Approximate Pattern Matching and the Query Complexity of Edit Distance

Krzysztof Onak
MIT

Joint work with:

Alexandr Andoni (CCI)

Robert Krauthgamer (Weizmann Institute)

Detecting an Internet Worm

- Typical task of antivirus software:

  Detecting incoming Internet worms and viruses

Detecting an Internet Worm

- Typical task of antivirus software:

  Detecting incoming Internet worms and viruses

- Must be very efficient if running on a user's computer:
  - Even a few MB/s to process.
  - Don't want to worsen user experience!

Detecting an Internet Worm

- Typical task of antivirus software:

  Detecting incoming Internet worms and viruses

- Must be very efficient if running on a user's computer:
  - Even a few MB/s to process.
  - Don't want to worsen user experience!
- Can detect harmful patterns by efficiently processing a fraction of a stream?

Subsampling Streams?

- Open Problems from IITK Workshop 2006

Question 13: Effects of Subsampling (Yossi Matias)

When processing very fast streams, it is not feasible to run a streaming algorithm on the entire stream, even one that can process each element in $O(1)$ time. Rather it is necessary to sample from the stream and to process the sub-stream using a streaming algorithm. For standard problems such as estimating F_0, how does the sub-sampling affect that the accuracy of the streaming algorithms? How should the sampling rate and the per-element time-complexity of a streaming algorithm be traded-off to achieve optimal results?

Another way to formalize this question, suggested by Muthukrishnan, is in terms of what part of the stream to skip and which to stream. A formal definition of the model and algorithms for estimating F_2 and others can be found in [BMMY07].

Subsampling Streams?

- Open Problems from IITK Workshop 2006

QUESTION 13: EFFECTS OF SUBSAMPLING (YOSSI MATIAS)

When processing very fast streams, it is not feasible to run a streaming algorithm on the entire stream, even one that can process each element in $O(1)$ time. Rather it is necessary to sample from the stream and to process the sub-stream using a streaming algorithm. For standard problems such as estimating F_0, how does the sub-sampling affect that the accuracy of the streaming algorithms? How should the sampling rate and the per-element time-complexity of a streaming algorithm be traded-off to achieve optimal results?

Another way to formalize this question, suggested by Muthukrishnan, is in terms of what part of the stream to skip and which to stream. A formal definition of the model and algorithms for estimating F_2 and others can be found in [BMMY07].

- ICDE 2007

How to scalably and accurately skip past streams

Supratik Bhattacharyya
Sprint ATL
supratik@gmail.com

André Madeira, S. Muthukrishnan
Rutgers University
[amadeira,muthu]@cs.rutgers.edu

Tao Ye
Sprint ATL
tao.ye@sprint.com

Abstract

Data stream methods look at each new item of the stream, perform a small number of operations while keeping a small amount of memory, and still perform much-needed analyses. However, in many situations, the update speed per item is extremely critical and not every item can be extensively examined. In practice, this has been addressed by only *examining every N^{th} item from the input; decreasing the input rate by a fraction $1/N$, but resulting in loss of guarantees on the accuracy of the post-hoc analyses.*

In this paper, we present a technique of skipping past streams and looking at only a fraction of the input. Unlike

amount of memory (aka *sketches* or *samples*), and still perform much-needed analyses on streams including data summarization, finding heavy hitters and quantiles, estimating self-join and statistical moments, etc. Operational DSMSs such as Gigascope [9] at AT&T and CMON [16] at Sprint are able to monitor hundreds of thousands of packet headers with these algorithms. This is essential for nearly every aspect of network management, including fault diagnosis, verifying service level agreements on network performance and most importantly, network security.

One of the most critical elements of a DSMS is the rate at which updates may be processed. In particular, in the IP network management application, there are three develop-

Streaming and Pattern Matching

Selected near-linear streaming algorithms:
($n =$ pattern size)

- Knuth, Morris, Pratt (1977)
  - deterministic
  - precomputes an array of proper prefixes in $O(n)$ time
  - amortized $O(1)$ time per each character
  - $O(n)$ space

Streaming and Pattern Matching

Selected near-linear streaming algorithms:
($n =$ pattern size)

- Knuth, Morris, Pratt (1977)
- Karp, Rabin (1987)
  - Exact algorithm
  - The idea of rolling hash
  - $O(1)$ time per character ($+$ perhaps check)
  - $O(n)$ space

Streaming and Pattern Matching

Selected near-linear streaming algorithms:
($n =$ pattern size)

- Knuth, Morris, Pratt (1977)
- Karp, Rabin (1987)
- Porat, Porat (tomorrow)
  - $O(\log n)$ space and update time
  - can also handle k mismatches in $O(k^2 \text{polylog}(n))$ time and $O(k^3 \text{polylog}(n))$ space

Approximate Pattern Matching

Data:

