Trace reconstruction for deletion channels

Yuval Peres
Microsoft Research

Based on joint work with

Fedor Nazarov
Kent State University

Alex Zhai
Stanford University

June 18, 2018
Problem statement
Suppose Alice wants to send to Bob an $n$-bit string $\mathbf{x} = (x_0, \ldots, x_{n-1}) \in \{0, 1\}^n$. Each bit has some probability $q$ of being deleted.
Suppose Alice wants to send to Bob an $n$-bit string
\[ \mathbf{x} = (x_0, \ldots, x_{n-1}) \in \{0, 1\}^n. \]

Alice transmits the bits one by one, but each bit has some probability $q$ of being deleted.
Suppose Alice wants to send to Bob an $n$-bit string 
\[ \mathbf{x} = (x_0, \ldots, x_{n-1}) \in \{0, 1\}^n. \]

Alice transmits the bits one by one, but each bit has some probability $q$ of being deleted.

Bob doesn’t know which positions were deleted; all he sees is a shortened string $(y_0, y_1, \ldots, y_{\ell-1})$. 

Suppose Alice wants to send to Bob an $n$-bit string
\[ x = (x_0, \ldots, x_{n-1}) \in \{0, 1\}^n. \]

Alice transmits the bits one by one, but each bit has some probability $q$ of being deleted.

Bob doesn’t know which positions were deleted; all he sees is a shortened string $(y_0, y_1, \ldots, y_{\ell-1})$
Notation: $\mathcal{D}_q(x)$ denotes the distribution over strings that Bob receives after passing through deletion channel.
Questions

- Notation: $\mathcal{D}_q(x)$ denotes the distribution over strings that Bob receives after passing through deletion channel.
- Given $T$ i.i.d. samples ("traces") $y^1, y^2, \ldots, y^T$ with each $y^t \sim \mathcal{D}_q(x)$, can Bob reconstruct $x$ (with probability $3/4$, say)?
Questions

- Notation: \( D_q(x) \) denotes the distribution over strings that Bob receives after passing through deletion channel.
- Given \( T \) i.i.d. samples ("traces") \( y_1, y_2, \ldots, y_T \) with each \( y^t \sim D_q(x) \), can Bob reconstruct \( x \) (with probability 3/4, say)?
- Closely related hypothesis testing problem: Given two strings \( x \) and \( x' \), determine if samples came from \( D_q(x) \) or \( D_q(x') \).
  
  If \( T \) traces suffice for this (with probability 3/4, say), then \( O(nT) \) traces suffice for reconstruction.
Notation: $\mathcal{D}_q(x)$ denotes the distribution over strings that Bob receives after passing through deletion channel.

Given $T$ i.i.d. samples ("traces") $y^1, y^2, \ldots, y^T$ with each $y^t \sim \mathcal{D}_q(x)$, can Bob reconstruct $x$ (with probability $3/4$, say)?

Closely related hypothesis testing problem: Given two strings $x$ and $x'$, determine if samples came from $\mathcal{D}_q(x)$ or $\mathcal{D}_q(x')$.

If $T$ traces suffice for this (with probability $3/4$, say), then $O(nT)$ traces suffice for reconstruction.

Can ask for worst case $x$ or for “average case” $x$ (where $x$ is chosen uniformly at random).
Questions

- Notation: $\mathcal{D}_q(x)$ denotes the distribution over strings that Bob receives after passing through deletion channel.
- Given $T$ i.i.d. samples ("traces") $y^1, y^2, \ldots, y^T$ with each $y^t \sim \mathcal{D}_q(x)$, can Bob reconstruct $x$ (with probability $3/4$, say)?
- Closely related hypothesis testing problem: Given two strings $x$ and $x'$, determine if samples came from $\mathcal{D}_q(x)$ or $\mathcal{D}_q(x')$.
  If $T$ traces suffice for this (with probability $3/4$, say), then $O(nT)$ traces suffice for reconstruction.
- Can ask for worst case $x$ or for "average case" $x$ (where $x$ is chosen uniformly at random).
- Arises naturally in various contexts: sensor networks, DNA sequencing.
- Problem raised in this form by Batu, Kannan, Khanna and McGregor (2004), who proved a lower bound: For all $n > 1$ there exist strings $x, x'$ of $n$ bits such that $\Omega(n)$ traces are needed to distinguish whether the input was $x$ or $x'$. 
Observation: $x$ and $x'$ are any two $n$-bit strings with different Hamming weights, then $T = O(n)$ traces suffice to distinguish them, using Hamming weight of the output as test statistic.

Previous upper bounds: $e^{O(\sqrt{n})}$ in worst case and $n^{O(1)}$ in random case for $q < 1/100$ (Holenstein-Mitzenmacher-Panigrahy-Wieder '08).

For worst case $x$, we can reconstruct using $e^{O(n^{1/3})}$ traces. Moreover, this is optimal for linear (mean-based) tests.

Same result obtained simultaneously and independently by De, O'Donnell and Servedio (STOC 2017).

