Deletion Code Bounds via Pseudorandomness

Xiaoyu He (Princeton University)

February 10, 2023

Setting the Stage

The Beginning

Noisy Channel Encoding Theorem (Shannon ’48)

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

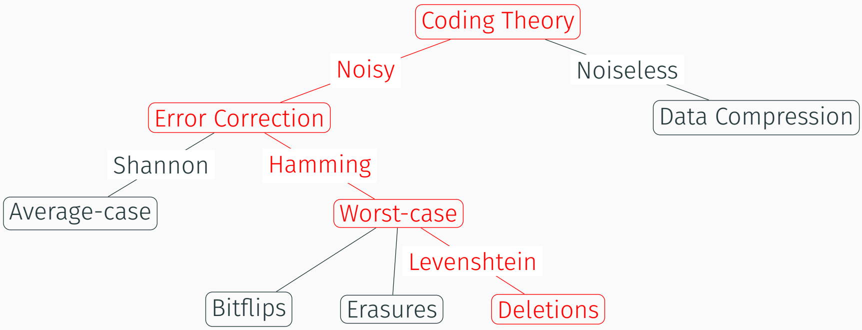
Noisy Channel Encoding Theorem (Shannon '48)

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

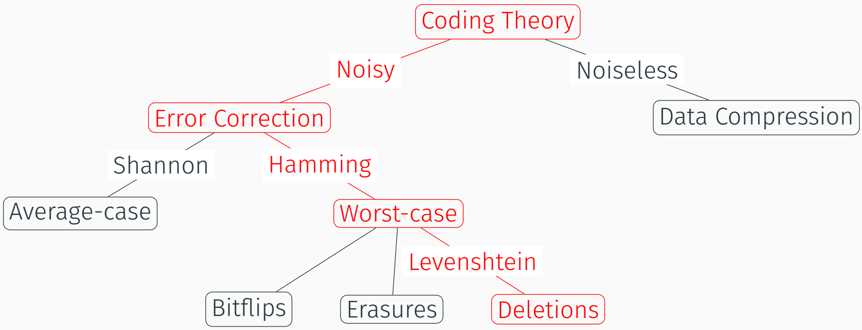
He wants to create a method of coding, but he doesn't know what to do so he makes a random code. Then he is stuck. ***And then he asks the impossible question, "What would the average random code do?"*** *He then proves that the average code is arbitrarily good, and that therefore there must be at least one good code. Who but a man of infinite courage could have dared to think those thoughts?*

— Richard Hamming, on Claude Shannon.

Phylogeny of Coding Theory

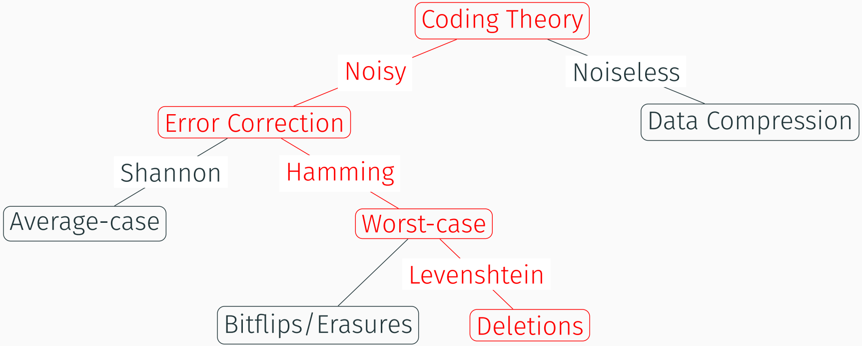


Phylogeny of Coding Theory



Everything is binary!

Phylogeny of coding theory



Everything is binary!