- Stream S of length m.
- Pattern P of length n

$S =$

1	1	1	0	0	1	1	0	1	0	1	0	0	1	1	1	0	1

$P =$

1	0	1	1	0

Approximate Pattern Matching

Data:

- Stream S of length m.
- Pattern P of length n

$S =$

1	1	1	0	0	1	1	0	1	0	1	0	0	1	1	1	0	1

$P =$

1	0	1	1	0

Goal:

- Report all length-n subwords x of S such that $\mathrm{dist}(x, P) \leq \alpha n$
- Don't report any x such that $\mathrm{dist}(x, P) \geq \beta n$

1	1	1	0	0	1	1	0	1	0	1	0	0	1	1	1	0	1

report **don't report**

Two Simple Algorithms: Hamming and Edit Distance

Hamming Distance

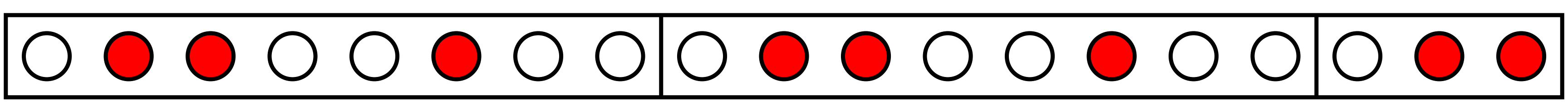
- Goal: want to report distance $\leq \alpha n$, but not $\geq \beta n$

Hamming Distance

- Goal: want to report distance $\leq \alpha n$, but not $\geq \beta n$
- Information theoretically:
  - Can use the Chernoff bound to estimate if a pattern approximately matches
  - Suffices to sample $O(\frac{m}{n} \cdot \frac{1}{(\beta-\alpha)^2} \cdot \log m)$ locations

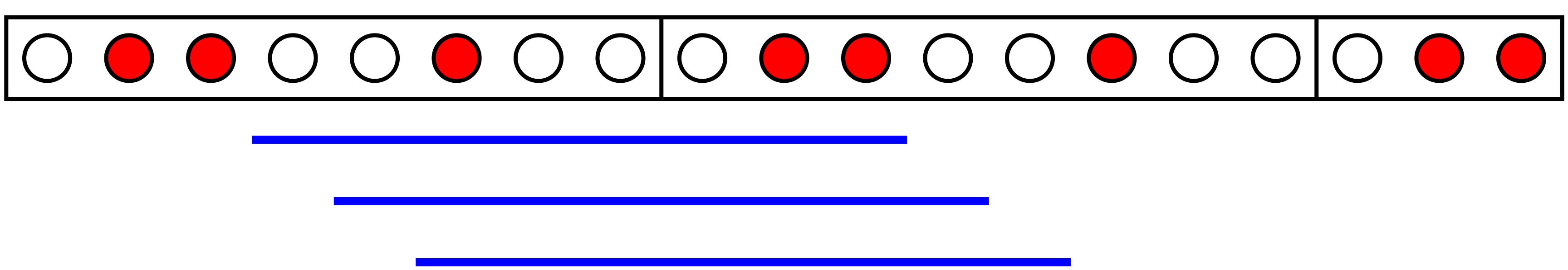
Hamming Distance

- Goal: want to report distance $\leq \alpha n$, but not $\geq \beta n$
- Efficient approach:
  - Sampling Pattern: random set of $q = O(\frac{1}{(\beta-\alpha)^2} \cdot \log m)$ indices in $\{1, 2, \ldots, n\}$, repeated modulo n



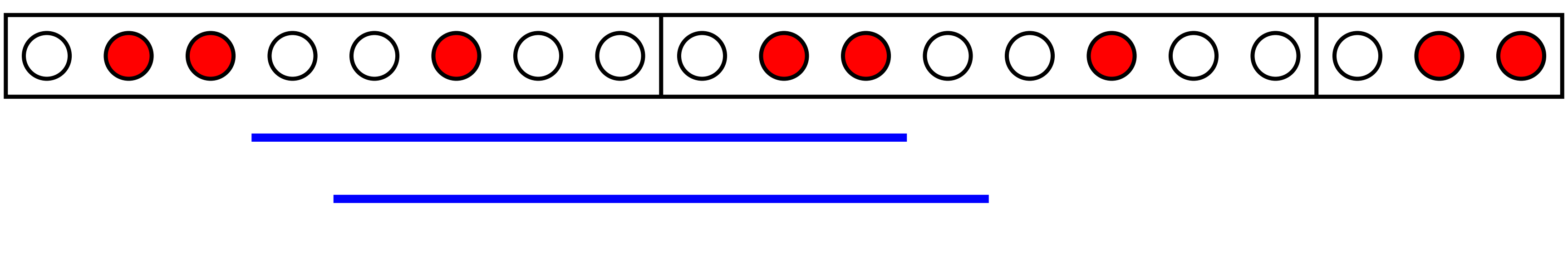
Hamming Distance

- Goal: want to report distance $\leq \alpha n$, but not $\geq \beta n$
- Efficient approach:
  - Sampling Pattern: random set of $q = O(\frac{1}{(\beta-\alpha)^2} \cdot \log m)$ indices in $\{1, 2, \ldots, n\}$, repeated modulo n
  - Use q approximate near neighbor data structures based on Locality Sensitive Hashing (Gionis, Indyk, Motwani 1999)