New result (P.-Zhai, FOCS 2017): For $q < 1/2$, we can reconstruct a uniform random input $x$ with probability $1 - o(1)$ using $T = e^{C\sqrt{\log n}} = n^{o(1)}$ traces.
Observation: $x$ and $x'$ are any two $n$-bit strings with different Hamming weights, then $T = O(n)$ traces suffice to distinguish them, using Hamming weight of the output as test statistic.

Previous upper bounds: $e^{O(\sqrt{n})}$ in worst case and $n^{O(1)}$ in random case for $q < 1/100$ (Holenstein-Mitzenmacher-Panigrahy-Wieder ’08).

(Nazarov-P., STOC 2017): For worst case $x$, we can reconstruct using $e^{O(n^{1/3})}$ traces. Moreover, this is optimal for linear (mean-based) tests.

Same result obtained simultaneously and independently by De, O’Donnell and Servedio (STOC 2017).
Observation: $x$ and $x'$ are any two $n$-bit strings with different Hamming weights, then $T = O(n)$ traces suffice to distinguish them, using Hamming weight of the output as test statistic.

Previous upper bounds: $e^{O(\sqrt{n})}$ in worst case and $n^{O(1)}$ in random case for $q < 1/100$ (Holenstein-Mitzenmacher-Panigrahy-Wieder ’08).

(Nazarov-P., STOC 2017): For worst case $x$, we can reconstruct using $e^{O(n^{1/3})}$ traces. Moreover, this is optimal for linear (mean-based) tests.

Same result obtained simultaneously and independently by De, O’Donnell and Servedio (STOC 2017).

New result (P.-Zhai, FOCS 2017): For $q < 1/2$, we can reconstruct a uniform random input $x$ with probability $1 - o(1)$ using

$$T = e^{C\sqrt{\log n}} = n^{o(1)}$$ traces.
Observation: $x$ and $x'$ are any two $n$-bit strings with different Hamming weights, then $T = O(n)$ traces suffice to distinguish them, using Hamming weight of the output as test statistic.

Previous upper bounds: $e^{O(\sqrt{n})}$ in worst case and $n^{O(1)}$ in random case for $q < 1/100$ (Holenstein-Mitzenmacher-Panigrahy-Wieder '08).

(Nazarov-P., STOC 2017): For worst case $x$, we can reconstruct using $e^{O(n^{1/3})}$ traces. Moreover, this is optimal for linear (mean-based) tests.

Same result obtained simultaneously and independently by De, O’Donnell and Servedio (STOC 2017).

New result (P.-Zhai, FOCS 2017): For $q < 1/2$, we can reconstruct a uniform random input $x$ with probability $1 - o(1)$ using

$$T = e^{C\sqrt{\log n}} = n^{o(1)}$$ traces.
Lower bounds

For worst case, consider $x = 0000 \ldots 000$ (n/2 zeroes) $111 \ldots 111$ (n/2 ones).

$x' = 0000 \ldots 0000$ (n/2 + 1 zeroes) $11 \ldots 111$ (n/2 − 1 ones).

Trace reconstruction is basically equivalent to distinguishing $\text{Binom}(n/2, p)$ and $\text{Binom}(n/2 + 1, p) \Rightarrow$ need $\Omega(n)$ traces.

For random case, need at least $\Omega(\log_2 n)$ traces (McGregor-Price-Vorotnikova '14).
For worst case, consider

\[ x = 0000 \cdots 000111 \cdots 1111 \]
\[ \text{\( \frac{n}{2} \) zeroes} \quad \text{\( \frac{n}{2} \) ones} \]

\[ x' = 0000 \cdots 00011 \cdots 1111 \]
\[ \text{\( \frac{n}{2} + 1 \) zeroes} \quad \text{\( \frac{n}{2} - 1 \) ones} \]
For worst case, consider

\[
\mathbf{x} = \underbrace{0000 \cdots 000}_{n/2 \text{ zeroes}}  \underbrace{111 \cdots 111}_{n/2 \text{ ones}}
\]

\[
\mathbf{x}' = \underbrace{0000 \cdots 000}_{n/2 + 1 \text{ zeroes}}  \underbrace{11 \cdots 111}_{n/2 - 1 \text{ ones}}.
\]

Trace reconstruction is basically equivalent to distinguishing \( \text{Binom}(n/2, p) \) and \( \text{Binom}(n/2 + 1, p) \) \( \implies \) need \( \Omega(n) \) traces.
For worst case, consider

\[ x = 0000 \cdots 000111 \cdots 1111 \]

\[ \text{n/2 zeroes} \quad \text{n/2 ones} \]

\[ x' = 0000 \cdots 000011 \cdots 1111. \]

\[ \text{n/2 + 1 zeroes} \quad \text{n/2 - 1 ones} \]

Trace reconstruction is basically equivalent to distinguishing \( \text{Binom}(n/2, p) \) and \( \text{Binom}(n/2 + 1, p) \) \( \implies \) need \( \Omega(n) \) traces.

For random case, need at least \( \Omega(\log^2 n) \) traces (McGregor-Price-Vorotnikova '14).
Reconstruction with bit statistics
For simplicity, take $q = 1/2$ (general case is similar).
For simplicity, take $q = 1/2$ (general case is similar).

