

Two New Bounds for Deletion Codes

Xiaoyu He (Princeton University)

February 9, 2023

Setting the Stage

The Beginning

Noisy Channel Encoding Theorem (Shannon '48)

For a binary symmetric channel with error rate $\epsilon \in (0; 1)$, let $C_\epsilon = \{1 - \epsilon, \epsilon\}$. For any rate $\rho < C_\epsilon$, there exists a code in $\{0, 1\}^n$ of size $2^{n\rho}$ that w.h.p. correctly transmits information.

The Beginning

Noisy Channel Encoding Theorem (Shannon '48)

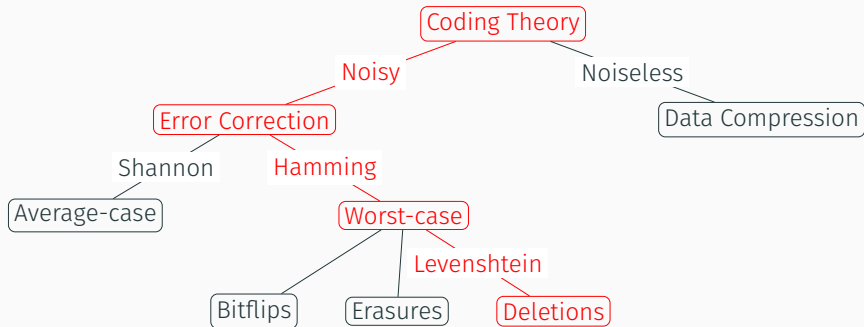
For a binary symmetric channel with error rate $\epsilon \in (0, 1)$, let $C = \{0, 1\}^n$. For any rate $R < C$, there exists a code in $\{0, 1\}^n$ of size 2^{nR} that w.h.p. correctly transmits information.

}

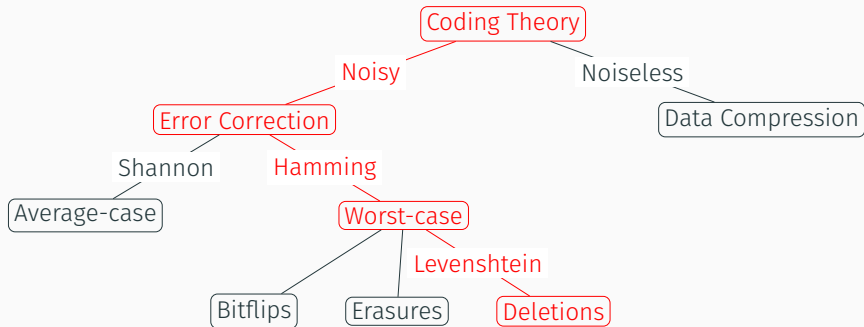
}

— Richard Hamming, on Claude Shannon.

Phylogeny of Coding Theory

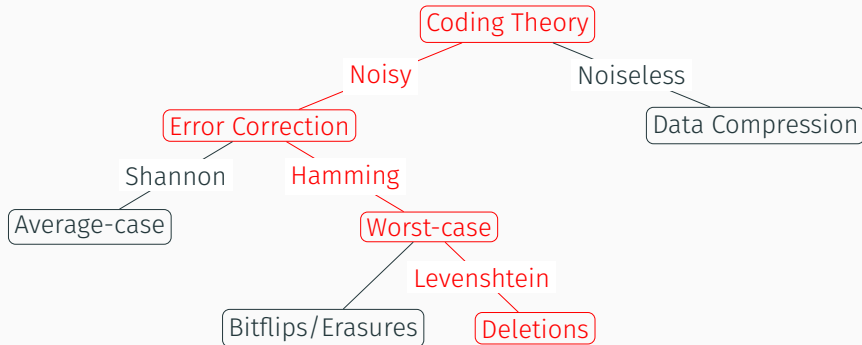


Phylogeny of Coding Theory



Everything is binary!

Phylogeny of coding theory



Everything is binary!

Noise models

Electromagnetic signal

Bitflip errors 1101 ! 1001

Noise Models

Noise models

Electromagnetic signal

Bitflip errors 1101 ! 1001

Auditory experience

Erasure errors 1101 ! 1?01

Noise Models

Noise models

Electromagnetic signal

Bitflip errors 1101 ! 1001

Auditory experience

Erasure errors 1101 ! 1?01

Transcribed lyrics

Deletion errors 1101 ! 101

Background: Bitflips and Erasures

Definition

A $\mathcal{C} \subseteq \{0,1\}^n$ is a subset; $\mathcal{C} \subseteq \{0,1\}^n$ such that
 $\min_{x \neq y \in \mathcal{C}} \text{Hamming}(x, y) = d$ over all distinct $x, y \in \mathcal{C}$. Let $M(n, d)$ denote the size of the largest such code.

Background: Bitflips and Erasures

Definition

A code C is a subset of $\{0,1\}^n$ such that $\min_{x \neq y \in C} \text{Hamming}(x, y) = d$ over all distinct $x, y \in C$. Let $n(d, c)$ denote the size of the largest such code.

