Trace Reconstruction Revisited

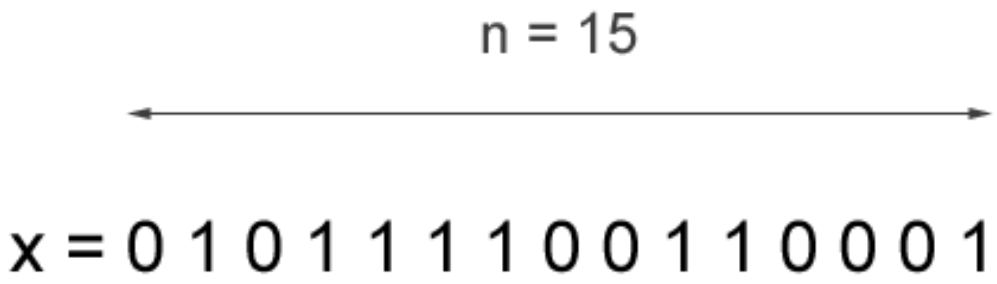
Andrew McGregor[1], Eric Price[2], Sofya Vorotnikova[1]

[1] University of Massachusetts Amherst
[2] IBM Almaden Research Center

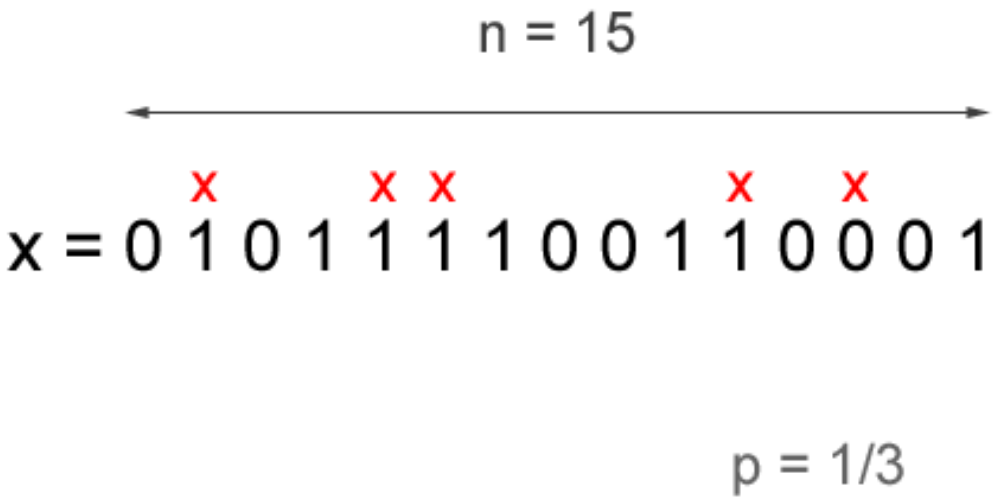
Problem Description

- Take **original string** x of length n. Randomly delete each character with probability p to obtain a **trace** - random subsequence of x.
- Find an algorithm that reconstructs x with high probability using the minimal number of independently obtained traces.
- Two cases:
  - **random** - probability of success is over both the choice of x and the deletions
  - **arbitrary** - probability over deletions only
- Previous work is on binary strings, however we also look at strings over the alphabet of size $\Theta(\log n)$.