Natural first attempt: suppose $y \sim D_q(x)$ and $y' \sim D_q(x')$. Does first bit of $y$ look different from first bit of $y'$?
For simplicity, take $q = 1/2$ (general case is similar).

Natural first attempt: suppose $y \sim D_q(x)$ and $y' \sim D_q(x')$. Does first bit of $y$ look different from first bit of $y'$?

$$E_{y_0} = \frac{1}{2}x_0 + \frac{1}{4}x_1 + \frac{1}{8}x_2 + \cdots$$

$$E_{y'_0} = \frac{1}{2}x'_0 + \frac{1}{4}x'_1 + \frac{1}{8}x'_2 + \cdots$$
For simplicity, take $q = 1/2$ (general case is similar).

Natural first attempt: suppose $y \sim \mathcal{D}_q(x)$ and $y' \sim \mathcal{D}_q(x')$. Does first bit of $y$ look different from first bit of $y'$?

$$
\mathbb{E}y_0 = \frac{1}{2}x_0 + \frac{1}{4}x_1 + \frac{1}{8}x_2 + \cdots 
$$
$$
\mathbb{E}y'_0 = \frac{1}{2}x'_0 + \frac{1}{4}x'_1 + \frac{1}{8}x'_2 + \cdots 
$$

If $x$ and $x'$ agree in first $k$ digits, then $|\mathbb{E}y_0 - \mathbb{E}y'_0|$ is only $\approx 2^{-k}$. 

Exponentially many samples needed: Requires at least $2^k$ traces to distinguish.
Bit statistics: the first bit

- For simplicity, take $q = 1/2$ (general case is similar).
- Natural first attempt: suppose $y \sim D_q(x)$ and $y' \sim D_q(x')$. Does first bit of $y$ look different from first bit of $y'$?

\[
E_{y_0} = \frac{1}{2}x_0 + \frac{1}{4}x_1 + \frac{1}{8}x_2 + \cdots
\]
\[
E_{y'_0} = \frac{1}{2}x'_0 + \frac{1}{4}x'_1 + \frac{1}{8}x'_2 + \cdots
\]

- If $x$ and $x'$ agree in first $k$ digits, then $|E_{y_0} - E_{y'_0}|$ is only $\approx 2^{-k}$.
- Exponentially many samples needed: Requires at least $2^k$ traces to distinguish.
We can try other output bits $y_j$ besides $y_0$. For $y_j$ to come from $x_k$, this bit and exactly $j$ bits among $x_0, \ldots, x_{k-1}$ should be retained, so
We can try other output bits $y_j$ besides $y_0$. For $y_j$ to come from $x_k$, this bit and exactly $j$ bits among $x_0, \ldots, x_{k-1}$ should be retained, so

$$
E y_j = \frac{1}{2} \sum_{k \geq j} \frac{1}{2^k} \binom{k}{j} x_k.
$$
We can try other output bits $y_j$ besides $y_0$. For $y_j$ to come from $x_k$, this bit and exactly $j$ bits among $x_0, \ldots, x_{k-1}$ should be retained, so

$$
E y_j = \frac{1}{2} \sum_{k \geq j} \frac{1}{2^k} \binom{k}{j} x_k.
$$

Formula for $E y_j$ is best summarized by a generating function identity:

$$
E \left[ \sum_{j=0}^{n-1} y_j w^j \right] = \frac{1}{2} \sum_{k=0}^{n-1} x_k \left( \frac{w + 1}{2} \right)^k.
$$
Using the key identity

\[
\Psi_y(w) := \mathbb{E} \left[ \sum_{j=0}^{n-1} y_j w^j \right] = \frac{1}{2} \sum_{k=0}^{n-1} x_k \left( \frac{w + 1}{2} \right)^k.
\]

- Goal: find small \( w \) so that \( \Psi_y(w) \) and \( \Psi_y'(w) \) differ substantially.
Using the key identity

\[
\psi_y(w) := E \left[ \sum_{j=0}^{n-1} y_j w^j \right] = \frac{1}{2} \sum_{k=0}^{n-1} x_k \left( \frac{w + 1}{2} \right)^k .
\]

- Goal: find small \( w \) so that \( \psi_y(w) \) and \( \psi_y'(w) \) differ substantially.
- Letting \( z = \frac{w + 1}{2} \), we have

\[
\psi_y(w) - \psi_y'(w) = \frac{1}{2} \sum_{k=0}^{n-1} (x_k - x'_k) z^k .
\]
Using the key identity

\[
\Psi_y(w) := E \left[ \sum_{j=0}^{n-1} y_j w^j \right] = \frac{1}{2} \sum_{k=0}^{n-1} x_k \left( \frac{w + 1}{2} \right)^k.
\]

- Goal: find small \( w \) so that \( \Psi_y(w) \) and \( \Psi_y'(w) \) differ substantially.
- Letting \( z = \frac{w + 1}{2} \), we have

\[
\Psi_y(w) - \Psi_y'(w) = \frac{1}{2} \sum_{k=0}^{n-1} (x_k - x_k') z^k.
\]

- Suffices to find small \( z \) so that RHS of above expression is large.
Theorem (Borwein-Erdélyi)

Let

\[ f(z) = \sum_{k=0}^{n-1} a_k z^k \]

be a polynomial with coefficients \( a_0 = 1 \) and \( |a_k| \leq 1 \).