Such a code corrects $\lfloor \frac{d-1}{2} \rfloor c$ bitflip errors or $\lfloor \frac{d-1}{2} \rfloor c + 1$ erasure errors.

Background: Bitflips and Erasures

Definition

A $\mathcal{C} \subseteq \{0,1\}^n$ is a subset; $\mathcal{C} \subseteq \{0,1\}^n$ such that
 $\min_{\mathcal{C}} \text{Hamming}(c; c') = d$ over all distinct $c, c' \in \mathcal{C}$. Let $n(c; d)$ denote the size of the largest such code.

Such a code corrects $\lfloor \frac{d-1}{2} \rfloor$ bitflip errors or $\lfloor \frac{d-1}{2} \rfloor$ erasure errors.

Basic Questions

(1) When n is fixed and $d \geq 1$, what is the order of $n(c; d)$?

Background: Bitflips and Erasures

Definition

A $\mathcal{C} \subseteq \{0,1\}^n$ is a subset; $\mathcal{C} \subseteq \{0,1\}^n$ such that
 $\min_{C \neq C'} \text{Hamming}(C; C') = d$ over all distinct $C, C' \in \mathcal{C}$. Let $N(n, d)$ denote the size of the largest such code.

Such a code corrects $\lfloor \frac{d-1}{2} \rfloor$ bitflip errors or $\lfloor \frac{d}{2} \rfloor$ erasure errors.

Basic Questions

- (1) When n is fixed and $d \geq 1$, what is the order of $N(n, d)$?
- (2) For which $d \in \{0,1\}$ is $N(n, d) \geq 2^{\epsilon n}$? A code with size $2^{\epsilon n}$ is called "positive rate."

Background: Bitflips and Erasures

Constant number of errors

When t is fixed and $n \rightarrow \infty$, what is the order of $\binom{n}{t}$?

Background: Bitflips and Erasures

Constant number of errors

When t is fixed and $n \gg 1$, what is the order of $\binom{n}{t}$?

BCH Codes (Hocquenghem '59, Bose and Ray-Chaudhuri '60)

For $t \geq 1$,

$$\binom{n}{t} = O(n^{b-\frac{1}{2}c}):$$

Background: Bitflips and Erasures

Constant number of errors

When b is fixed and $c \ll 1$, what is the order of $\binom{b}{c}$?

BCH Codes (Hocquenghem '59, Bose and Ray-Chaudhuri '60)

For $c \ll 1$,

$$\binom{b}{c} = (2 = b - \frac{1}{2}c):$$

Linear number of errors

For which c is $\binom{b}{c} \approx 2^c$?

Background: Bitflips and Erasures

Constant number of errors

When t is fixed and $n \rightarrow \infty$, what is the order of $V(n; t)$?

BCH Codes (Hocquenghem '59, Bose and Ray-Chaudhuri '60)

For $t \leq 1$,

$$V(n; t) = O(n^{2t-1})$$

Linear number of errors

For which $\delta \in (0; 1)$ is $V(n; \delta n) = O(n)$?

Theorem (Gilbert '52, Varshamov '57)

For $\delta \in (0; \frac{1}{2})$,

$$V(n; \delta n) = O(n^{\frac{1}{1-\delta}})$$

Background: Bitflips and Erasures

Constant number of errors

When ϵ is fixed and $n \rightarrow \infty$, what is the order of $A(n, \epsilon)$?

BCH Codes (Hocquenghem '59, Bose and Ray-Chaudhuri '60)

For $t \leq 1$,

$$A(n, \epsilon) = \Theta(2^{n - b - \frac{1}{2}c}):$$

Linear number of errors

For which $\epsilon < 1/2$ is $A(n, \epsilon) = \Theta(n)$?

Theorem (Gilbert '52, Varshamov '57)

For $\epsilon < 1/2$,

$$A(n, \epsilon) = \Theta(2^{n(1 - \epsilon)}):$$

Varshamov's Proof. For a uniform random $(1 - \epsilon)n \times n$ matrix r over F_2 , its rowspace is w.h.p. a code of distance ϵn . \square

Background: Bitflips and Erasures

Constant number of errors

When ϵ is fixed and $n \rightarrow \infty$, what is the order of $N(n, \epsilon)$?

BCH Codes (Hocquenghem '59, Bose and Ray-Chaudhuri '60)

For $t \leq 1$,

$$N(n, t) = 2^{n - b - \frac{1}{2}c}:$$

Linear number of errors

For which ϵ is $N(n, \epsilon) \sim 2^{\epsilon n}$?

Theorem (Gilbert '52, Varshamov '57)

For $\epsilon < \frac{1}{2}$,

$$N(n, \epsilon) \sim 2^{(1 - \epsilon)n}:$$

Varshamov's Proof. For a uniform random $(n \times n)$ matrix r over F_2 , its rowspace is w.h.p. a code of distance ϵn . □

(For $\epsilon > \frac{1}{2}$, $N(n, \epsilon) \sim 2^{\epsilon n}$.)

Deletion Codes

Worst-Case Deletion Errors

Definition

The binary ϵ -deletion channel takes in a string $x \in \{0,1\}^n$ and (adversarially) outputs a substring of length $(1-\epsilon)n$.