Problem Description

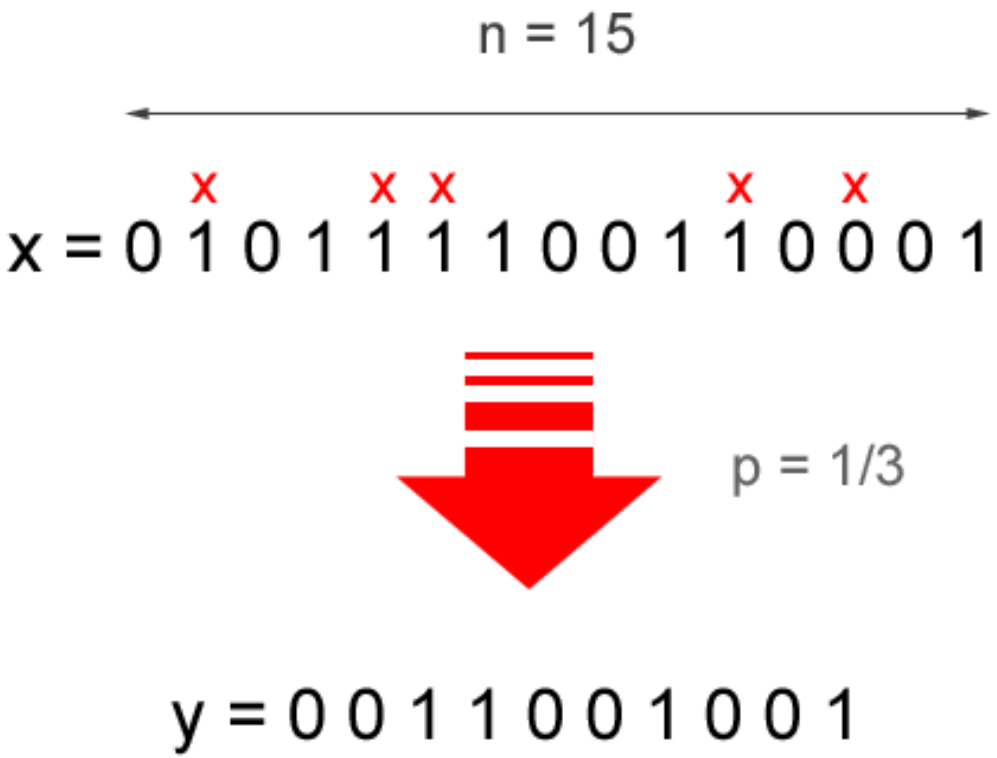


Problem Description



p = 1/3

Problem Description



Reconstructing Random Strings

p = deletion probability

m = number of traces

p	$O(1/\log n)$	small constant
m	$O(\log n)$ Batu et al.[1]	$O(poly(n))$ Holenstein et al.[2] $O(\exp(\sqrt{\log n} \cdot poly(\log\log n)))$ larger alphabet $\Omega(\log^2(n))$

Our main results for constant deletion probability:

- Sub-polynomial number of traces is sufficient if we increase the alphabet size to $\Theta(\log n)$.
- Super-logarithmic number of traces is necessary for binary x.

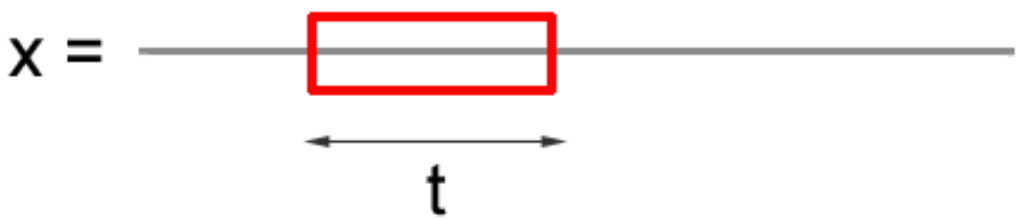
Warm-up Result

If deletion probability is $p = O(\frac{1}{\log n})$, random string x can be reconstructed using $O(\log n)$ traces.

Instead of reconstructing x left to right (as in previous work), we find all sufficiently long substrings of x independently.

Definition

A t-**substring** of x is a string consisting of t consecutive characters of x.



Algorithm Idea

- Take $t = O(\log n)$.
- Go through all t-substrings in the traces.
- Record t-substrings that appear in at least $3/4$ of all traces.
  - *Lemma 1*: Those substrings are exactly the t-substrings of x.
- Reconstruct x from the t-substrings.
  - *Lemma 2*: The set of all t-substrings uniquely defines x.

Lemma 1: Proof Idea

Lemma 1: w is a t-substring of $x \iff w$ is a t-substring of at least $3/4$ of the traces.

Proof: Observe that since p is small, any sufficiently short substring of x is present in $3/4$ fraction of the traces.

$\Rightarrow$: Follows from the observation.