For any arc of length \( 1/L \) on the unit circle, there is a point \( z \) on the arc such that \( |f(z)| \geq e^{-cL} \), where \( c \) is a universal constant.
The maximum of a polynomial on a small arc

Theorem (Borwein-Erdélyi)

Let

\[ f(z) = \sum_{k=0}^{n-1} a_k z^k \]

be a polynomial with coefficients \( a_0 = 1 \) and \( |a_k| \leq 1 \).

For any arc of length \( 1/L \) on the unit circle, there is a point \( z \) on the arc such that \( |f(z)| \geq e^{-cL} \), where \( c \) is a universal constant.

Apply to

\[ f(z) = \sum_{j=0}^{n-1} (x_j - x'_j) z^j, \]

dividing out by a power of \( z \) if needed.
The maximum of a polynomial on a small arc

Theorem (Borwein-Erdélyi)

Let

$$f(z) = \sum_{k=0}^{n-1} a_k z^k$$

be a polynomial with coefficients $a_0 = 1$ and $|a_k| \leq 1$.

For any arc of length $1/L$ on the unit circle, there is a point $z$ on the arc such that $|f(z)| \geq e^{-cL}$, where $c$ is a universal constant.

- Apply to

$$f(z) = \sum_{j=0}^{n-1} (x_j - x'_j) z^j,$$

dividing out by a power of $z$ if needed.

- Can find $z$ in given arc so that $|\Psi_y(w) - \Psi'_y(w)| \geq e^{-cL}$. 

Y. Peres (MSR)
How to make $w$ small?
Choose $z$ near 1.

If $z = e^{i \theta}$, then $|w| = 1 + O(\theta^2)$.

With $\theta = O(1/L)$, we obtain $|w| = 1 + O(1/L^2)$. 
How to make $w$ small?
Choose $z$ near 1.

If $z = e^{i\theta}$, then

$$|w| = 1 + O(\theta^2).$$
How to make $w$ small?
Choose $z$ near 1.

If $z = e^{i\theta}$, then

$$|w| = 1 + O(\theta^2).$$

With $\theta = O(1/L)$, we obtain

$$|w| = 1 + O(1/L^2).$$
Conclusion:

\[
\left| \sum_{j=0}^{n-1} (E_{yj} - E_{y'_j}) w^j \right| \geq e^{-cL}
\]

where \( |w| = 1 + O(1/L^2) \implies |w^j| \leq e^{Cn/L^2} \). We may assume \( C > c \).
Conclusion:

\[
\left| \sum_{j=0}^{n-1} (E_{y_j} - E_{y'_j}) w^j \right| \geq e^{-cL}
\]

where \( |w| = 1 + O(1/L^2) \), \( w^j \leq e^{Cn/L^2} \). We may assume \( C > c \).

Thus there is some \( j \) such that

\[
|E_{y_j} - E_{y'_j}| \geq \frac{1}{n} e^{-CL - Cn/L^2} \geq e^{-3Cn^{1/3}} =: \epsilon.
\]

(taking \( L = n^{1/3} \) to minimize \( L + n/L^2 \) and absorbing \( 1/n \) term)
Conclusion:

\[
\left| \sum_{j=0}^{n-1} (E y_j - E y'_j) w^j \right| \geq e^{-cL}
\]

where \( |w| = 1 + O(1/L^2) \) \( \implies \) \( |w|^j \leq e^{Cn/L^2} \). We may assume \( C > c \).

Thus there is some \( j \) such that

\[
|E y_j - E y'_j| \geq \frac{1}{n} e^{-CL-Cn/L^2} \geq e^{-3Cn^{1/3}} =: \epsilon.
\]

(taking \( L = n^{1/3} \) to minimize \( L + n/L^2 \) and absorbing \( 1/n \) term)

\( \implies \ T = e^{7cn^{1/3}} \) samples suffice to detect the difference in means:

Probability of choosing wrongly between \( x \) and \( x' \) is \( e^{-\Omega(T\epsilon^2)} \) which is much smaller than \( 2^{-n} \).
To avoid enumerating over all $2^n$ possible input strings, one can use linear programming, following Holenstein et al (2008). Suppose that $x_0, \ldots, x_{m-1}$ have been reconstructed and we wish to determine $x_m$. Write $\bar{y}_j$ for the empirical average of the output bits $\frac{1}{T} \sum_{t=1}^{T} y_j^t$. Let $L := n^{1/3}$ and consider two linear programs (one where $x_m = 0$ and one where $x_m = 1$) in the relaxed variables $x_{m+1}, \ldots, x_n$ in $[0, 1]$:

$$|\mathbf{E}(y_j) - \bar{y}_j| < e^{-cL} \text{ where } \mathbf{E}(y_j) = \frac{1}{2} \sum_{k \geq j} \frac{1}{2^k} \binom{k}{j} x_k.$$  

Only one of these programs (either the LP determined by $x_m = 0$ or by $x_m = 1$) will be feasible if $C$ is large enough and $T = e^{7cn^{1/3}}$. 