Worst-Case Deletion Errors

Definition

The binary ϵ -deletion channel takes in a string $x \in \{0,1\}^n$ and (adversarially) outputs a substring of length $(1-\epsilon)n$.

A subset S of length n is a subset $S \subseteq \{0,1\}^n$ such that $\text{LCS}(x, y) < (1-\epsilon)n$ for all distinct $x, y \in S$.

Worst-Case Deletion Errors

Definition

The binary ϵ -deletion channel takes in a string $x \in \{0,1\}^n$ and (adversarially) outputs a substring of length $(1-\epsilon)n$.

A set S of length n is a subset ϵ -deletion resilient if $|S \cap \{0,1\}^n| \geq (1-\epsilon)n$ for all distinct $x, y \in \{0,1\}^n$ such that $\text{LCS}(x, y) < (1-\epsilon)n$.

Comparing deletions to bitflip/erasure errors:

Worst-Case Deletion Errors

Definition

The binary ϵ -deletion channel takes in a string $x \in \{0,1\}^n$ and (adversarially) outputs a substring of length $(1-\epsilon)n$.

A code C of length n is a subset $S \subseteq \{0,1\}^n$ such that $\text{LCS}(x, y) < (1-\epsilon)n$ for all distinct $x, y \in S$.

Comparing deletions to bitflip/erasure errors:

- Deletion codes also correct bitflips and erasures.

Worst-Case Deletion Errors

Definition

The binary δ -deletion channel takes in a string $x \in \{0,1\}^n$ and (adversarially) outputs a substring of length $(1-\delta)n$.

A code C of length n is a subset $S \subseteq \{0,1\}^n$ such that $\text{LCS}(x, y) < (1-\delta)n$ for all distinct $x, y \in S$.

Comparing deletions to bitflip/erasure errors:

- Deletion codes also correct bitflips and erasures.
- Deletion errors are **not invariant under permutations**.

Worst-Case Deletion Errors

Definition

The binary ϵ -deletion channel takes in a string $x \in \{0,1\}^n$ and (adversarially) outputs a substring of length $(1-\epsilon)n$.

A code C of length n is a subset $S \subseteq \{0,1\}^n$ such that $\text{LCS}(x, y) < (1-\epsilon)n$ for all distinct $x, y \in S$.

Comparing deletions to bitflip/erasure errors:

- Deletion codes also correct bitflips and erasures.
- Deletion errors are **not invariant under permutations**.

Let G be the **confusability graph** on $\{0,1\}^n$ defined by $(x, y) \in G$ if $\text{LCS}(x, y) \geq (1-\epsilon)n$. A deletion code is just an independent set in G .

Deletion Codes

Definition

A \mathcal{C} of length n is a set $\{c_1, \dots, c_g\}$ such that $\text{LCS}(c_i, c_j) < n - 1$ for all distinct i, j .

Let $D(n, k)$ be the size of the largest k -deletion code of length n .

Deletion Codes

Definition

A (k, ℓ) -deletion code of length n is a set $C \subseteq \Sigma^n$ such that $\text{LCS}(x, y) \geq k$ for all distinct $x, y \in C$.

Let $D(n, k, \ell)$ be the size of the largest (k, ℓ) -deletion code of length n .

Ex. $C = \{f^k, g^k\}$ is an $(k, 1)$ -deletion code of length n and size 2.

Deletion Codes

Definition

A (k, ℓ) -deletion code of length n is a set $C \subseteq \Sigma^n$ such that $\text{LCS}(c, c') < k$ for all distinct $c, c' \in C$.

Let $D(n, k)$ be the size of the largest (k, ℓ) -deletion code of length n .

Ex. $C = \{f^k, g^k\}$ is an $(k-1)$ -deletion code of length k and size 2.

Ex (VT code '65). $C = \{f^k, g^k, f^{k-1}g, fg^{k-1}\} \cup \{f^k, g^k, f^{k-1}g^j \mid 0 \leq j < k \text{ (mod } \ell + 1)\}$ is a $(k-1)$ -deletion code of length k , and has size $(2 + \ell)$.

Deletion Codes

Definition

A (k, ℓ) -deletion code of length n is a set $C \subseteq \Sigma^n$ such that $\text{LCS}(c, c') < k$ for all distinct $c, c' \in C$.

Let $D(n, k)$ be the size of the largest (k, ℓ) -deletion code of length n .

Ex. $C = \{f^k, g^k\}$ is an $(k-1)$ -deletion code of length k and size 2.

Ex (VT code '65). $C = \{f^k, g^k, f^{k-1}g, fg^{k-1}\}$ is a 1-deletion code of length k , and has size $2 \binom{k}{\ell}$.

These turn out to be the only known optimal deletion codes!

Deletion Codes

Definition

A (k, ℓ) -deletion code of length n is a set $C \subseteq \Sigma^n$ such that $\text{LCS}(c, c') < k$ for all distinct $c, c' \in C$.

Let $D(n, k)$ be the size of the largest (k, ℓ) -deletion code of length n .

Ex. $C = \{f^k, g^k\}$ is an $(k-1)$ -deletion code of length k and size 2.