Noise Models

Noise Models

Noise models

Electromagnetic signal

Bitflip errors 1101 $\rightarrow$ 1001

Noise Models

Noise models	
Electromagnetic signal	Bitflip errors 1101 $\to$ 1001
Auditory experience	Erasure errors 1101 $\to$ 1?01

Noise models

Electromagnetic signal	Bitflip errors $1101 \to 1001$
Auditory experience	Erasure errors $1101 \to 1?01$
Transcribed lyrics	Deletion errors $1101 \to 101$

Background: Bitflips and Erasures

Definition

A *code of length n and distance d* is a subset $C \subseteq \{0,1\}^n$ such that $\min d_{\text{Hamming}}(s,t) = d$ over all distinct $s, t \in C$. Let $A(n,d)$ denote the size of the largest such code.

Background: Bitflips and Erasures

Definition

A *code of length n and distance d* is a subset $C \subseteq \{0,1\}^n$ such that $\min d_{\text{Hamming}}(s,t) = d$ over all distinct $s,t \in C$. Let $A(n,d)$ denote the size of the largest such code.

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

Background: Bitflips and Erasures

Definition

A *code of length n and distance d* is a subset $C \subseteq \{0,1\}^n$ such that $\min d_{\text{Hamming}}(s,t) = d$ over all distinct $s,t \in C$. Let $A(n,d)$ denote the size of the largest such code.

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

Basic Questions

(1) When d is fixed and $n \to \infty$, what is the order of $A(n,d)$?

Background: Bitflips and Erasures

Definition

A *code of length n and distance d* is a subset $C \subseteq \{0,1\}^n$ such that $\min d_{\text{Hamming}}(s,t) = d$ over all distinct $s, t \in C$. Let $A(n,d)$ denote the size of the largest such code.

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

Basic Questions

(1) When d is fixed and $n \to \infty$, what is the order of $A(n,d)$?

(2) For which $p \in (0,1)$ is $A(n,pn) \geq 2^{\Omega(n)}$? A code with size $2^{\Omega(n)}$ is called "positive rate."

Background: Bitflips and Erasures

Constant number of errors

When d is fixed and $n \to \infty$, what is the order of $A(n, d)$?

Background: Bitflips and Erasures

Constant number of errors

When d is fixed and $n \to \infty$, what is the order of $A(n,d)$?

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

For $d \geq 1$,

$$A(n,d) = \Theta(2^n / n^{\lfloor \frac{d-1}{2} \rfloor}).$$

Background: Bitflips and Erasures

Constant number of errors

When d is fixed and $n \to \infty$, what is the order of $A(n, d)$?

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

For $d \geq 1$,

$$A(n, d) = \Theta(2^n / n^{\lfloor \frac{d-1}{2} \rfloor}).$$

Linear number of errors

For which $p \in (0, 1)$ is $A(n, pn) \geq 2^{\Omega(n)}$?

Background: Bitflips and Erasures

Constant number of errors

When d is fixed and $n \to \infty$, what is the order of $A(n, d)$?

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

For $d \geq 1$,

$$A(n, d) = \Theta(2^n / n^{\lfloor \frac{d-1}{2} \rfloor}).$$

Linear number of errors

For which $p \in (0, 1)$ is $A(n, pn) \geq 2^{\Omega(n)}$?

Theorem (Gilbert '52, Varshamov '57)

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

$$A(n, pn) \geq 2^{(1-H(p)-o(1))n}.$$

Background: Bitflips and Erasures

Constant number of errors

When d is fixed and $n \to \infty$, what is the order of $A(n, d)$?

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

For $d \geq 1$,

$$A(n, d) = \Theta(2^n / n^{\lfloor \frac{d-1}{2} \rfloor}).$$

Linear number of errors

For which $p \in (0, 1)$ is $A(n, pn) \geq 2^{\Omega(n)}$?

Theorem (Gilbert '52, Varshamov '57)

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

$$A(n, pn) \geq 2^{(1-H(p)-o(1))n}.$$

Varshamov's Proof. For a uniform random $(1 - H(p) - \varepsilon)n \times n$ matrix G over $\mathbb{F}_2$, its rowspace is w.h.p. a code of distance pn. □

Background: Bitflips and Erasures

Constant number of errors

When d is fixed and $n \to \infty$, what is the order of $A(n, d)$?

