\ge$ sparsity $\implies b(0) + p(0)c(0) = 0$.
4. If $c(0) = 0$, then $b(0) = 0$. Refine sparsity bound $\implies$ $b(1) + p(1)c(1) = 0, \ldots, b(i_0) + p(i_0)c(i_0) = 0$.

□

Decoding: Step 1

$$\alpha = O(\sqrt{n\log(n)}), \quad |R| = n - 2\alpha + 1, \quad b(x) = \sum_{j\in[0,\alpha)\cup R} b_j \binom{x}{j}, \quad c(x) = \sum_{j=0}^{\alpha-1} c_j x^j$$

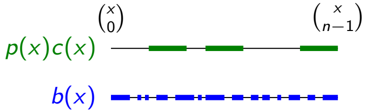
$$\forall i \in [0,n),\ b(i) + c(i)\widehat{z_i} = 0$$

$$p(x) = \textstyle\sum_{j=0}^{n-1} a_j \binom{x}{j},\ p(i) = z_i$$

sparsity$(p(x)) = O(\sqrt{n/\log(n)})$, $z_i = \widehat{z_i}$ except for $O(\sqrt{n/\log(n)})$ errors.

Proof that $b(i_0) + p(i_0)c(i_0) = 0$.

1. $b(x) + p(x)c(x)$ has $n - \Omega(\sqrt{n/\log(n)})$ roots (each i s.t. $\widehat{z_i} = z_i$).
2. $b(x) + p(x)c(x)$ is $n - \Omega(\sqrt{n/\log(n)})$ sparse w.h.p.:
   - sparsity$(p(x)c(x)) \leq$ sparsity$(p(x)) \times \deg(c(x))$ $\leq O(n)$.
   - $+b(x)$: union with $[0,\alpha) \cup R$ of size $n - \Omega(\sqrt{n\log(n)})$.



3. $\#$ roots $\geq$ sparsity $\implies b(0) + p(0)c(0) = 0$.
4. If $c(0) = 0$, then $b(0) = 0$. Refine sparsity bound $\implies$ $b(1) + p(1)c(1) = 0, \ldots, b(i_0) + p(i_0)c(i_0) = 0$. □

$$\alpha = O(\sqrt{n \log(n)}), \quad |R| = n - 2\alpha + 1, \quad b(x) = \sum_{j \in [0,\alpha) \cup R} b_j \binom{x}{j}, \quad c(x) = \sum_{j=0}^{\alpha-1} c_j x^j$$

$$\forall i \in [0, n),\ b(i) + c(i)\widehat{z_i} = 0$$

$$p(x) = \textstyle\sum_{j=0}^{n-1} a_j \binom{x}{j},\ p(i) = z_i$$

sparsity$(p(x)) = O(\sqrt{n/\log(n)})$, $z_i = \widehat{z_i}$ except for $O(\sqrt{n/\log(n)})$ errors.

Proof that $b(i_0) + p(i_0)c(i_0) = 0$.

1. $b(x) + p(x)c(x)$ has $n - \Omega(\sqrt{n/\log(n)})$ roots (each i s.t. $\widehat{z_i} = z_i$).
2. $b(x) + p(x)c(x)$ is $n - \Omega(\sqrt{n/\log(n)})$ sparse w.h.p.:
   - sparsity$(p(x)c(x)) \leq$ sparsity$(p(x)) \times \deg(c(x)) \leq O(n)$.
   - $+b(x)$: union with $[0, \alpha) \cup R$ of size $n - \Omega(\sqrt{n \log(n)})$.

   In the Reed-Solomon code version, these are degrees, which add.
3. $\#$ roots $\geq$ sparsity $\implies b(0) + p(0)c(0) = 0$.
4. If $c(0) = 0$, then $b(0) = 0$. Refine sparsity bound $\implies$ $b(1) + p(1)c(1) = 0, \ldots, b(i_0) + p(i_0)c(i_0) = 0$. □

2. Using a duality trick, use the same algorithm to locate $\alpha(n)/2$ indices $j \in [0, \alpha(n))$ such that $\widehat{a_j} = a_j$, for some $\alpha(n) = \Omega(\sqrt{n \log(n)})$.

2. Using a duality trick, use the same algorithm to locate $\alpha(n)/2$ indices $j \in [0, \alpha(n))$ such that $\widehat{a_j} = a_j$, for some $\alpha(n) = \Omega(\sqrt{n \log(n)})$.