Ex (VT code '65). $C = \{f^k, g^k, f^{k-1}g, fg^{k-1}\}$ is a 1-deletion code of length k , and has size $2 \binom{k}{\ell}$.

These turn out to be the only known optimal deletion codes!

Theorem (Levenshtein '65)

For $k = 1$, $D(n, k) = 2 \binom{n}{\ell}$.

Deletion Codes

Definition

A (k, ℓ) -deletion code of length n is a set $C \subseteq \Sigma^n$ such that $\text{LCS}(x, y) \leq k$ for all distinct $x, y \in C$.

Let $D(n, k, \ell)$ be the size of the largest (k, ℓ) -deletion code of length n .

Ex. $C = \{f^k, g^k\}$ is an $(k-1, 0)$ -deletion code of length k and size 2.

Ex (VT code '65). $C = \{f^k, g^k, f^{k-1}g, fg^{k-1}\}$ is a $(k-1, 1)$ -deletion code of length k , and has size $2 \binom{k}{\ell}$.

These turn out to be the only known optimal deletion codes!

Theorem (Levenshtein '65)

For $\ell = 1$, $D(n, k, 1) = \binom{n}{k}$.

For $\ell \geq 2$,

$$\frac{2}{\ell} \leq D(n, k, \ell) \leq \frac{2}{\ell-1}.$$

Question

$Z(0;1)$ is $D(\cdot; \cdot)$ $2^{-\cdot}$?

That is, for which

High error rate

Question

$\mathcal{Z}(0;1)$ is $D(\cdot; \cdot) = 2^{-\cdot}$?

We call $\epsilon = \sup$ over all such the adversarial deletion channel.

That is, for which

of the

High error rate

Question

$D(0;1)$ is $D(\cdot; \cdot)$ 2^{-1} ?

That is, for which

We call $\epsilon = \sup$ over all such Δ the ϵ of the adversarial deletion channel.

Trivial upper bound: If $\epsilon \geq \frac{1}{2}$, $D(\cdot; \cdot) = 2$ because among any three strings, two share the same majority bit. Thus, $\epsilon \leq \frac{1}{2}$.

High error rate

Question

$D(0;1)$ is $D(x; y) = 2^{-LCS(x; y)}$?

That is, for which

We call $D(x; y) = \sup_{\Delta} \text{over all such } \Delta$ the $D(x; y)$ of the adversarial deletion channel.

Trivial upper bound: If $\Delta = \frac{1}{2}$, $D(x; y) = 2$ because among any three strings, two share the same majority bit. Thus, $D(x; y) \leq \frac{1}{2}$.

Theorem (Lueker '03)

If x and y are uniform random elements of $\{0,1\}^n$, then w.v.h.p. $D(x; y) \leq \frac{1}{2}$.

High error rate

Question

That is, for which

$D(0;1)$ is $D(\cdot; \cdot) = 2^{-\epsilon}$?

We call $\epsilon = \sup$ over all such δ the ϵ of the adversarial deletion channel.

Trivial upper bound: If $\epsilon > \frac{1}{2}$, $D(\cdot; \cdot) = 2$ because among any three strings, two share the same majority bit. Thus, $\epsilon \leq \frac{1}{2}$.

Theorem (Lueker '03)

If x and y are uniform random elements of $\{0,1\}^n$, then w.v.h.p. $\text{LCS}(x; y) \geq \frac{7}{8}n$. Thus, $\epsilon \geq \frac{1}{8}$ and uniform random codes can't correct more than $\frac{1}{8}n$ deletion errors.

High error rate

Question

That is, for which

$$D(0;1) \text{ is } D(\cdot; \cdot) \text{ } 2 \text{ } (\cdot)?$$

We call $D(\cdot; \cdot) = \sup$ over all such the of the adversarial deletion channel.

Trivial upper bound: If $D(\cdot; \cdot) = \frac{1}{2}$, $D(\cdot; \cdot) = 2$ because among any three strings, two share the same majority bit. Thus, $D(\cdot; \cdot) \leq \frac{1}{2}$.

Theorem (Lueker '03)

If x and y are uniform random elements of $\{0,1\}^n$, then w.v.h.p. $\text{LCS}(x, y) \geq \frac{1}{2}n$. Thus, $D(\cdot; \cdot) \leq \frac{1}{2}$ and uniform random codes can't correct more than $\frac{1}{2}$ deletion errors.

Theorem (Bukh, Guruswami, Håstad '16)

There exist explicit, efficient $(1 - \frac{1}{2})$ -deletion codes up to $n \leq \frac{1}{2} \log \frac{1}{\epsilon}$.

Constant number of errors.

For $\epsilon > 0$,

$$\frac{2}{\epsilon} D(\epsilon; \epsilon) \leq \frac{2}{\epsilon}.$$

Constant number of errors.

For $\epsilon > 2$,

$$\frac{2}{\epsilon} D(\epsilon; \epsilon) = \frac{2}{\epsilon}.$$

Linear number of errors.

The zero-rate threshold satisfies $\frac{P}{2} = 1 - \frac{1}{\epsilon}$.