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

For $d \geq 1$,

$$A(n, d) = \Theta(2^n / n^{\lfloor \frac{d-1}{2} \rfloor}).$$

Linear number of errors

For which $p \in (0, 1)$ is $A(n, pn) \geq 2^{\Omega(n)}$?

Theorem (Gilbert '52, Varshamov '57)

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

$$A(n, pn) \geq 2^{(1-H(p)-o(1))n}.$$

Varshamov's Proof. For a uniform random $(1 - H(p) - \varepsilon)n \times n$ matrix G over $\mathbb{F}_2$, its rowspace is w.h.p. a code of distance pn. $\square$

(For $p \geq \frac{1}{2}$, $A(n, pn) \leq 2n$.)

Deletion Codes

Worst-Case Deletion Errors

Definition

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

Worst-Case Deletion Errors

Definition

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

A *binary d-deletion code* of length n is a subset $C \subseteq \{0,1\}^n$ such that $\mathrm{LCS}(s,t) < n - d$ for all distinct $s, t \in C$.

Worst-Case Deletion Errors

Definition

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

A *binary d-deletion code* of length n is a subset $C \subseteq \{0,1\}^n$ such that $\mathrm{LCS}(s,t) < n - d$ for all distinct $s, t \in C$.

Comparing deletions to bitflip/erasure errors:

Worst-Case Deletion Errors

Definition

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

A *binary d-deletion code* of length n is a subset $C \subseteq \{0,1\}^n$ such that $\mathrm{LCS}(s,t) < n - d$ for all distinct $s, t \in C$.

Comparing deletions to bitflip/erasure errors:

- Deletion codes also correct bitflips and erasures.

Worst-Case Deletion Errors

Definition

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

A *binary d-deletion code* of length n is a subset $C \subseteq \{0,1\}^n$ such that $\text{LCS}(s,t) < n - d$ for all distinct $s, t \in C$.

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 d-deletion channel takes in a string $s \in \{0,1\}^n$ and (adversarially) outputs a substring of length $n - d$.

A *binary d-deletion code* of length n is a subset $C \subseteq \{0,1\}^n$ such that $\mathrm{LCS}(s,t) < n - d$ for all distinct $s, t \in C$.

Comparing deletions to bitflip/erasure errors:

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

Let $\Gamma_{n,d}$ be the **confusability graph** on $\{0,1\}^n$ defined by $s \sim t$ if $\mathrm{LCS}(s,t) \geq n - d$. A deletion code is just an independent set in $\Gamma_{n,d}$.

Deletion Codes

Definition

A *d-deletion code* of length n is a set $C \subseteq \{0,1\}^n$ such that $\mathrm{LCS}(s,t) < n - d$ for all distinct $s, t \in C$.

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

Deletion Codes

Definition

A *d-deletion code* of length n is a set $C \subseteq \{0,1\}^n$ such that $\mathrm{LCS}(s,t) < n - d$ for all distinct $s,t \in C$.

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

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

Deletion Codes

Definition

A *d-deletion code* of length n is a set $C \subseteq \{0,1\}^n$ such that $\mathrm{LCS}(s,t) < n - d$ for all distinct $s, t \in C$.

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

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

Ex (VT code '65). $C = \{s \in \{0,1\}^n | \sum i s_i \equiv 0 \pmod{n+1}\}$ is a 1-deletion code of length n, and has size $\Theta(2^n/n)$.

Deletion Codes

Definition

A *d-deletion code* of length n is a set $C \subseteq \{0,1\}^n$ such that $\mathrm{LCS}(s,t) < n - d$ for all distinct $s, t \in C$.

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

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

Ex (VT code '65). $C = \{s \in \{0,1\}^n \mid \sum i s_i \equiv 0 \pmod{n+1}\}$ is a 1-deletion code of length n, and has size $\Theta(2^n/n)$.