Hamming Distance

- Goal: want to report distance $\leq \alpha n$, but not $\geq \beta n$
- Efficient approach:
  - Sampling Pattern: random set of $q = O(\frac{1}{(\beta-\alpha)^2} \cdot \log m)$ indices in $\{1, 2, \ldots, n\}$, repeated modulo n
  - Use q approximate near neighbor data structures based on Locality Sensitive Hashing (Gionis, Indyk, Motwani 1999)
  - Approximate complexity ($\rho = \frac{\alpha}{\beta}$):
    - Time $\approx qn^{1+\rho} \cdot \log m + \frac{n}{m} \cdot q \cdot qn^{\rho} \cdot \log m +$ #matches $\cdot\, q$
    - Space $\approx qn^{1+\rho} \cdot \log m$



Edit Distance

- Batu, Ergün, Kilian, Magen, Raskhodnikova, Rubinfeld, Sami 2003:
  For a fixed constant $\alpha \in (0, 1)$, one can tell edit distance $O(n^\alpha)$ from $\Omega(n)$ in $\tilde{O}(n^{\max\{\alpha/2,\ 2\alpha-1\}})$ time

Edit Distance

- Batu, Ergün, Kilian, Magen, Raskhodnikova, Rubinfeld, Sami 2003:
  For a fixed constant $\alpha \in (0,1)$, one can tell edit distance $O(n^\alpha)$ from $\Omega(n)$ in $\tilde{O}(n^{\max\{\alpha/2,\ 2\alpha-1\}})$ time
- Reporting all subwords at distance $O(n^\alpha)$ and none at distance $\Omega(n)$:
  - It suffices to consider shifts by multiples of $\Theta(n^\alpha)$

Edit Distance

- Batu, Ergün, Kilian, Magen, Raskhodnikova, Rubinfeld, Sami 2003:
  For a fixed constant $\alpha \in (0,1)$, one can tell edit distance $O(n^\alpha)$ from $\Omega(n)$ in $\tilde{O}(n^{\max\{\alpha/2,\ 2\alpha-1\}})$ time
- Reporting all subwords at distance $O(n^\alpha)$ and none at distance $\Omega(n)$:
  - It suffices to consider shifts by multiples of $\Theta(n^\alpha)$
  - Run the BEKMRRS algorithm for each shift

Edit Distance

- Batu, Ergün, Kilian, Magen, Raskhodnikova, Rubinfeld, Sami 2003:

  For a fixed constant $\alpha \in (0,1)$, one can tell edit distance $O(n^\alpha)$ from $\Omega(n)$ in $\tilde{O}(n^{\max\{\alpha/2,\, 2\alpha-1\}})$ time

- Reporting all subwords at distance $O(n^\alpha)$ and none at distance $\Omega(n)$:
  - It suffices to consider shifts by multiples of $\Theta(n^\alpha)$
  - Run the BEKMRRS algorithm for each shift
  - Total time:

$$O(m/n^\alpha) \cdot \tilde{O}(n^{\max\{\alpha/2,\, 2\alpha-1\}}) \cdot O(\log m)$$

$$= O\left(\frac{m \cdot \log m \cdot \mathrm{polylog}(n)}{n^{\min\{\alpha/2,\, 1-\alpha\}}}\right)$$

Query Lower Bound for Edit Distance

How much do we have to see?

- Goal: Want to tell distance $.49n$ from $.51n$

How much do we have to see?

- Goal: Want to tell distance $.49n$ from $.51n$
- Hamming distance:
  - Upper bound: $O\left(\frac{m}{n}\log m\right)$
  - Trivial lower bound: $\Omega\left(\frac{m}{n}\right)$

How much do we have to see?

- Goal: Want to tell distance $.49n$ from $.51n$
- Hamming distance:
  - Upper bound: $O\left(\frac{m}{n}\log m\right)$
  - Trivial lower bound: $\Omega\left(\frac{m}{n}\right)$
- Main question:

Is edit distance harder?

How much do we have to see?

- Goal: Want to tell distance $.49n$ from $.51n$
- Hamming distance:
  - Upper bound: $O\left(\frac{m}{n}\log m\right)$
  - Trivial lower bound: $\Omega\left(\frac{m}{n}\right)$
- Main question:

  Is edit distance harder?