$\Leftarrow$: All matches of w come from the same local region of x whp. But any small region of x appears intact in $3/4$ fraction of the traces. So w must be in x.

Lemma 2: Proof Idea

Lemma 2: The set of all t-substrings of x uniquely defines x.

Proof: Since x is random, if $t = c \log n$, all t-substrings of x are unique whp for sufficiently large c.

- Take a t-substring w_0.
- Find w_i that matches first $t-1$ bits of w_0 and w_j that matches last $t-1$ bits of w_0.
- Continue building up the string on both ends.

Combining t-substrings - Example

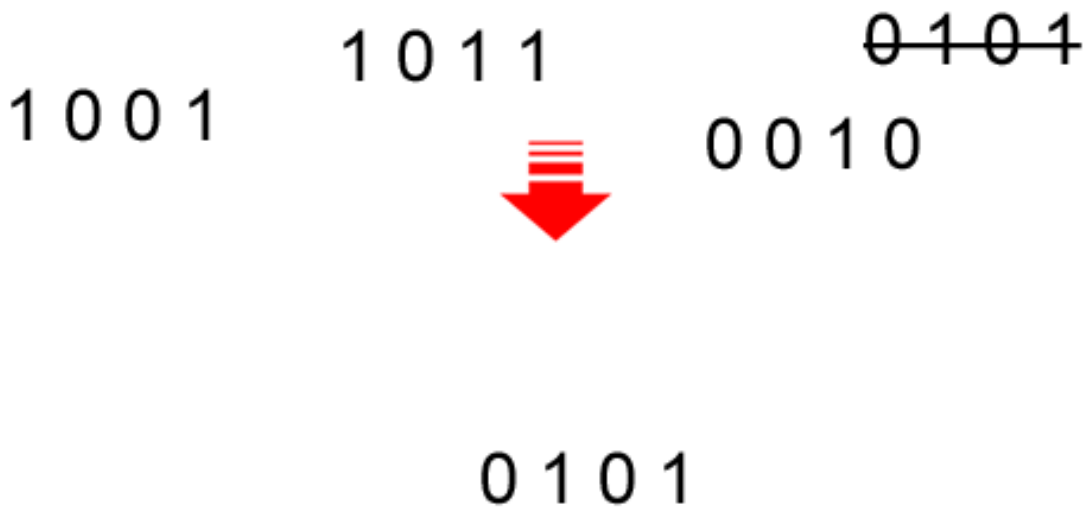
1 0 0 1

1 0 1 1

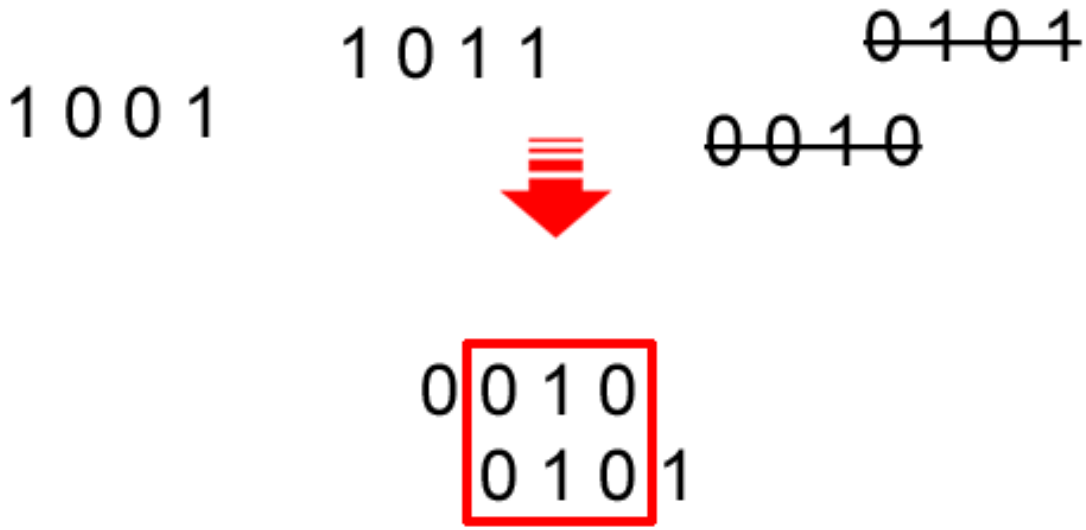
0 1 0 1

0 0 1 0

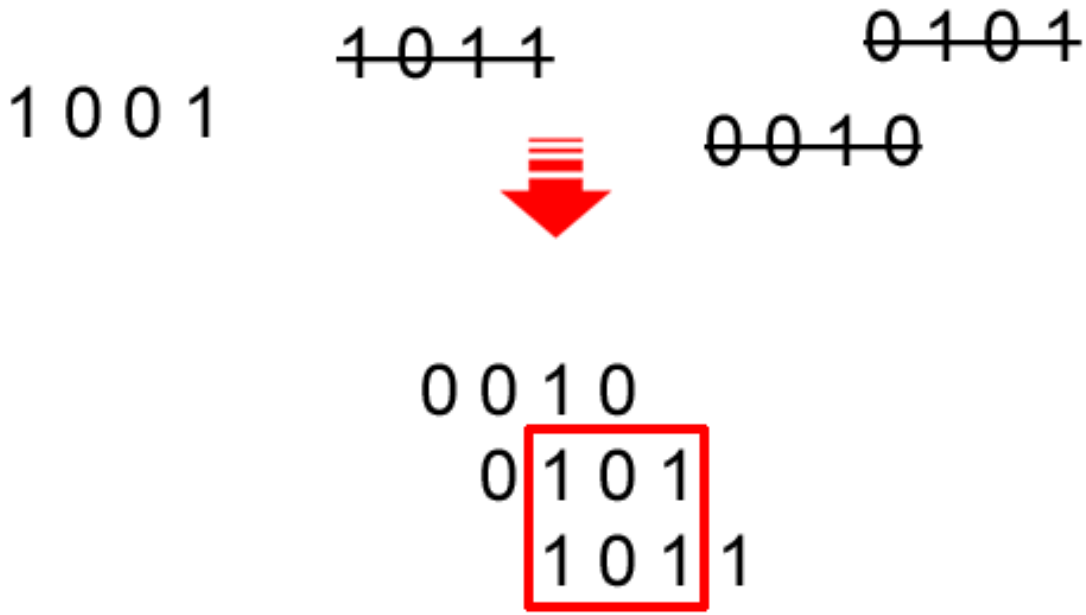
Combining t-substrings - Example



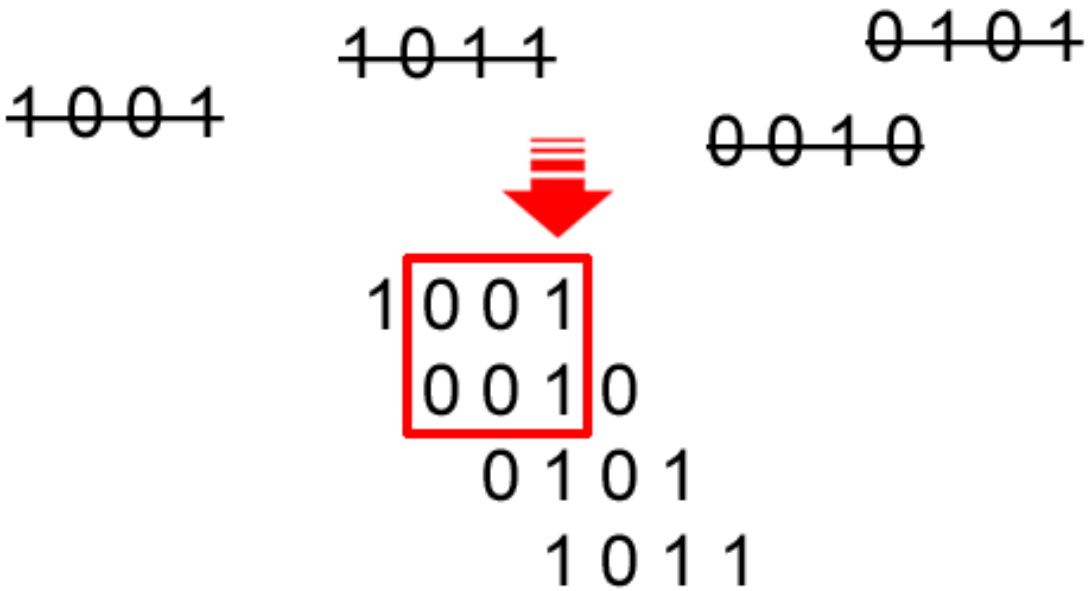
Combining t-substrings - Example



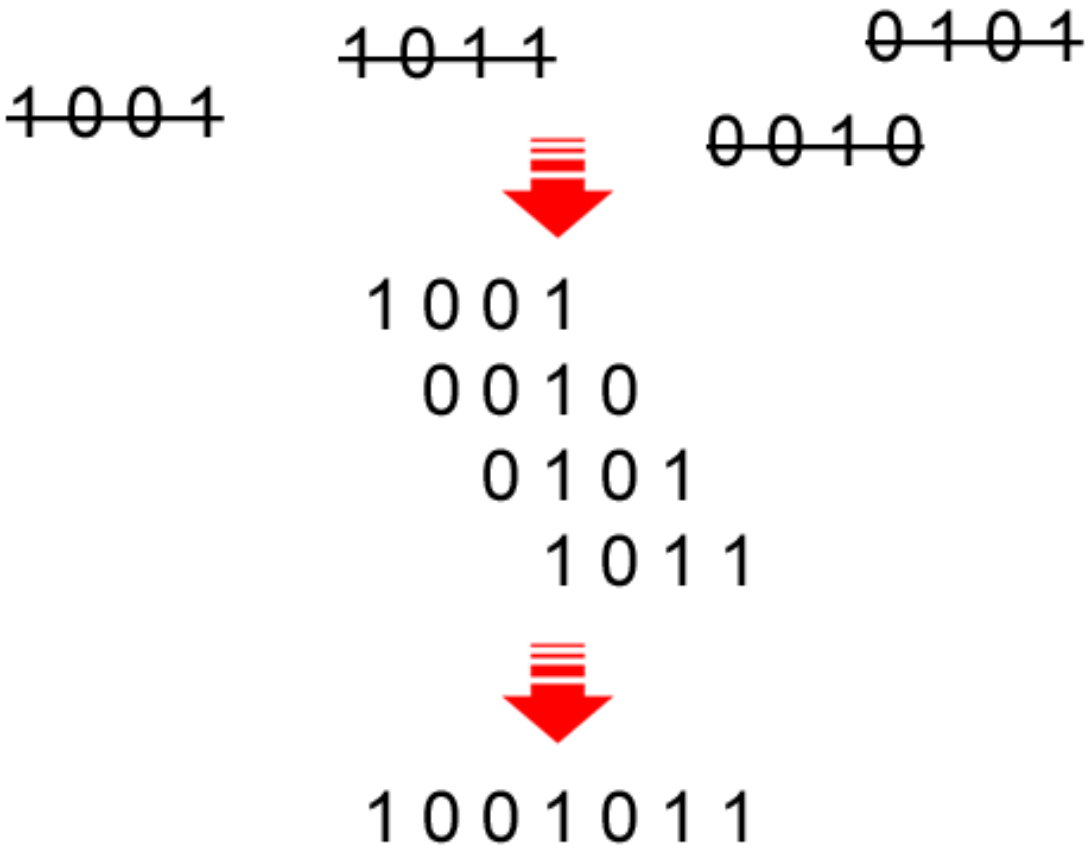
Combining t-substrings - Example



Combining t-substrings - Example



Combining t-substrings - Example



Result 1: Increasing the Alphabet Size

Increase the alphabet size to $\Theta(\log n)$. Consider p to be a small constant. We show that x can be reconstructed from $O(\exp(\sqrt{\log n} \cdot poly(\log \log n)))$ traces.

Previously, we looked for identical t-substrings, but now we have to consider pairs of substrings with a long LCS.