Our Results

Constant number of errors. (Alon, Bourla, Graham, H., Kravitz '22)

For $\epsilon > 2$,

$$\frac{2 \log}{\epsilon} D(\epsilon; \epsilon) \leq \frac{2}{\epsilon}.$$

Linear number of errors. (Guruswami, H., Li '22)

The zero-rate threshold satisfies $\frac{P}{2} \leq 1 - \frac{1}{2} \cdot 10^{-60}$.

Constant Number of Deletions

Constant Number of Deletions

Theorem (Alon, Bourla, Graham, H., Kravitz '22)

If $\epsilon > 0$, then

$$D(\epsilon; \epsilon) = O\left(\frac{2 \log \epsilon}{\epsilon}\right).$$

Constant Number of Deletions

Theorem (Alon, Bourla, Graham, H., Kravitz '22)

If $\epsilon > 2$, then

$$D(G; \epsilon) \leq \frac{2 \log n}{\epsilon}.$$

Observe that $D(G; \epsilon) = \alpha(G; \epsilon)$, the size of the largest independent set in G . This graph has $n = 2^k$ vertices and max degree $\Delta = 2^{k-2}$.

Constant Number of Deletions

Theorem (Alon, Bourla, Graham, H., Kravitz '22)

If $\Delta \geq 2$, then

$$D(G; \Delta) \leq \frac{2 \log \Delta}{\Delta}.$$

Observe that $D(G; \Delta) = \alpha(G; \Delta)$, the size of the largest independent set in G . This graph has $n = 2^{\Delta}$ vertices and max degree $\Delta = 2^{\Delta-1} - 2^{\Delta-2}$.

Lemma (Ajtai, Komlós, Szemerédi '80, Shearer '83)

If G is a triangle-free graph with n vertices, maximum degree Δ , and $O(n \Delta^2)$ triangles, then

$$\alpha(G) \geq \frac{n \log \Delta}{\Delta}.$$

Constant Number of Deletions

Theorem (Alon, Bourla, Graham, H., Kravitz '22)

If $\Delta \geq 2$, then

$$D(n, \Delta) \leq \frac{2 \log n}{\Delta}.$$

Observe that $D(n, \Delta) = \alpha(n, \Delta)$, the size of the largest independent set in $G(n, \Delta)$. This graph has $n = 2^k$ vertices and max degree $\Delta = 2^{k-2}$.

Lemma (Ajtai, Komlós, Szemerédi '80, Shearer '83)

If G is a triangle-free graph with n vertices, maximum degree Δ , and $\leq C(\Delta^2)$ triangles, then

$$\alpha(G) \geq \frac{n \log \Delta}{\Delta}.$$

Thus, it suffices to show that the number of triangles in $G(n, \Delta)$ is $\leq C(\Delta^2)$.

Counting Triangles

Lemma.

The number of triangles in G ; is $\mathcal{O}(n^3 \log n)$.

Counting Triangles

Lemma.

The number of triangles in G is $\Theta(n^3 \log^{-3} n)$.

Proof Sketch ($d = 1$)

Let v be a uniform random element of $(V, E) = (n, 1/2n)$, and let u, w be two i.i.d. uniform random neighbors of v . We show that

$\Pr[u, w \text{ are neighbors}] = \Theta(\log^{-2} n)$.

Counting Triangles

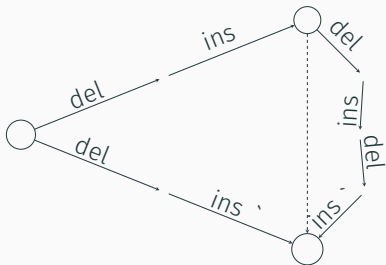
Lemma.

The number of triangles in G is $\Theta(n^3 \log^{-3} n)$.

Proof Sketch ($d = 1$)

Let v be a uniform random element of $(V, E) = f(0; 1)g$, and let u, w be two i.i.d. uniform random neighbors of v . We show that

$\Pr[u, w \text{ are neighbors}] = \Theta(\log^{-2} n)$.



Counting Triangles

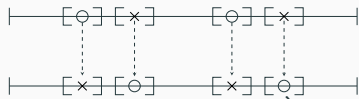
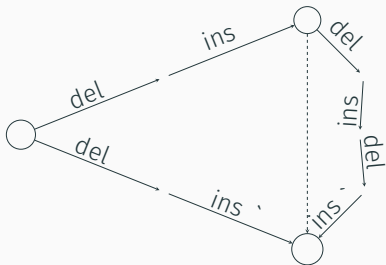
Lemma.

The number of triangles in G is $\Theta(n^3 \log n)$.

Proof Sketch ($d = 1$)

Let v be a uniform random element of $(V, E) = f(0; 1)g$, and let u, w be two i.i.d. uniform random neighbors of v . We show that

$\Pr[u, w \text{ are neighbors}] = \Theta(\log n)$.