- We show higher dependence on n

The Model

- Input:
  - two strings x and y of length n
  - x is known to the algorithm
  - y is not known, the algorithm can query it

The Model

- Input:
  - two strings x and y of length n
  - x is known to the algorithm
  - y is not known, the algorithm can query it

- Question: How many queries are necessary to tell $\mathrm{ed}(x, y) \leq .49n$ from $\mathrm{ed}(x, y) \geq .51n$?

The Model

- Input:
  - two strings x and y of length n
  - x is known to the algorithm
  - y is not known, the algorithm can query it
- Question: How many queries are necessary to tell $\mathrm{ed}(x, y) \leq .49n$ from $\mathrm{ed}(x, y) \geq .51n$?
- From our point of view:
  - x is a pattern
  - y is any consecutive n characters of the stream

1st Attempt: Shifted Random Strings

- Pick two random strings z_0 and z_1 in $\{0,1\}^n$
- Very likely: $\mathrm{ed}(z_0, z_1) \geq .7n$

1st Attempt: Shifted Random Strings

- Pick two random strings z_0 and z_1 in $\{0,1\}^n$
- Very likely: $\mathrm{ed}(z_0, z_1) \geq .7n$
- Hard to tell apart:
  - Close:

$$x = z_0$$

$$y = (z_0 \text{ rotated by a random } s \text{ in [0,.01n]})$$

1st Attempt: Shifted Random Strings

- Pick two random strings z_0 and z_1 in $\{0,1\}^n$
- Very likely: $\mathrm{ed}(z_0, z_1) \geq .7n$
- Hard to tell apart:
  - Close:

$$x = z_0$$
$$y = (z_0 \text{ rotated by a random } s \text{ in [0,.01n]})$$

  - Far:

$$x = z_0$$
$$y = (z_1 \text{ rotated by a random } s \text{ in [0,.01n]})$$

1st Attempt: Shifted Random Strings

- Pick two random strings z_0 and z_1 in $\{0,1\}^n$
- Very likely: $\mathrm{ed}(z_0, z_1) \geq .7n$
- Hard to tell apart:
  - Close:

$$x = z_0$$
$$y = (z_0 \text{ rotated by a random } s \text{ in [0,.01n]})$$

  - Far:

$$x = z_0$$
$$y = (z_1 \text{ rotated by a random } s \text{ in [0,.01n]})$$

- Can show that $\Omega(\log n)$ queries necessary for constant success probability

1st Attempt: Shifted Random Strings

- Pick two random strings z_0 and z_1 in $\{0,1\}^n$
- Very likely: $\mathrm{ed}(z_0, z_1) \geq .7n$
- Hard to tell apart:
  - Close:

$$x = z_0$$
$$y = (z_0 \text{ rotated by a random } s \text{ in [0,.01n]})$$

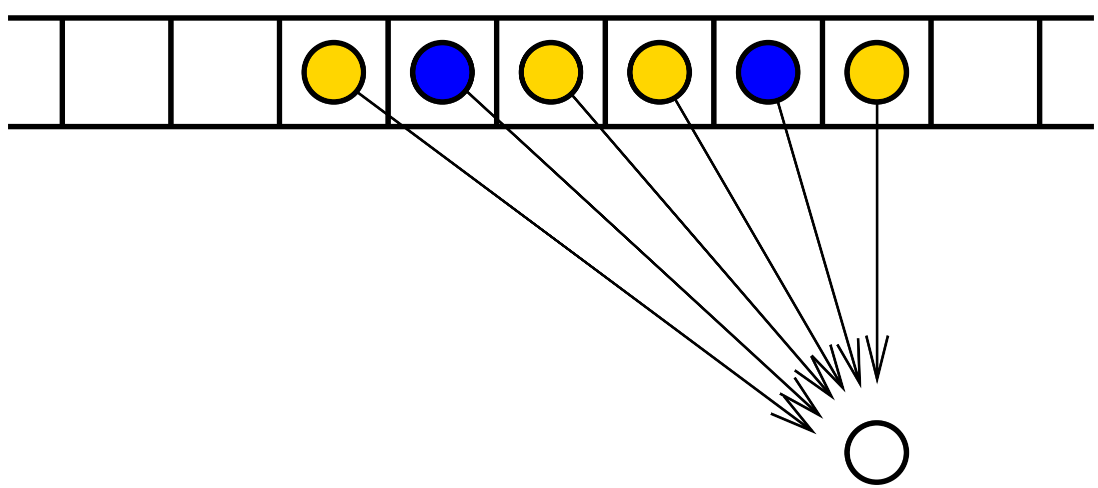
  - Far:

$$x = z_0$$
$$y = (z_1 \text{ rotated by a random } s \text{ in [0,.01n]})$$

- Can show that $\Omega(\log n)$ queries necessary for constant success probability
- Why? If $q =$ #queries small, the distribution on the view close to uniform