Inversion formula:

$$z_i = \sum_{j=0}^{n-1} a_j \binom{i}{j}, \qquad a_j = \sum_{i=0}^{n-1} (-1)^{j-i} \binom{j}{i} z_i.$$

2. Using a duality trick, use the same algorithm to locate $\alpha(n)/2$ indices $j \in [0, \alpha(n))$ such that $\widehat{a_j} = a_j$, for some $\alpha(n) = \Omega(\sqrt{n \log(n)})$.

Inversion formula:

$$z_i = \sum_{j=0}^{n-1} a_j \binom{i}{j}, \qquad a_j = \sum_{i=0}^{n-1} (-1)^{j-i} \binom{j}{i} z_i.$$

Thus the matrix of the transformation $(a_j) \mapsto (z_i)$ is the same as its inverse except that the rows and columns are scaled by $\begin{pmatrix} 1 & -1 & 1 & -1 & \cdots \end{pmatrix}$.

2. Using a duality trick, use the same algorithm to locate $\alpha(n)/2$ indices $j \in [0, \alpha(n))$ such that $\widehat{a_j} = a_j$, for some $\alpha(n) = \Omega(\sqrt{n \log(n)})$.

Inversion formula:

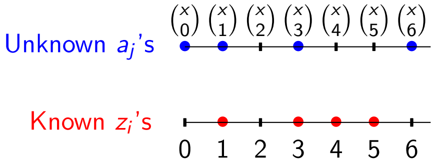
$$z_i = \sum_{j=0}^{n-1} a_j \binom{i}{j}, \qquad a_j = \sum_{i=0}^{n-1} (-1)^{j-i} \binom{j}{i} z_i.$$

Thus the matrix of the transformation $(a_j) \mapsto (z_i)$ is the same as its inverse except that the rows and columns are scaled by $\begin{pmatrix} 1 & -1 & 1 & -1 & \cdots \end{pmatrix}$.

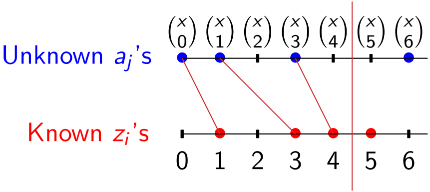
Treat $((\widehat{a_0}, \widehat{z_0}), (-\widehat{a_1}, -\widehat{z_1}), (\widehat{a_2}, \widehat{z_2}), \dots)$ as an errored encoding of $(a_0, -a_1, a_2, \dots)$ and apply step 1 again.

3. Among the first $\alpha(n)$ coordinates, we have as many known z_i's as unknown a_j's. Interpolate the remaining unknown a_j's using the Lindström-Gessel-Viennot Lemma, recovering $z_0 = a_0$.

3. Among the first $\alpha(n)$ coordinates, we have as many known z_i's as unknown a_j's. Interpolate the remaining unknown a_j's using the Lindström-Gessel-Viennot Lemma, recovering $z_0 = a_0$.

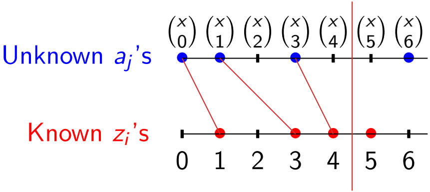


3. Among the first $\alpha(n)$ coordinates, we have as many known z_i's as unknown a_j's. Interpolate the remaining unknown a_j's using the Lindström-Gessel-Viennot Lemma, recovering $z_0 = a_0$.



$J = \{\text{nonzero } a_j \text{ indices}\} = \{0, 1, 3\}$, $I = \{\text{roots}\} = \{1, 3, 4\}$.

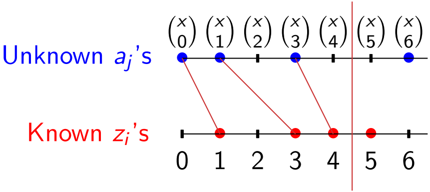
❸ Among the first $\alpha(n)$ coordinates, we have as many known z_i's as unknown a_j's. Interpolate the remaining unknown a_j's using the Lindström-Gessel-Viennot Lemma, recovering $z_0 = a_0$.