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

Deletion Codes

Definition

A *d-deletion code* of length n is a set $C \subseteq \{0,1\}^n$ such that $\mathrm{LCS}(s,t) < n - d$ for all distinct $s, t \in C$.

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

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

Ex (VT code '65). $C = \{s \in \{0,1\}^n | \sum i s_i \equiv 0 \pmod{n+1}\}$ is a 1-deletion code of length n, and has size $\Theta(2^n/n)$.

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

Theorem (Levenshtein '65)

For $d = 1$, $D(n,1) = \Theta(2^n/n)$.

Deletion Codes

Definition

A *d-deletion code* of length n is a set $C \subseteq \{0,1\}^n$ such that $\mathrm{LCS}(s,t) < n - d$ for all distinct $s, t \in C$.

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

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

Ex (VT code '65). $C = \{s \in \{0,1\}^n \mid \sum i s_i \equiv 0 \pmod{n+1}\}$ is a 1-deletion code of length n, and has size $\Theta(2^n/n)$.

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

Theorem (Levenshtein '65)

For $d = 1$, $D(n,1) = \Theta(2^n/n)$.

For $d \geq 2$,

$$\frac{2^n}{n^{2d}} \ll_d D(n,d) \ll_d \frac{2^n}{n^d}.$$

High error rate

Question

When is positive information rate achievable? That is, for which $p \in (0,1)$ is $D(n, pn) \geq 2^{\Omega(n)}$?

High error rate

Question

When is positive information rate achievable? That is, for which $p \in (0,1)$ is $D(n, pn) \geq 2^{\Omega(n)}$?

We call $p^* = \sup p$ over all such p the *zero-rate threshold* of the adversarial deletion channel.

High error rate

Question

When is positive information rate achievable? That is, for which $p \in (0,1)$ is $D(n, pn) \geq 2^{\Omega(n)}$?

We call $p^* = \sup p$ over all such p the *zero-rate threshold* of the adversarial deletion channel.

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

High error rate

Question

When is positive information rate achievable? That is, for which $p \in (0,1)$ is $D(n, pn) \geq 2^{\Omega(n)}$?

We call $p^* = \sup p$ over all such p the *zero-rate threshold* of the adversarial deletion channel.

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

Theorem (Lueker '03)

If s and t are uniform random elements of $\{0,1\}^n$, then w.v.h.p. $.78n \leq \mathrm{LCS}(s,t) \leq .82n$.

High error rate

Question

When is positive information rate achievable? That is, for which $p \in (0,1)$ is $D(n, pn) \geq 2^{\Omega(n)}$?

We call $p^* = \sup p$ over all such p the *zero-rate threshold* of the adversarial deletion channel.

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

Theorem (Lueker '03)

If s and t are uniform random elements of $\{0,1\}^n$, then w.v.h.p. $.78n \leq \text{LCS}(s,t) \leq .82n$. Thus, $p^* \geq .18$ and uniform random codes can't correct more than $.22n$ deletion errors.

High error rate

Question

When is positive information rate achievable? That is, for which $p \in (0,1)$ is $D(n, pn) \geq 2^{\Omega(n)}$?

We call $p^* = \sup p$ over all such p the *zero-rate threshold* of the adversarial deletion channel.

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

Theorem (Lueker '03)

If s and t are uniform random elements of $\{0,1\}^n$, then w.v.h.p. $.78n \leq \mathrm{LCS}(s,t) \leq .82n$. Thus, $p^* \geq .18$ and uniform random codes can't correct more than $.22n$ deletion errors.

Theorem (Bukh, Guruswami, Håstad '16)

There exist explicit, efficient pn-deletion codes up to $p^* \geq \sqrt{2} - 1 \approx .414$.

Bounds

Constant number of errors.

For $d \geq 2$,

$$\frac{2^n}{n^{2d}} \ll_d D(n,d) \ll_d \frac{2^n}{n^d}.$$

Bounds

Constant number of errors.