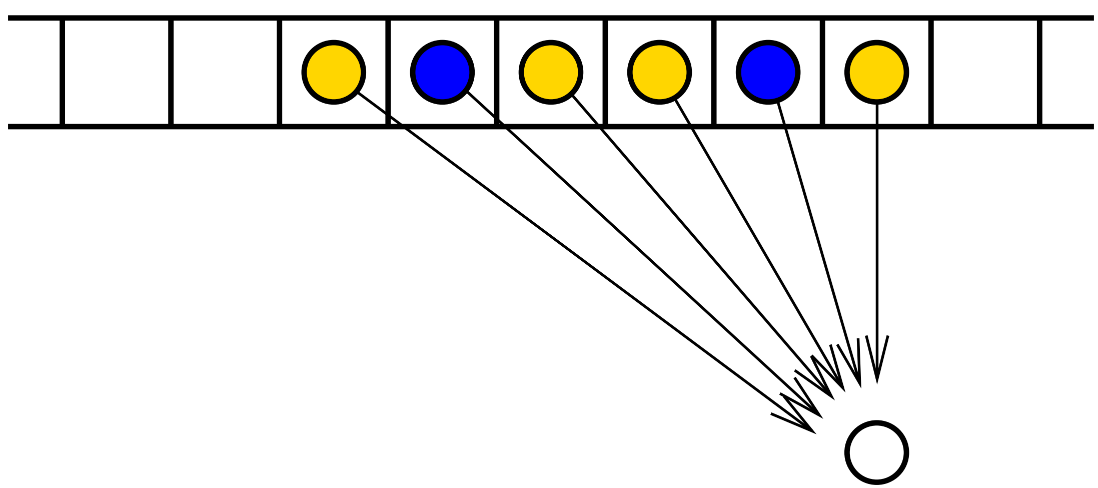
More Formally

- Let S = #shifts
- one query:
  - S random bits mapped to the query point



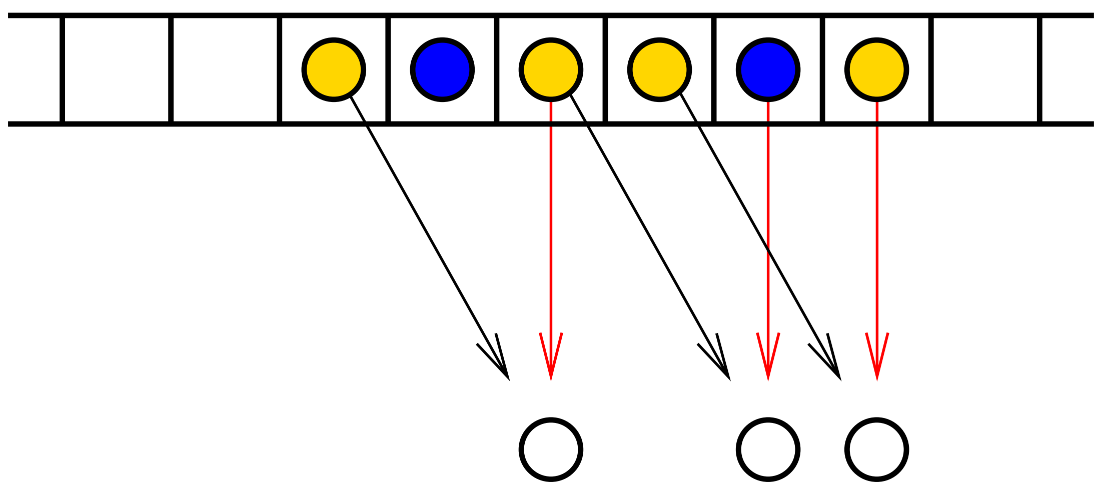
More Formally

- Let S = #shifts
- one query:
  - S random bits mapped to the query point
  - Chernoff $+$ union:
    probability any query point gives $\geq .01$ statistical difference bounded by $n \cdot 2^{-\Omega(S)}$ = negligible



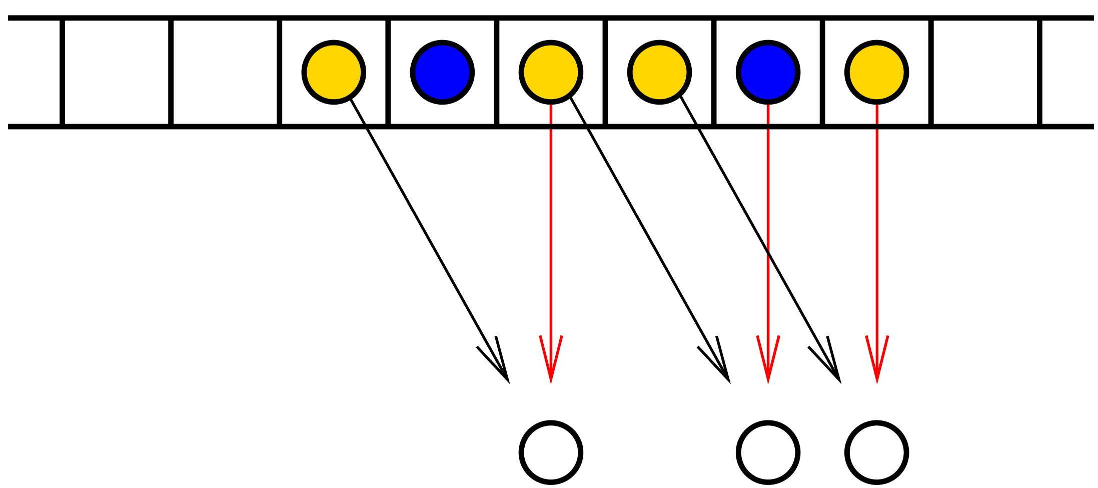
More Formally ($\#\text{queries} \geq 1$)