Constant number of errors. (Alon, Bourla, Graham, H., Kravitz '22)

For $\epsilon > 0$,

$$\frac{2 \log \frac{1}{\epsilon}}{\epsilon} D(\epsilon; \epsilon) \leq \frac{2}{\epsilon}.$$

Bounds

Constant number of errors. (Alon, Bourla, Graham, H., Kravitz '22)

For $\epsilon > 2$,

$$\frac{2 \log}{\epsilon} D(\epsilon; \epsilon) = \frac{2}{\epsilon}:$$

- First order-of-growth improvement to Levenshtein's original bounds.

Bounds

Constant number of errors. (Alon, Bourla, Graham, H., Kravitz '22)

For $\epsilon > 2$,

$$\frac{2 \log}{\epsilon} D(n, \epsilon) \leq \frac{2}{\epsilon} n$$

- First order-of-growth improvement to Levenshtein's original bounds.
- Same technique was used by Jiang and Vardy '04 to improve the Gilbert-Varshamov bound for bitflip errors by a logarithmic factor.

Bounds

Constant number of errors. (Alon, Bourla, Graham, H., Kravitz '22)

For $\epsilon > 2$,

$$\frac{2 \log \epsilon}{\epsilon} D(n; \epsilon) \leq \frac{2}{\epsilon} n$$

- First order-of-growth improvement to Levenshtein's original bounds.
- Same technique was used by Jiang and Vardy '04 to improve the Gilbert-Varshamov bound for bitflip errors by a logarithmic factor.
- Uses a **strong pseudorandomness property** of random strings $x \in \{0,1\}^n$: every $\log \epsilon$ -length subinterval is unique.

Linear Number of Deletions

Linear Number of Deletions

Theorem (Guruswami, H., Li '22)

If $\frac{1}{2} \leq \epsilon \leq 10^{-60}$, then $D(\epsilon) \leq 2(\log \frac{1}{\epsilon})^{10^{60}}$. Thus $D(\epsilon) \leq \frac{1}{2} \leq 10^{-60}$.

Linear Number of Deletions

Theorem (Guruswami, H., Li '22)

If $\frac{1}{2} \leq \epsilon \leq 10^{-60}$, then $D(\epsilon; \epsilon) \leq 2(\log \frac{1}{\epsilon})^{10^{60}}$. Thus $\frac{1}{2} \leq \epsilon \leq 10^{-60}$.

Equivalently, among any $2(\log \frac{1}{\epsilon})^{10^{60}}$ strings in $\{0,1\}^{\epsilon}$, some two satisfy $\text{LCS}(x, y) \leq (\frac{1}{2} + \epsilon^{60}) \cdot \epsilon$.

Linear Number of Deletions

Theorem (Guruswami, H., Li '22)

If $\frac{1}{2} \leq \epsilon \leq 10^{-60}$, then $D(x, y) \leq 2(\log \frac{1}{\epsilon})^{10^{60}}$. Thus $\frac{1}{2} \leq \epsilon \leq 10^{-60}$.

Equivalently, among any $2^{(\log \frac{1}{\epsilon})^{10^{60}}}$ strings in $\{0,1\}^g$, some two satisfy $LCS(x, y) \geq (\frac{1}{2} + \epsilon)g$.

Proof Strategy

Classify strings according to how much they “look like” $1^k 0^k 1^k 0^k$ for each power of two k . A crude analogy is assigning **log** “Fourier coefficients” to x that measure its oscillation on each scale.

Pigeonhole to find x, y with the same oscillation statistics. This guarantees $LCS(x, y)$ is large for three possible reasons:

- (1) If x, y oscillate at a large scale $k = \Omega(g)$.
- (2) If x, y share at least one “large Fourier coefficient” at the same scale.
- (3) If x, y share many “small Fourier coefficients” at different scales.

Matching Strategies

Case 1: Imbalanced Strings



Matching Strategies

Case 1: Imbalanced Strings



Case 2: Single-Frequency Strings



Flagged bits	
Blue	(1) 0:99
Green	(1) 0:6
Yellow	(1) 0:49

Matching Strategies

Case 1: Imbalanced Strings



Case 2: Single-Frequency Strings

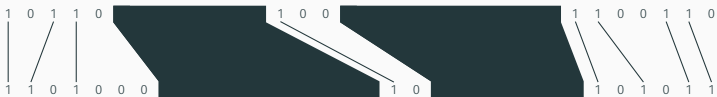


Case 3: Many-Frequency Strings



Flagged bits	
Blue	(\neq) 0:99
Green	(\neq) 0:6
Yellow	(\neq) 0:49

Case 3: Many-Frequency Strings

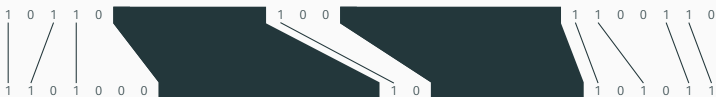


Case 3: Many-Frequency Strings



Blue and yellow regions are adjusted to have the same number of zeros, but different numbers of ones, so our two pointers can get misaligned!

Case 3: Many-Frequency Strings



Blue and yellow regions are adjusted to have the same number of zeros, but different numbers of ones, so our two pointers can get misaligned!

The regularity method comes to the rescue!