Reducing complexity
Take $\Gamma$ to be a curve overlapping with the unit circle in an arc of length $1/L$, as shown.
Take $\Gamma$ to be a curve overlapping with the unit circle in an arc of length $1/L$, as shown.

Since $f$ is analytic, $\log |f(x)|$ is subharmonic. Thus,

\[
0 = \log |f(0)| \leq \int_{z \in \Gamma} \log |f(z)| \, d\omega(z).
\]
Rearranging yields

\[ \int_{z \text{ blue}} \log |f(z)| \, d\omega(z) \geq - \int_{z \text{ green}} \log |f(z)| \, d\omega(z). \]
Rearranging yields

\[ \int_{z \in \text{blue}} \log |f(z)| \, d\omega(z) \geq - \int_{z \in \text{green}} \log |f(z)| \, d\omega(z). \]

For $|z| < 1$, we have

\[ |f(z)| \leq \sum_{j=0}^{\infty} |z|^j = \frac{1}{1 - |z|}. \]
Rearranging yields

$$\int_{z \text{ blue}} \log |f(z)| \, d\omega(z) \geq - \int_{z \text{ green}} \log |f(z)| \, d\omega(z).$$

For $|z| < 1$, we have

$$|f(z)| \leq \sum_{j=0}^{\infty} |z|^j = \frac{1}{1 - |z|}.$$ 

Can show that this implies green part is $O(1)$. 
Borwein-Erdélyi theorem: sketch of proof

- Rearranging yields

\[
\int_{z \text{ blue}} \log |f(z)| \, d\omega(z) \geq -\int_{z \text{ green}} \log |f(z)| \, d\omega(z).
\]

- For $|z| < 1$, we have

\[
|f(z)| \leq \sum_{j=0}^{\infty} |z|^j = \frac{1}{1 - |z|}.
\]

- Can show that this implies green part is $O(1)$.

- This means $\log |f(z)|$ must be at least $e^{-O(L)}$ somewhere on blue part, or else the integral over blue part is too negative.
The Borwein-Erdelyi theorem is sharp. As shown in [NP] and [DOS], this implies that for some $c > 0$ and all $n$ large enough, there exist input strings $x, x'$ of length $n$ such that the corresponding outputs satisfy $|E y_j - E y'_j| < e^{-cn^{1/3}}$ for all $j$. Thus if $T = e^{o(n^{1/3})}$, then we cannot distinguish between $x, x'$ by a linear test. However, the existence of such a pair $x, x'$ is proved via a pigeonhole argument, and we are unable to produce them explicitly.
Reconstruction of random strings
From now on, fix $q < 1/2$ and write $p = 1 - q > 1/2$. 
From now on, fix $q < 1/2$ and write $p = 1 - q > 1/2$.

Given a trace $y$, figure out roughly which position in $y$ corresponds to last reconstructed position so far. Two steps:

1. Greedy matching: try to fit $y$ as a subsequence of $x$, gets within $\log n$.
2. Aligning subsequences: analyze subsequences more carefully to align within $\log 1/2 n$.

Use bit statistics as before to reconstruct next several bits. However, alignment is not exact! But approach can be modified to tolerate random shifts. Can only tolerate a small amount of shifting, hence need to align accurately.
From now on, fix $q < 1/2$ and write $p = 1 - q > 1/2$.

Given a trace $y$, figure out roughly which position in $y$ corresponds to last reconstructed position so far. Two steps:

- Greedy matching: try to fit $y$ as a subsequence of $x$, gets within $\log n$. 

Using bit statistics as before to reconstruct next several bits. However, alignment is not exact! But approach can be modified to tolerate random shifts. Can only tolerate a small amount of shifting, hence need to align accurately.
From now on, fix $q < 1/2$ and write $p = 1 - q > 1/2$.

Given a trace $y$, figure out roughly which position in $y$ corresponds to last reconstructed position so far. Two steps:

- Greedy matching: try to fit $y$ as a subsequence of $x$, gets within $\log n$.
- Aligning subsequences: analyze subsequences more carefully to align within $\log^{1/2} n$.

Use bit statistics as before to reconstruct next several bits. However, alignment is not exact! But approach can be modified to tolerate random shifts. Can only tolerate a small amount of shifting, hence need to align accurately.
From now on, fix $q < 1/2$ and write $p = 1 - q > 1/2$.

Given a trace $y$, figure out roughly which position in $y$ corresponds to last reconstructed position so far. Two steps:

- **Greedy matching**: try to fit $y$ as a subsequence of $x$, gets within $\log n$.
- **Aligning subsequences**: analyze subsequences more carefully to align within $\log^{1/2} n$.

Use bit statistics as before to reconstruct next several bits.
From now on, fix $q < 1/2$ and write $p = 1 - q > 1/2$.