- Obstacle:
  - Can't use Chernoff directly
  - Subsets of random bits that map to the query subset can intersect



More Formally (#queries ≥ 1)

- Obstacle:
  - Can't use Chernoff directly
  - Subsets of random bits that map to the query subset can intersect
- Solution:
  - Every subset can intersect with less than q^2 other shifts
  - Balanced coloring of the shifts with q^2 colors
  - Apply Chernoff independently to each of them



2nd Attempt: Recursion

- The previous approach cannot give a lower bound better than logarithmic

2nd Attempt: Recursion

- The previous approach cannot give a lower bound better than logarithmic
- Substitution product $\circledast$:
  - $t \in \{0,1\}^k$ and $z_0, z_1 \in \{0,1\}^{k'}$:
$$t \circledast (z_0, z_1) = z_{t_1} z_{t_0} \ldots z_{t_{k-1}} z_{t_k}$$

$$z_0 = \mathbf{110}$$
$$z_1 = \mathbf{010}$$
$$t = \mathbf{10110} \longrightarrow t \circledast (z_0, z_1) = \mathbf{010\ 110\ 010\ 010\ 110}$$

2nd Attempt: Recursion

- The previous approach cannot give a lower bound better than logarithmic
- Substitution product $\circledast$:
  - $t \in \{0,1\}^k$ and $z_0, z_1 \in \{0,1\}^{k'}$:

    $$t \circledast (z_0, z_1) = z_{t_1} z_{t_0} \ldots z_{t_{k-1}} z_{t_k}$$

  - Distribution $\mathcal{T}$ on $\{0,1\}^k$, distributions $\mathcal{Z}_0, \mathcal{Z}_1$ on $\{0,1\}^{k'}$:

    Distribution $\mathcal{T} \circledast (\mathcal{Z}_0, Z_1)$:
    - take random t according to $\mathcal{T}$
    - replace each t_i with a random string independently chosen from $\mathcal{Z}_{t_i}$

$$\mathcal{Z}_0 \equiv \{\mathbf{110}, \mathbf{101}\}$$

$$\mathcal{Z}_1 \equiv \{\mathbf{001}, \mathbf{010}\}$$

$$\mathcal{T} \to t = \mathbf{1101} \longrightarrow t \circledast (z_0, z_1) = \mathbf{001\ 010\ 101\ 010}$$

2nd Attempt: Recursion

- Pick two random strings $z_0, z_1 \in \{0, 1\}^{\sqrt{n}}$

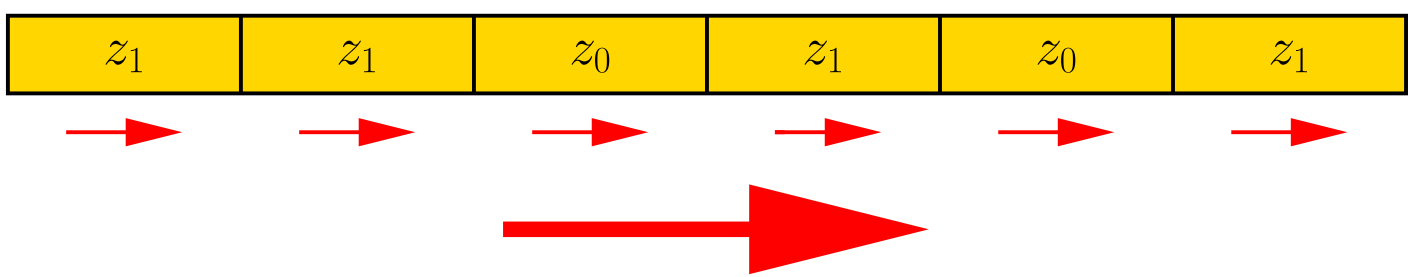
2nd Attempt: Recursion

- Pick two random strings $z_0, z_1 \in \{0,1\}^{\sqrt{n}}$
- Define two distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ on $\{0,1\}^{\sqrt{n}}$:
  - $\mathcal{D}_0 \sim$ random rotation of z_0 by s in $[0, .01\sqrt{n}]$
  - $\mathcal{D}_1 \sim$ random rotation of z_1 by s in $[0, .01\sqrt{n}]$