The Regularity Method

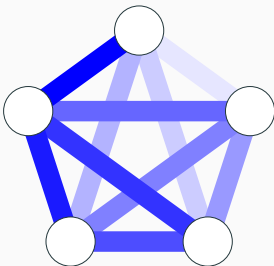
Theorem (Szemerédi '78)

For every $\epsilon > 0$, all sufficiently large graphs G can be partitioned into $O(1)$ (nearly-)equal-sized vertex sets such that all but an ϵ -fraction of these pairs are ϵ -regular.

The Regularity Method

Theorem (Szemerédi '78)

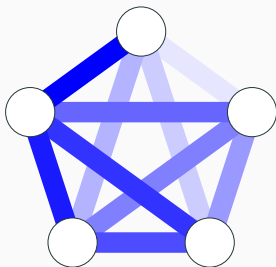
For every $\epsilon > 0$, all sufficiently large graphs G can be partitioned into $O(\epsilon^{-1})$ (nearly-)equal-sized vertex sets such that all but an ϵ -fraction of these pairs are ϵ -regular.



The Regularity Method

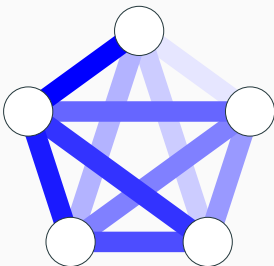
Theorem (Szemerédi '78)

For every $\epsilon > 0$, all sufficiently large graphs G can be partitioned into $O(1/\epsilon^2)$ (nearly-)equal-sized vertex sets such that all but an ϵ -fraction of these pairs are ϵ -regular.



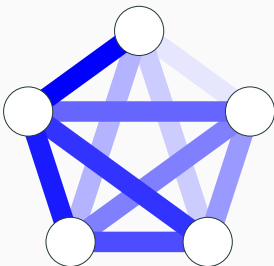
A pair of vertex sets A and B are ϵ -regular if for all subsets $A' \subseteq A$ and $B' \subseteq B$ satisfying $|A'| \geq \epsilon|A|$ and $|B'| \geq \epsilon|B|$, we have $|e(A', B') - \epsilon|A||B|| < \epsilon^2|A||B|$.

The Regularity Method



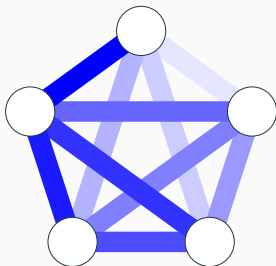
- Shows that for a **weak pseudorandomness property**, every graph can be nearly partitioned into pseudorandom parts.

The Regularity Method



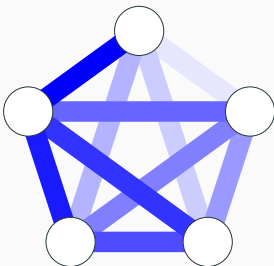
- Shows that for a **weak pseudorandomness property**, every graph can be nearly partitioned into pseudorandom parts.
- Useful for counting and embedding subgraphs.

The Regularity Method



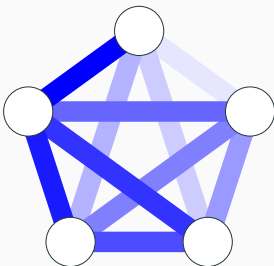
- Shows that for a **weak pseudorandomness property**, every graph can be nearly partitioned into pseudorandom parts.
- Useful for counting and embedding subgraphs.
- Numerous applications in extremal and additive combinatorics, for example Szemerédi's Theorem on arithmetic progressions.

The Regularity Method



- Shows that for a **weak pseudorandomness property**, every graph can be nearly partitioned into pseudorandom parts.
- Useful for counting and embedding subgraphs.
- Numerous applications in extremal and additive combinatorics, for example Szemerédi's Theorem on arithmetic progressions.
- Connections to dynamics starting from the work of Furstenberg.

The Regularity Method



- Shows that for a **weak pseudorandomness property**, every graph can be nearly partitioned into pseudorandom parts.
- Useful for counting and embedding subgraphs.
- Numerous applications in extremal and additive combinatorics, for example Szemerédi's Theorem on arithmetic progressions.
- Connections to dynamics starting from the work of Furstenberg.
- Notorious for horrible quantitative bounds.

Regularity Lemmas for Binary Strings

Lemma (Axenovich, Person, Puzynina '12)

For every $\epsilon > 0$ all sufficiently long binary strings can be partitioned into $2^{1/\epsilon}$ (nearly-)equal-sized subintervals such that all but an ϵ -fraction of these subintervals are ϵ -regular.

Regularity Lemmas for Binary Strings

Lemma (Axenovich, Person, Puzynina '12)

For every $\epsilon > 0$ all sufficiently long binary strings can be partitioned into $2^{\epsilon n}$ (nearly-)equal-sized subintervals such that all but an ϵ -fraction of these subintervals are ϵ -regular.

A string is ϵ -regular if for every subinterval I of length at least ϵn , we have $|I \cap \{0\}| \in [j, (j + \epsilon)n]$:

Regularity Lemmas for Binary Strings

Lemma (Axenovich, Person, Puzynina '12)

For every $\epsilon > 0$ all sufficiently long binary strings can be partitioned into $2^{\lfloor \epsilon n \rfloor}$ (nearly-)equal-sized subintervals such that all but an ϵ -fraction of these subintervals are ϵ -regular.