$J = \{\text{nonzero } a_j \text{ indices}\} = \{0, 1, 3\}$, $I = \{\text{roots}\} = \{1, 3, 4\}$.

$$M_{IJ} \cdot \begin{pmatrix} a_{j_1} \\ \vdots \\ a_{j_k} \end{pmatrix} = \begin{pmatrix} z_{i_1} = \widehat{z_{i_1}} \\ \vdots \\ z_{i_k} = \widehat{z_{i_k}} \end{pmatrix}, \quad M = \left\{ \binom{i}{j} \right\}_{0 \le i,j \le n-1}$$

3. Among the first $\alpha(n)$ coordinates, we have as many known z_i's as unknown a_j's. Interpolate the remaining unknown a_j's using the Lindström-Gessel-Viennot Lemma, recovering $z_0 = a_0$.



$J = \{\text{nonzero } a_j \text{ indices}\} = \{0, 1, 3\}$, $I = \{\text{roots}\} = \{1, 3, 4\}$.

$$M_{IJ} \cdot \begin{pmatrix} a_{j_1} \\ \vdots \\ a_{j_k} \end{pmatrix} = \begin{pmatrix} z_{i_1} = \widehat{z_{i_1}} \\ \vdots \\ z_{i_k} = \widehat{z_{i_k}} \end{pmatrix}, \quad M = \left\{ \binom{i}{j} \right\}_{0 \le i,j \le n-1}$$

All $j_\ell \le i_\ell \Longrightarrow \det(M_{IJ}) \ne 0$ by LGV. Solve for $a_{j_1} = a_0 = z_0$. □

Number-theoretic Variants of $TC_{\mathbb{Z}}$ with Similar Parameters

We generalized the CHS integer tree code construction and found variants with similar parameters.

Number-theoretic Variants of $TC_{\mathbb{Z}}$ with Similar Parameters

We generalized the CHS integer tree code construction and found variants with similar parameters.

In place of binomials $\binom{i}{j}$, use *q-binomials* $\begin{bmatrix} i \\ j \end{bmatrix}_q$:

$$(x+y)^i = \sum_{j=0}^{i} \begin{bmatrix} i \\ j \end{bmatrix}_q x^j y^{i-j} \qquad \text{where } yx = qxy.$$

We generalized the CHS integer tree code construction and found variants with similar parameters.

In place of binomials $\binom{i}{j}$, use *q-binomials* $\begin{bmatrix} i \\ j \end{bmatrix}_q$:

$$(x+y)^i = \sum_{j=0}^{i} \begin{bmatrix} i \\ j \end{bmatrix}_q x^j y^{i-j} \qquad \text{where } yx = qxy.$$

- $q = \zeta_\ell$ for $\ell > n^3$ prime
- $q = e^{2\pi\iota\theta}$ for $\theta \in \mathbf{R}$ irrational & algebraic (w/ rounding)

Speculative Framework for Decoding via Convex Optimization

Decoding problem: given $\widehat{x}$ = errored received output, solve

$$\min_{z \in \mathbf{R}^n} ||\widehat{x} - \mathrm{Enc}(z)||_{\ell_0}.$$

Speculative Framework for Decoding via Convex Optimization

Decoding problem: given $\widehat{x}$ = errored received output, solve

$$\min_{z \in \mathbf{R}^n} ||\widehat{x} - \mathrm{Enc}(z)||_{\ell_0}.$$

Idea: Instead solve

$$\min_{z \in \mathbf{R}^n} ||\widehat{x} - \mathrm{Enc}(z)||_{\ell_1}.$$

Speculative Framework for Decoding via Convex Optimization

Decoding problem: given $\widehat{x} =$ errored received output, solve

$$\min_{z \in \mathbf{R}^n} ||\widehat{x} - \mathrm{Enc}(z)||_{\ell_0}.$$

Idea: Instead solve

$$\min_{z \in \mathbf{R}^n} ||\widehat{x} - \mathrm{Enc}(z)||_{\ell_1}.$$

Candes & Tao 2005: This works for a constant fraction of errors if the parity check matrix F satisfies the *restricted isometry property (RIP)*.

Speculative Framework for Decoding via Convex Optimization

Decoding problem: given $\widehat{x}$ = errored received output, solve

$$\min_{z \in \mathbf{R}^n} ||\widehat{x} - \text{Enc}(z)||_{\ell_0}.$$

Idea: Instead solve

$$\min_{z \in \mathbf{R}^n} ||\widehat{x} - \text{Enc}(z)||_{\ell_1}.$$

Candes & Tao 2005: This works for a constant fraction of errors if the parity check matrix F satisfies the *restricted isometry property (RIP)*.