For $d \geq 2$,

$$\frac{2^n}{n^{2d}} \ll_d D(n,d) \ll_d \frac{2^n}{n^d}.$$

Linear number of errors.

The zero-rate threshold satisfies $\sqrt{2} - 1 \leq p^* \leq \frac{1}{2}$.

Our Results

Bounds

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

For $d \geq 2$,

$$\frac{2^n \log n}{n^{2d}} \ll_d D(n,d) \ll_d \frac{2^n}{n^d}.$$

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

The zero-rate threshold satisfies $\sqrt{2}-1 \leq p^* \leq \frac{1}{2} - 10^{-60}$.

Constant Number of Deletions

Constant Number of Deletions

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

If $d \geq 2$, then

$$D(n, d) \gg_d \frac{2^n \log n}{n^{2d}}.$$

Constant Number of Deletions

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

If $d \geq 2$, then

$$D(n, d) \gg_d \frac{2^n \log n}{n^{2d}}.$$

Observe that $D(n, d) = \alpha(\Gamma_{n,d})$, the size of the largest independent set in $\Gamma_{n,d}$. This graph has $N = 2^n$ vertices and max degree $\Delta = 2^d \binom{n}{d}^2 \leq n^{2d}$.

Constant Number of Deletions

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

If $d \geq 2$, then

$$D(n, d) \gg_d \frac{2^n \log n}{n^{2d}}.$$

Observe that $D(n, d) = \alpha(\Gamma_{n,d})$, the size of the largest independent set in $\Gamma_{n,d}$. This graph has $N = 2^n$ vertices and max degree $\Delta = 2^d \binom{n}{d}^2 \leq n^{2d}$.

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

If Γ is a graph with N vertices, maximum degree Δ, and $O(ND^{2-\varepsilon})$ triangles, then

$$\alpha(\Gamma) \gg \frac{N \log \Delta}{\Delta}.$$

Constant Number of Deletions

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

If $d \geq 2$, then

$$D(n, d) \gg_d \frac{2^n \log n}{n^{2d}}.$$

Observe that $D(n, d) = \alpha(\Gamma_{n,d})$, the size of the largest independent set in $\Gamma_{n,d}$. This graph has $N = 2^n$ vertices and max degree $\Delta = 2^d \binom{n}{d}^2 \leq n^{2d}$.

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

If Γ is a graph with N vertices, maximum degree Δ, and $O(ND^{2-\varepsilon})$ triangles, then

$$\alpha(\Gamma) \gg \frac{N \log \Delta}{\Delta}.$$

Thus, it suffices to show that the number of triangles in $\Gamma_{n,d}$ is $O(2^n n^{4d-\varepsilon})$.

Counting Triangles

Lemma.

The number of triangles in $\Gamma_{n,d}$ is $O(2^n n^{3d} \log^d n)$.

Counting Triangles

Lemma.

The number of triangles in $\Gamma_{n,d}$ is $O(2^n n^{3d} \log^d n)$.

Proof Sketch ($d = 1$)

Let u be a uniform random element of $V(\Gamma_{n,1}) = \{0,1\}^n$, and let v, w be two i.i.d. uniform random neighbors of u. We show that $\Pr[v \sim w] = O(\log n / n)$.

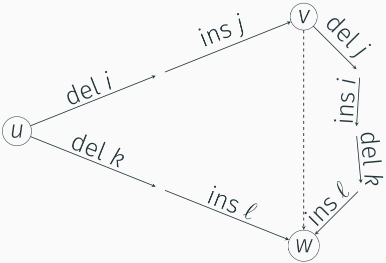
Counting Triangles

Lemma.

The number of triangles in $\Gamma_{n,d}$ is $O(2^n n^{3d} \log^d n)$.

Proof Sketch ($d = 1$)

Let u be a uniform random element of $V(\Gamma_{n,1}) = \{0,1\}^n$, and let v, w be two i.i.d. uniform random neighbors of u. We show that $\Pr[v \sim w] = O(\log n / n)$.