2nd Attempt: Recursion

- Pick two random strings $z_0, z_1 \in \{0,1\}^{\sqrt{n}}$
- Define two distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ on $\{0,1\}^{\sqrt{n}}$:
  - $\mathcal{D}_0 \sim$ random rotation of z_0 by s in $[0, .01\sqrt{n}]$
  - $\mathcal{D}_1 \sim$ random rotation of z_1 by s in $[0, .01\sqrt{n}]$
- Hard to tell apart:
  - Close:

$$x = z_0 \circledast (z_0, z_1)$$
$$y \leftarrow \mathcal{D}_0 \circledast (\mathcal{D}_0, \mathcal{D}_1)$$



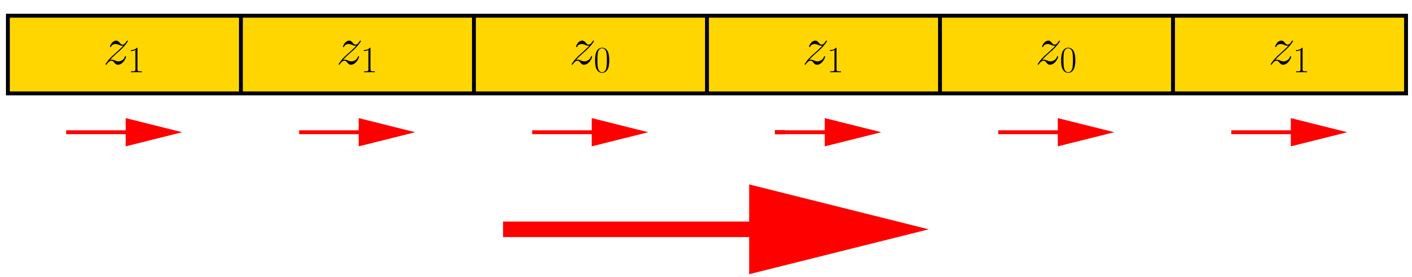
2nd Attempt: Recursion

- Pick two random strings $z_0, z_1 \in \{0,1\}^{\sqrt{n}}$
- Define two distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ on $\{0,1\}^{\sqrt{n}}$:
  - $\mathcal{D}_0 \sim$ random rotation of z_0 by s in $[0, .01\sqrt{n}]$
  - $\mathcal{D}_1 \sim$ random rotation of z_1 by s in $[0, .01\sqrt{n}]$
- Hard to tell apart:
  - Close:

$$x = z_0 \circledast (z_0, z_1)$$
$$y \leftarrow \mathcal{D}_0 \circledast (\mathcal{D}_0, \mathcal{D}_1)$$

  - Far:

$$x = z_0 \circledast (z_0, z_1)$$
$$y \leftarrow \mathcal{D}_1 \circledast (\mathcal{D}_0, \mathcal{D}_1)$$



Analysis

1. Is $y \leftarrow \mathcal{D}_1 \circledast (\mathcal{D}_0, \mathcal{D}_1)$ far from $z_0 \circledast (z_0, z_1)$?
   - Suffices to show for $z_1 \circledast (z_0, z_1)$
   - If they were too close, then z_0 would be close to z_1, which is unlikely

Analysis

1. Is $y \leftarrow \mathcal{D}_1 \circledast (\mathcal{D}_0, \mathcal{D}_1)$ far from $z_0 \circledast (z_0, z_1)$?
   - Suffices to show for $z_1 \circledast (z_0, z_1)$
   - If they were too close, then z_0 would be close to z_1, which is unlikely
2. Is it hard to tell $\mathcal{D}_0 \circledast (\mathcal{D}_0, \mathcal{D}_1)$ from $\mathcal{D}_1 \circledast (\mathcal{D}_0, \mathcal{D}_1)$?

Analysis

1. Is $y \leftarrow \mathcal{D}_1 \circledast (\mathcal{D}_0, \mathcal{D}_1)$ far from $z_0 \circledast (z_0, z_1)$?
   - Suffices to show for $z_1 \circledast (z_0, z_1)$
   - If they were too close, then z_0 would be close to z_1, which is unlikely
2. Is it hard to tell $\mathcal{D}_0 \circledast (\mathcal{D}_0, \mathcal{D}_1)$ from $\mathcal{D}_1 \circledast (\mathcal{D}_0, \mathcal{D}_1)$?
   - q-Query Advantage:
     - $Q \subseteq \{1, \ldots, k\}$: $\mathcal{D}|_Q = \mathcal{D}$ projected on coordinates in Q
     - distributions $\mathcal{A}_0, \mathcal{A}_1$ on $\{0,1\}^k$
     - $\mathrm{Adv}_q(A_0, A_i) = \max_{Q \subseteq \{1,\ldots,k\}, |Q|=q} \Delta(A_0|_Q, A_1|_Q)$