Given a trace $y$, figure out roughly which position in $y$ corresponds to last reconstructed position so far. Two steps:

- Greedy matching: try to fit $y$ as a subsequence of $x$, gets within $\log n$.
- Aligning subsequences: analyze subsequences more carefully to align within $\log^{1/2} n$.

Use bit statistics as before to reconstruct next several bits. However, alignment is not exact! But approach can be modified to tolerate random shifts.
From now on, fix $q < 1/2$ and write $p = 1 - q > 1/2$.

Given a trace $y$, figure out roughly which position in $y$ corresponds to last reconstructed position so far. Two steps:

- Greedy matching: try to fit $y$ as a subsequence of $x$, gets within $\log n$.
- Aligning subsequences: analyze subsequences more carefully to align within $\log^{1/2} n$.

Use bit statistics as before to reconstruct next several bits. However, alignment is not exact! But approach can be modified to tolerate random shifts.

Can only tolerate a small amount of shifting, hence need to align accurately.
Suppose we see both the input $x$ and the output $y$. We still don’t know which bits came from where.
Greedy matching

Suppose we see both the input $x$ and the output $y$. We still don’t know which bits came from where.

Nevertheless, we can try to fit $y$ as a subsequence of $x$. 

```
1 1 0 0 0 1 1 0
```

```
1 0 1 0
```
Greedy matching

Suppose we see both the input $x$ and the output $y$. We still don’t know which bits came from where.

Nevertheless, we can try to fit $y$ as a subsequence of $x$.

Simple approach: just map bits in $y$ “greedily” to the first possible match in $x$. 
Greedy matching

Suppose we see both the input $x$ and the output $y$. We still don’t know which bits came from where.

Nevertheless, we can try to fit $y$ as a subsequence of $x$.

Simple approach: just map bits in $y$ “greedily” to the first possible match in $x$. 
Suppose we see both the input $x$ and the output $y$. We still don’t know which bits came from where.

Nevertheless, we can try to fit $y$ as a subsequence of $x$.

Simple approach: just map bits in $y$ “greedily” to the first possible match in $x$. 
Greedy matching

Suppose we see both the input $x$ and the output $y$. We still don’t know which bits came from where.

Nevertheless, we can try to fit $y$ as a subsequence of $x$.

Simple approach: just map bits in $y$ “greedily” to the first possible match in $x$. 
Suppose we see both the input $x$ and the output $y$. We still don’t know which bits came from where.

Nevertheless, we can try to fit $y$ as a subsequence of $x$.

Simple approach: just map bits in $y$ “greedily” to the first possible match in $x$. 
Greedy matching

Suppose we see both the input $x$ and the output $y$. We still don’t know which bits came from where.

Nevertheless, we can try to fit $y$ as a subsequence of $x$.

Simple approach: just map bits in $y$ “greedily” to the first possible match in $x$. 
Greedy matching

Suppose we see both the input $x$ and the output $y$. We still don’t know which bits came from where.

Nevertheless, we can try to fit $y$ as a subsequence of $x$.

Simple approach: just map bits in $y$ “greedily” to the first possible match in $x$. 

![Diagram showing the mapping of bits from $y$ to $x$.]
Suppose we see both the input $x$ and the output $y$. We still don’t know which bits came from where.

Nevertheless, we can try to fit $y$ as a subsequence of $x$.

Simple approach: just map bits in $y$ “greedily” to the first possible match in $x$. 
Suppose we see both the input $x$ and the output $y$. We still don’t know which bits came from where.

Nevertheless, we can try to fit $y$ as a subsequence of $x$.

Simple approach: just map bits in $y$ “greedily” to the first possible match in $x$. 
Suppose we see both the input $x$ and the output $y$. We still don’t know which bits came from where.

Nevertheless, we can try to fit $y$ as a subsequence of $x$.

Simple approach: just map bits in $y$ “greedily” to the first possible match in $x$. 
Suppose we see both the input $x$ and the output $y$. We still don’t know which bits came from where.

Nevertheless, we can try to fit $y$ as a subsequence of $x$.

Simple approach: just map bits in $y$ “greedily” to the first possible match in $x$. 

Suppose we see both the input $x$ and the output $y$. We still don’t know which bits came from where.

Nevertheless, we can try to fit $y$ as a subsequence of $x$.

Simple approach: just map bits in $y$ “greedily” to the first possible match in $x$. 
Greedy matching

Suppose we see both the input $x$ and the output $y$. We still don’t know which bits came from where.

Nevertheless, we can try to fit $y$ as a subsequence of $x$.

Simple approach: just map bits in $y$ “greedily” to the first possible match in $x$. 
Greedy matching

- The "true location" (gray arrows) advances like a geometric with mean $1/p$.

- The location given by greedy algorithm (red arrows) advances like a geometric with mean $2 > 1/p$, capped at hitting the true location.

- Gap between true and greedy location is like a random walk biased towards zero $\Rightarrow$ stays $O(\log n)$ over the course of the length-$n$ string.
The “true location” (gray arrows) advances like a geometric with mean $1/p$.