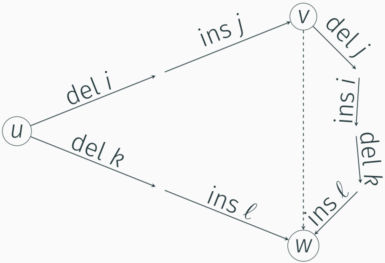
Counting Triangles

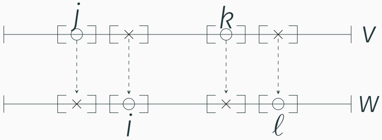
Lemma.

The number of triangles in $\Gamma_{n,d}$ is $O(2^n n^{3d} \log^d n)$.

Proof Sketch ($d = 1$)

Let u be a uniform random element of $V(\Gamma_{n,1}) = \{0,1\}^n$, and let v, w be two i.i.d. uniform random neighbors of u. We show that $\Pr[v \sim w] = O(\log n / n)$.





Bounds

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

For $d \geq 2$,

$$\frac{2^n \log n}{n^{2d}} \ll_d D(n, d) \ll_d \frac{2^n}{n^d}.$$

Bounds

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

For $d \geq 2$,

$$\frac{2^n \log n}{n^{2d}} \ll_d D(n,d) \ll_d \frac{2^n}{n^d}.$$

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

Bounds

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

For $d \geq 2$,

$$\frac{2^n \log n}{n^{2d}} \ll_d D(n,d) \ll_d \frac{2^n}{n^d}.$$

- 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 $d \geq 2$,

$$\frac{2^n \log n}{n^{2d}} \ll_d D(n,d) \ll_d \frac{2^n}{n^d}.$$

- 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 u, v, w: every $\log n$-length subinterval is unique.

Linear Number of Deletions

Linear Number of Deletions

Theorem (Guruswami, H., Li '22)

If $p \geq \frac{1}{2} - 10^{-60}$, then $D(n, pn) \leq 2^{(\log n)^{10^{60}}}$. Thus $p^* \leq \frac{1}{2} - 10^{-60}$.

Linear Number of Deletions

Theorem (Guruswami, H., Li '22)

If $p \geq \frac{1}{2} - 10^{-60}$, then $D(n, pn) \leq 2^{(\log n)^{10^{60}}}$. Thus $p^* \leq \frac{1}{2} - 10^{-60}$.

Equivalently, among any $2^{(\log n)^{10^{60}}}$ strings in $\{0,1\}^n$, some two satisfy $\mathrm{LCS}(s,t) \geq (\frac{1}{2} + 10^{-60})n$.

Linear Number of Deletions

Theorem (Guruswami, H., Li '22)

If $p \geq \frac{1}{2} - 10^{-60}$, then $D(n, pn) \leq 2^{(\log n)^{10^{60}}}$. Thus $p^* \leq \frac{1}{2} - 10^{-60}$.

Equivalently, among any $2^{(\log n)^{10^{60}}}$ strings in $\{0,1\}^n$, some two satisfy $\mathrm{LCS}(s,t) \geq (\frac{1}{2} + 10^{-60})n$.

Proof Strategy

Classify strings according to how much they "look like" $1^\ell 0^\ell 1^\ell 0^\ell \cdots$ for each power of two ℓ. A crude analogy is assigning $\log n$ "Fourier coefficients" to s that measure its oscillation on each scale.

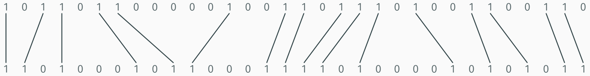
Pigeonhole to find s, t with the same oscillation statistics. This guarantees $\mathrm{LCS}(s,t)$ is large for three possible reasons:

(1) If s, t oscillate at a large scale $\ell = \Omega(n)$.

(2) If s, t share at least one "large Fourier coefficient" at the same scale.

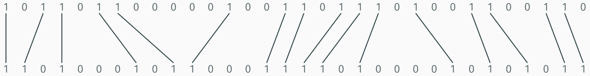
(3) If s, t share many "small Fourier coefficients" at different scales.