Suffices to construct such an F with shape:

$$\begin{pmatrix} * & * & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ * & * & * & * & 0 & 0 & \cdots & 0 & 0 \\ * & * & * & * & * & * & \cdots & 0 & 0 \\ \vdots & & & & & & \ddots & & \vdots \\ * & * & * & * & * & * & \cdots & * & * \end{pmatrix}$$

Speculative Framework for Decoding via Convex Optimization

Decoding problem: given $\widehat{x} =$ errored received output, solve

$$\min_{z \in \mathbf{R}^n} ||\widehat{x} - \mathrm{Enc}(z)||_{\ell_0}.$$

Idea: Instead solve

$$\min_{z \in \mathbf{R}^n} ||\widehat{x} - \mathrm{Enc}(z)||_{\ell_1}.$$

Candes & Tao 2005: This works for a constant fraction of errors if the parity check matrix F satisfies the *restricted isometry property (RIP)*.

Suffices to construct such an F with shape:

$$\begin{pmatrix} * & * & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ * & * & * & * & 0 & 0 & \cdots & 0 & 0 \\ * & * & * & * & * & * & \cdots & 0 & 0 \\ \vdots & & & & & & \ddots & & \vdots \\ * & * & * & * & * & * & \cdots & * & * \end{pmatrix}$$

However, this appears impossible. Need new "online RIP".

Closing Thought: Hyperinvertible Matrices

Hyperinvertible matrix—matrix where every square minor is invertible.

Closing Thought: Hyperinvertible Matrices

Hyperinvertible matrix—matrix where every square minor is invertible.

CHS encoding of $\vec{z}$ is $(\vec{z}, M\vec{z})$, where M is an $n \times n$ "online hyperinvertible" matrix—lower triangular analog.

Hyperinvertible matrix—matrix where every square minor is invertible.

CHS encoding of $\vec{z}$ is $(\vec{z}, M\vec{z})$, where M is an $n \times n$ "online hyperinvertible" matrix—lower triangular analog.

Open problem: construct $n \times n$ online hyperinvertible matrices over small finite fields.

Closing Thought: Hyperinvertible Matrices

Hyperinvertible matrix—matrix where every square minor is invertible.

CHS encoding of $\vec{z}$ is $(\vec{z}, M\vec{z})$, where M is an $n \times n$ "online hyperinvertible" matrix—lower triangular analog.

Open problem: construct $n \times n$ online hyperinvertible matrices over small finite fields.

- Binomial matrix gives $\mathbb{F}_{2^{\mathrm{poly}(n)}}$.

Closing Thought: Hyperinvertible Matrices

Hyperinvertible matrix—matrix where every square minor is invertible.

CHS encoding of $\vec{z}$ is $(\vec{z}, M\vec{z})$, where M is an $n \times n$ "online hyperinvertible" matrix—lower triangular analog.

Open problem: construct $n \times n$ online hyperinvertible matrices over small finite fields.

- Binomial matrix gives $\mathbb{F}_{2^{\text{poly}(n)}}$.
- $\mathbb{F}_{\text{poly}(n)} \Rightarrow$ binary tree codes of distance $1/2$ and alphabet size $2^{\log^*(n)}$.

Closing Thought: Hyperinvertible Matrices

Hyperinvertible matrix—matrix where every square minor is invertible.

CHS encoding of $\vec{z}$ is $(\vec{z}, M\vec{z})$, where M is an $n \times n$ "online hyperinvertible" matrix—lower triangular analog.

Open problem: construct $n \times n$ online hyperinvertible matrices over small finite fields.

- Binomial matrix gives $\mathbb{F}_{2^{\text{poly}(n)}}$.
- $\mathbb{F}_{\text{poly}(n)} \Rightarrow$ binary tree codes of distance $1/2$ and alphabet size $2^{\log^*(n)}$.

Without online condition, Beerliová-Trubíniová and Hirt construct

$$M = \left\{ \prod_{k \neq j} \frac{\beta_i - \alpha_k}{\alpha_j - \alpha_k} \right\}_{0 \leq i,j \leq n-1}$$

over $\mathbb{F}_{2n} = \{\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n\}$. (Interpolate $g(\alpha_j) = z_j$, then output $(g(\beta_i))_i$.)