The location given by greedy algorithm (red arrows) advances like a geometric with mean $2 > 1/p$, capped at hitting the true location.
Greedy matching

- The “true location” (gray arrows) advances like a geometric with mean $1/p$.
- The location given by greedy algorithm (red arrows) advances like a geometric with mean $2 > 1/p$, capped at hitting the true location.
- Gap between true and greedy location is like a random walk biased towards zero.
Greedy matching

- The “true location” (gray arrows) advances like a geometric with mean $1/p$.
- The location given by greedy algorithm (red arrows) advances like a geometric with mean $2 > 1/p$, capped at hitting the true location.
- Gap between true and greedy location is like a random walk biased towards zero $\implies$ stays $O(\log n)$ over the course of the length-$n$ string.
To get subpolynomial, need to align more precisely than $\log n$. 

Consider a block of length $\log n$ and focus on the middle $a = \log \frac{1}{2}(n)$ bits. After deletion channel, it becomes a subsequence of length $\approx pa$. But could this subsequence come from elsewhere (bad event)?
Aligning by subsequences

To get subpolynomial, need to align more precisely than $\log n$.

Consider a block of length $\log n$ and focus on the middle $a := \log^{1/2}(n)$ bits.

\[a = \log^{1/2}(n)\]
To get subpolynomial, need to align more precisely than $\log n$.

Consider a block of length $\log n$ and focus on the middle $a := \log^{1/2}(n)$ bits.

After deletion channel, it becomes a subsequence of length $\approx pa$. 

\[ a = \log^{1/2}(n) \]
Aligning by subsequences

- To get subpolynomial, need to align more precisely than \( \log n \).
- Consider a block of length \( \log n \) and focus on the middle \( a := \log^{1/2}(n) \) bits.
- After deletion channel, it becomes a subsequence of length \( \approx pa \).
- But could this subsequence come from elsewhere (bad event)?
Aligning by subsequences

Pick $b$ such that $(1 + \epsilon)a < b < (2 - \epsilon)pa$. 

- Only $pa$ bits are retained from a block of length $b$.
- A random string of length $< b$ has $pa$ string as a substring.

#1 only depends on randomness of deletion, not input
#2 only depends on randomness of input, not deletion

Y. Peres (MSR)
Trace reconstruction for deletion channels
June 18, 2018 23 / 28
Pick $b$ such that $(1 + \epsilon)a < b < (2 - \epsilon)pa$.

Bad event covered by two unlikely events (of probability $\approx e^{-\text{const} \cdot a}$).
Aligning by subsequences

- Pick $b$ such that $(1 + \epsilon)a < b < (2 - \epsilon)pa$.
- Bad event covered by two unlikely events (of probability $\approx e^{-\text{const} \cdot a}$)
  1. Only $pa$ bits are retained from a block of length $> b$. 
- Pick $b$ such that $(1 + \epsilon)a < b < (2 - \epsilon)pa$.
- Bad event covered by two unlikely events (of probability $\approx e^{-\text{const} \cdot a}$)
  1. Only $pa$ bits are retained from a block of length $> b$.
  2. A random string of length $< b$ has a specific length $pa$ string as a substring.
Aligning by subsequences

- Pick $b$ such that $(1 + \epsilon)a < b < (2 - \epsilon)pa$.
- Bad event covered by two unlikely events (of probability $\approx e^{-\text{const} \cdot a}$)
  1. Only $pa$ bits are retained from a block of length $> b$.
  2. A random string of length $< b$ has a specific length $pa$ string as a substring.
- #1 only depends on randomness of deletion, not input
Aligning by subsequences

- Pick $b$ such that $(1 + \epsilon)a < b < (2 - \epsilon)pa$.
- Bad event covered by two unlikely events (of probability $\approx e^{-\text{const} \cdot a}$)
  1. Only $pa$ bits are retained from a block of length $> b$.
  2. A random string of length $< b$ has a specific length $pa$ string as a substring.
- #1 only depends on randomness of deletion, not input
- #2 only depends on randomness of input, not deletion
By “most” we will mean all but $e^{-\text{const} \cdot a}$.
By “most” we will mean all but $e^{-\text{const} \cdot a}$.

We say an input is **good** if most length-$pa$ subsequences of its middle $a$ bits cannot be found elsewhere as subsequences of blocks of length $b$. 
Aligning by subsequences

- By “most” we will mean all but $e^{-\text{const} \cdot a}$. 
- We say an input is **good** if most length-$pa$ subsequences of its middle $a$ bits cannot be found elsewhere as subsequences of blocks of length $b$.
- For good input, we can align to the middle $a$ bits by finding a subsequence of length $pa$. 
By “most” we will mean all but $e^{-\text{const} \cdot a}$.

We say an input is **good** if most length-$pa$ subsequences of its middle $a$ bits cannot be found elsewhere as subsequences of blocks of length $b$.

For good input, we can align to the middle $a$ bits by finding a subsequence of length $pa$.

Most inputs are good.
Greedy matching can align to within $\log n$. 