Case 1: Imbalanced Strings

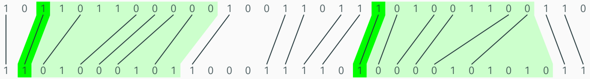


Matching Strategies

Case 1: Imbalanced Strings

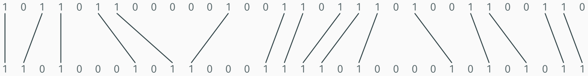


Case 2: Single-Frequency Strings

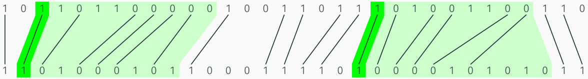


Flagged bits	
Blue	$d(I) \geq 0.99$
Green	$d(I) \geq 0.6$
Yellow	$d(I) \geq 0.49$

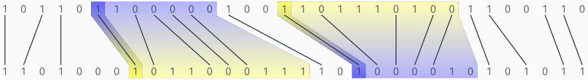
Case 1: Imbalanced Strings



Case 2: Single-Frequency Strings



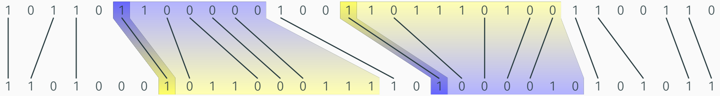
Case 3: Many-Frequency Strings



Flagged bits	
Blue	$d(I) \geq 0.99$
Green	$d(I) \geq 0.6$
Yellow	$d(I) \geq 0.49$

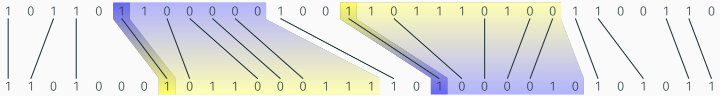
Technical Difficulty

Case 3: Many-Frequency Strings



Technical Difficulty

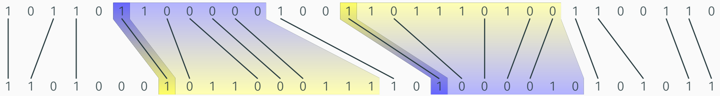
Case 3: Many-Frequency Strings



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!

Technical Difficulty

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

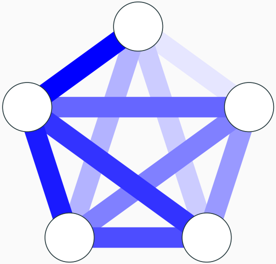
Theorem (Szemerédi '78)

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

The Regularity Method

Theorem (Szemerédi '78)

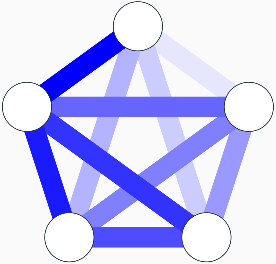
For every $\varepsilon > 0$, all sufficiently large graphs G can be partitioned into $O_\varepsilon(1)$ (nearly-)equal-sized vertex sets such that all but an ε-fraction of these pairs are ε-regular.



The Regularity Method

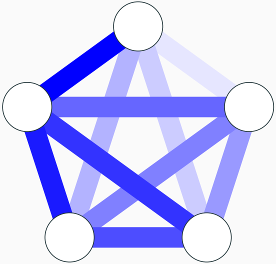
Theorem (Szemerédi '78)

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



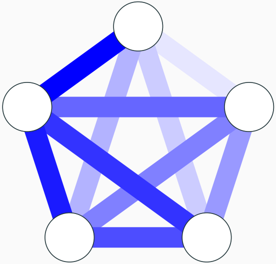
A pair of vertex sets X and Y are ε-regular if for all subsets $A \subseteq X$ and $B \subseteq Y$ satisfying $|A| \geq \varepsilon|X|$ and $|B| \geq \varepsilon|Y|$, we have $|d(A,B) - d(X,Y)| < \varepsilon$.