A string is ϵ -regular if for every subinterval I of length at least $\epsilon |I|$, we have $|I \cap \{1\}| - \epsilon |I| < \epsilon |I|$:

We prove a stronger regularity lemma which states that the distribution of the blue intervals of length ϵ is regular at every dyadic scale ϵ^k simultaneously.



Regularity Lemmas for Binary Strings

Lemma (Axenovich, Person, Puzynina '12)

For every $\epsilon > 0$ all sufficiently long binary strings can be partitioned into 2^j (nearly-)equal-sized subintervals such that all but an ϵ -fraction of these subintervals are ϵ -regular.

A string is ϵ -regular if for every subinterval I of length at least ϵ^j , we have $|j(I) - \binom{j}{\epsilon}| < \epsilon$:

We prove a stronger regularity lemma which states that the distribution of the blue intervals of length ϵ^j is regular at every dyadic scale ϵ^k simultaneously.



Linear number of errors. (Guruswami, H., Li '22)

The zero-rate threshold satisfies $P_{\bar{2}} \approx 1 - \frac{1}{2} \cdot 10^{-60}$.

Linear number of errors. (Guruswami, H., Li '22)

The zero-rate threshold satisfies $P_{\bar{2}} \approx 1 - \frac{1}{2} \cdot 10^{-60}$.

- First application (to our knowledge) of the regularity method for strings to coding theory.

Linear number of errors. (Guruswami, H., Li '22)

The zero-rate threshold satisfies $P \geq \frac{1}{2} - \frac{1}{2} \cdot 10^{-60}$.

- First application (to our knowledge) of the regularity method for strings to coding theory.
- It would be interesting to determine $D(\epsilon; \delta)$ for $\epsilon = 1/2 - \delta$. The best known bounds are now

$$\log D(\epsilon; \delta) \leq 2^{(\log \frac{1}{\delta})^{1060}};$$

Linear number of errors. (Guruswami, H., Li '22)

The zero-rate threshold satisfies $P_{\bar{2}} \approx 1 - \frac{1}{2} \cdot 10^{-60}$.

- First application (to our knowledge) of the regularity method for strings to coding theory.
- It would be interesting to determine $D(\epsilon; \delta)$ for $\epsilon = 1/2 - \delta$. The best known bounds are now

$$\log \epsilon \approx D(\epsilon; \delta) \approx 2^{(\log \epsilon)^{10^{60}}};$$

- Fuzzy question: graph regularity leads to graphons. Is there a useful theory of the limit objects coming from string regularity?

Epilogue

An Algorithms Connection

Computing LCS of two binary strings can be done in quadratic time using dynamic programming, and this is best possible (up to logarithms) under certain complexity theory hypotheses.

An Algorithms Connection

Computing LCS of two binary strings can be done in quadratic time using dynamic programming, and this is best possible (up to logarithms) under certain complexity theory hypotheses.

What about approximation?

An Algorithms Connection

Computing LCS of two binary strings can be done in quadratic time using dynamic programming, and this is best possible (up to logarithms) under certain complexity theory hypotheses.

What about approximation?

There exists a trivial $1/2$ -approximation algorithm in linear time: pick the longest constant common subsequence.

An Algorithms Connection

Computing LCS of two binary strings can be done in quadratic time using dynamic programming, and this is best possible (up to logarithms) under certain complexity theory hypotheses.

What about approximation?

There exists a trivial $1/2$ -approximation algorithm in linear time: pick the longest constant common subsequence.

Morally, this is the $1/2$ barrier that appears in the deletion codes problem!

An Algorithms Connection

Computing LCS of two binary strings can be done in quadratic time using dynamic programming, and this is best possible (up to logarithms) under certain complexity theory hypotheses.

What about approximation?

There exists a trivial $1/2$ -approximation algorithm in linear time: pick the longest constant common subsequence.

Morally, this is the $1/2$ barrier that appears in the deletion codes problem!

Theorem (H., Li '23, building on Rubinfeld, Song '20)

For all $\epsilon > 0$, there exists $\delta > 0$ and a $O(n^{1+\delta})$ -time algorithm which gives a $(\frac{1}{2} + \epsilon)$ -approximation for the LCS of two binary strings.

An Algorithms Connection

Computing LCS of two binary strings can be done in quadratic time using dynamic programming, and this is best possible (up to logarithms) under certain complexity theory hypotheses.

What about approximation?

There exists a trivial $1/2$ -approximation algorithm in linear time: pick the longest constant common subsequence.

Morally, this is the $1/2$ barrier that appears in the deletion codes problem!

Theorem (H., Li '23, building on Rubinfeld, Song '20)

For all $\epsilon > 0$, there exists $\delta > 0$ and a $O(n^{1+\delta})$ -time algorithm which gives a $(\frac{1}{2} + \epsilon)$ -approximation for the LCS of two binary strings.

Uses the same “oscillation statistics” machinery.