Analysis

1. Is $y \leftarrow \mathcal{D}_1 \circledast (\mathcal{D}_0, \mathcal{D}_1)$ far from $z_0 \circledast (z_0, z_1)$?
   - Suffices to show for $z_1 \circledast (z_0, z_1)$
   - If they were too close, then z_0 would be close to z_1, which is unlikely
2. Is it hard to tell $\mathcal{D}_0 \circledast (\mathcal{D}_0, \mathcal{D}_1)$ from $\mathcal{D}_1 \circledast (\mathcal{D}_0, \mathcal{D}_1)$?
   - q-Query Advantage:
     - $Q \subseteq \{1, \ldots, k\}$: $\mathcal{D}|_Q = \mathcal{D}$ projected on coordinates in Q
     - distributions $\mathcal{A}_0, \mathcal{A}_1$ on $\{0,1\}^k$
     - $\mathrm{Adv}_q(A_0, A_i) = \max_{Q \subseteq \{1,\ldots,k\}, |Q|=q} \Delta(A_0|_Q, A_1|_Q)$
   - Composition Lemma:
     - distributions $\mathcal{D}_0$, $\mathcal{D}_1$, $\mathcal{E}_0$, $\mathcal{E}_1$
       s.t. $\mathrm{Adv}_q(\mathcal{D}_0, \mathcal{D}_1) \leq \frac{q}{A}$ and $\mathrm{Adv}_q(\mathcal{E}_0, \mathcal{E}_1) \leq \frac{q}{B}$
     - $\mathrm{Adv}_q(\mathcal{D}_0 \circledast (\mathcal{E}_0, \mathcal{E}_1), \mathcal{D}_1 \circledast (\mathcal{E}_0, \mathcal{E}_1)) = \frac{q}{AB}$

Sketch of Proof

- Consider any set Q of q queries

Sketch of Proof

- Consider any set Q of q queries
- $q_i =$ number of queries to block i

Sketch of Proof

- Consider any set Q of q queries
- $q_i =$ number of queries to block i
- The view of i-th block hits the difference between $\mathcal{E}_0$ and $\mathcal{E}_1$ with probability $\leq q_i/B$ (the block is hit)

Sketch of Proof

- Consider any set Q of q queries
- $q_i =$ number of queries to block i
- The view of i-th block hits the difference between $\mathcal{E}_0$ and $\mathcal{E}_1$ with probability $\leq q_i/B$ (the block is hit)
- For every subset H of hit blocks, cannot tell $\mathcal{D}_0 \circledast (\mathcal{E}_0, \mathcal{E}_1)$ from $\mathcal{D}_1 \circledast (\mathcal{E}_0, \mathcal{E}_1)$ with probability greater than $|H|/A$

Sketch of Proof

- Consider any set Q of q queries
- $q_i =$ number of queries to block i
- The view of i-th block hits the difference between $\mathcal{E}_0$ and $\mathcal{E}_1$ with probability $\le q_i/B$ (the block is hit)
- For every subset H of hit blocks, cannot tell $\mathcal{D}_0 \circledast (\mathcal{E}_0, \mathcal{E}_1)$ from $\mathcal{D}_1 \circledast (\mathcal{E}_0, \mathcal{E}_1)$ with probability greater than $|H|/A$
- Finally:

$$\Delta(\mathcal{D}_0 \circledast (\mathcal{E}_0, \mathcal{E}_1)|_Q, \mathcal{D}_1 \circledast (\mathcal{E}_0, \mathcal{E}_1)|_Q) \le \sum_{\text{blocks } H} \Pr[H \text{ are hit}] \cdot \frac{|H|}{A}$$

$$= \frac{E[\text{\#hit blocks}]}{A} \le \frac{1}{A} \sum_i \frac{q_i}{B} = \frac{q}{AB}$$

How Far Can This Take Us?

- For every constant k, can repeat k times to get $\Omega(\log^k n)$

How Far Can This Take Us?

- For every constant k, can repeat k times to get $\Omega(\log^k n)$
- Distances shrink with k:
  - Must switch to a larger alphabet
  - Can map at random to $\{0, 1\}$ at the end

How Far Can This Take Us?

- For every constant k, can repeat k times to get $\Omega(\log^k n)$
- Distances shrink with k:
  - Must switch to a larger alphabet
  - Can map at random to $\{0, 1\}$ at the end
- Final bound: $2^{\Omega\left(\frac{\log n}{\log \log n}\right)}$

How Far Can This Take Us?

- For every constant k, can repeat k times to get $\Omega(\log^k n)$
- Distances shrink with k:
  - Must switch to a larger alphabet
  - Can map at random to $\{0, 1\}$ at the end
- Final bound: $2^{\Omega\left(\frac{\log n}{\log \log n}\right)}$
- Main open question:

Is there a polynomial lower bound?

Thank you!