The Regularity Method



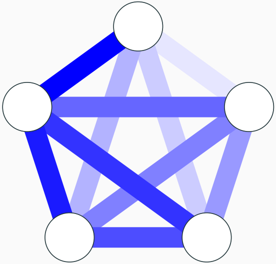
- Shows that for a **weak pseudorandomness property**, every graph can be nearly partitioned into pseudorandom parts.

The Regularity Method



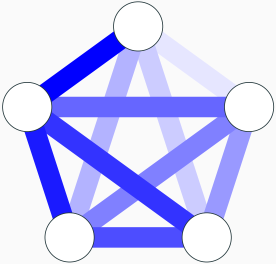
- Shows that for a **weak pseudorandomness property**, every graph can be nearly partitioned into pseudorandom parts.
- Useful for counting and embedding subgraphs.

The Regularity Method



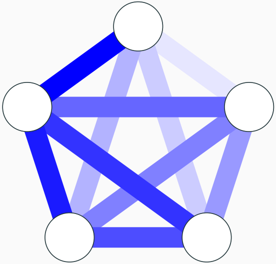
- Shows that for a **weak pseudorandomness property**, every graph can be nearly partitioned into pseudorandom parts.
- Useful for counting and embedding subgraphs.
- Numerous applications in extremal and additive combinatorics, for example Szemerédi's Theorem on arithmetic progressions.

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

A string s is ε-regular if for every subinterval I of length at least $\varepsilon|s|$, we have $|d(s_I) - d(s)| < \varepsilon$.

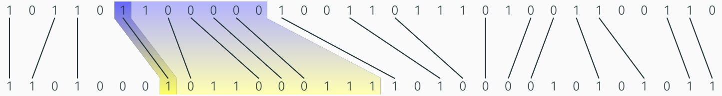
Regularity Lemmas for Binary Strings

Lemma (Axenovich, Person, Puzynina '12)

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

A string s is ε-regular if for every subinterval I of length at least $\varepsilon|s|$, we have $|d(s_I) - d(s)| < \varepsilon$.

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

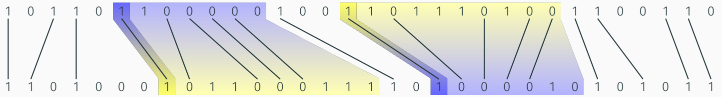


Lemma (Axenovich, Person, Puzynina '12)

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

A string s is ε-regular if for every subinterval I of length at least $\varepsilon|s|$, we have $|d(s_I) - d(s)| < \varepsilon$.

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



Bounds

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

The zero-rate threshold satisfies $\sqrt{2} - 1 \leq p^* \leq \frac{1}{2} - 10^{-60}$.

Bounds

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

The zero-rate threshold satisfies $\sqrt{2} - 1 \leq p^* \leq \frac{1}{2} - 10^{-60}$.

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

Bounds

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

The zero-rate threshold satisfies $\sqrt{2}-1 \leq p^* \leq \frac{1}{2} - 10^{-60}$.

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

$$\log n \ll_\varepsilon D(n, pn) \ll 2^{(\log n)^{10^{60}}}.$$

Bounds

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

The zero-rate threshold satisfies $\sqrt{2} - 1 \leq p^* \leq \frac{1}{2} - 10^{-60}$.

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

$$\log n \ll_\varepsilon D(n, pn) \ll 2^{(\log n)^{10^{60}}}.$$

- 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 *same* 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 *same* 1/2 barrier that appears in the deletion codes problem!

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

For all $\varepsilon > 0$, there exists $\delta > 0$ and a $O(n^{1+\varepsilon})$-time algorithm which gives a $(\frac{1}{2} + \delta)$-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 *same* 1/2 barrier that appears in the deletion codes problem!

Theorem (H., Li ’23, building on Rubinstein, Song ’20)

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

Uses the same “oscillation statistics” machinery.