- In a typical random block of length $\log n$, can align to within $\log \frac{1}{2} n$. 
- But this fails in a fraction $e^{-\text{const}} \cdot \log \frac{1}{2} n \gg \frac{1}{n}$ of blocks. 
- Not all blocks will be good, but among $\log \frac{1}{2} n$ consecutive blocks, there will (most likely) be a good one.
Greedy matching can align to within \( \log n \).

In a typical random block of length \( \log n \), can align to within \( \log^{1/2} n \). But this fails in a fraction \( e^{-\text{const} \cdot \log^{1/2} n} \gg \frac{1}{n} \) of blocks.
Greedy matching can align to within $\log n$.

In a typical random block of length $\log n$, can align to within $\log^{1/2} n$. But this fails in a fraction $e^{-\text{const} \cdot \log^{1/2} n} \gg \frac{1}{n}$ of blocks.

Not all blocks will be good, but among $\log^{1/2} n$ consecutive blocks there will (most likely) be a good one.
• Recall: using bit statistics we can recover $m$ bits using $e^{O(m^{1/3})}$ traces.
Putting it all together

Recall: using bit statistics we can recover $m$ bits using $e^{O(m^{1/3})}$ traces.

Modification of proof allows us to tolerate random shifts by $O(m^{1/3})$. 
Recall: using bit statistics we can recover $m$ bits using $e^{O(m^{1/3})}$ traces.

Modification of proof allows us to tolerate random shifts by $O(m^{1/3})$.

Can align to within $\log^{1/2} n$ and want to reconstruct $\log^{3/2} n$ bits ahead.
Putting it all together

- Recall: using bit statistics we can recover $m$ bits using $e^{O(m^{1/3})}$ traces.
- Modification of proof allows us to tolerate random shifts by $O(m^{1/3})$.
- Can align to within $\log^{1/2} n$ and want to reconstruct $\log^{3/2} n$ bits ahead.
- Number of traces used is $e^{O(\log^{1/2} n)} = n^{o(1)}$. 
Holden-Pemantle-Peres’17: For arbitrary deletion probability $q \in [0, 1)$ we can reconstruct random strings with $e^{O(\log^{1/3} n)} = n^{o(1)}$ traces. We also allow insertions and substitutions. Further improvement for random strings cannot be obtained without an improvement for worst-case strings.
Alignment with error $\log(n)$

reconstructed bits
unknown bits

$\tilde{w}$

$\log^{5/3} n$

$p \log^{5/3} n$

Was $\tilde{w}$ likely obtained by sending $w$ through the deletion channel?
reconstructed bits unknown bits
\begin{align*}
\text{w} & \quad \text{log}^{5/3} n \\
\tilde{\text{w}} & \quad \text{log}^{5/3} n \\
p \log^{5/3} n
\end{align*}

\begin{align*}
\text{w} & \quad \text{log}^{2/3} n \\
\tilde{\text{w}} & \quad \text{log}^{2/3} n \\
p \log^{2/3} n
\end{align*}

- Was \( \tilde{\text{w}} \) likely obtained by sending \( \text{w} \) through the deletion channel?
  
  - Divide \( \tilde{\text{w}} \) and \( \text{w} \) into \( \log n \) blocks.
  
  - Let \( S \) be the number of corresponding blocks in \( \tilde{\text{w}} \) and \( \text{w} \) with the same majority bit.
  
  - Answer \textbf{YES} if \( S > (1/2 + c) \log n \); answer \textbf{NO} otherwise.
Was $\tilde{w}$ likely obtained by sending $w$ through the deletion channel?

- Divide $\tilde{w}$ and $w$ into $\log n$ blocks.
- Let $S$ be the number of corresponding blocks in $\tilde{w}$ and $w$ with the same majority bit.
- Answer **YES** if $S > (1/2 + c)\log n$; answer **NO** otherwise.
- Repeat with all strings $\tilde{w}$ of appropriate length.
Alignment with error $\log(n)$

- Was $\tilde{w}$ likely obtained by sending $w$ through the deletion channel?
  - Divide $\tilde{w}$ and $w$ into $\log n$ blocks.
  - Let $S$ be the number of corresponding blocks in $\tilde{w}$ and $w$ with the same majority bit.
  - Answer **YES** if $S > (1/2 + c) \log n$; answer **NO** otherwise.
- Repeat with all strings $\tilde{w}$ of appropriate length.
- Alignment error $O(\log n)$ with probability $1 - \exp(-\Omega(\log^{1/3} n))$. 
Was $\tilde{w}$ likely obtained by sending $w$ through the deletion channel?

- Divide $\tilde{w}$ and $w$ into $\log n$ blocks.
- Let $S$ be the number of corresponding blocks in $\tilde{w}$ and $w$ with the same majority bit.
- Answer **YES** if $S > (1/2 + c) \log n$; answer **NO** otherwise.
- Repeat with all strings $\tilde{w}$ of appropriate length.
- Alignment error $O(\log n)$ with probability $1 - \exp(-\Omega(\log^{1/3} n))$.
- Alignment error improved with second refined alignment step.