Similarly to the warm-up result, we can show that:

- *Lemma 3*: If two t-substrings in different traces match, they come from the same local region of x.
- *Lemma 4*: Any t-substring w_1 from the first trace has a *matching* t-substring w_i in each trace i. I.e., $lcs(w_1, w_i) \geq 0.99t$.

Lemma 3: Proof Idea

Lemma 3: If two t-substrings in different traces match, they come from the same local region of x.

Proof:

- Let u and v be t-substrings of different traces. Suppose they come from different regions of x.
- Since x is random, u and v are independent random strings.
- Therefore, $Pr[lcs(u, v) \geq 0.99t] < \binom{t}{0.99t}^2 (1/2)^{0.99t} < 1/poly(n)$ since $t = O(\log n)$.

Lemma 4: Proof Idea

Lemma 4: Any t-substring w_1 from the first trace has a matching t-substring w_i in each trace i.

Proof:

- Expected number of deletions in a t-substring of x is pt. By Chernoff, the number of deletions is smaller than $2pt$ whp.
- Therefore, there are at least $(1-2p)t$ characters of some t-substring u of x in each w_i.
- Then, $lcs(w_1, w_i) \geq (1-4p)t \geq 0.99t$ for sufficiently small constant p.

The advantage of having a larger alphabet is in the existence of *useful* characters.

Definition

We say a character from x is a **useful character** if:

- It was not deleted in the first trace and appears in some t-substring w.
- It is locally unique, i.e., appears at most once in w and in each of the matches of w in other traces.
- It appears in most matches of w.

Algorithm Idea

- Partition the first trace into t-substrings of length $O(\log n)$.
- For each t-substring w of the first trace, identify its matches in other traces.
- In each set of matches identify a useful character.
- Consider the part of each trace between two consecutive useful characters. Use those to reconstruct a part of x by employing an algorithm for arbitrary string reconstruction.

Example

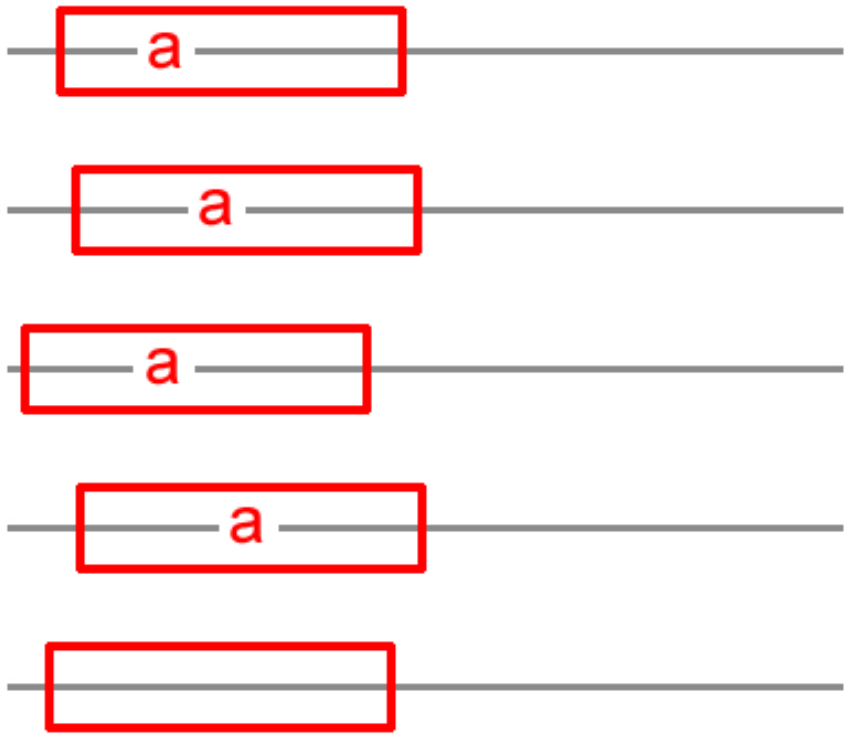
Take a t-substring in the first trace.

Example

Find matching t-substrings in other traces.

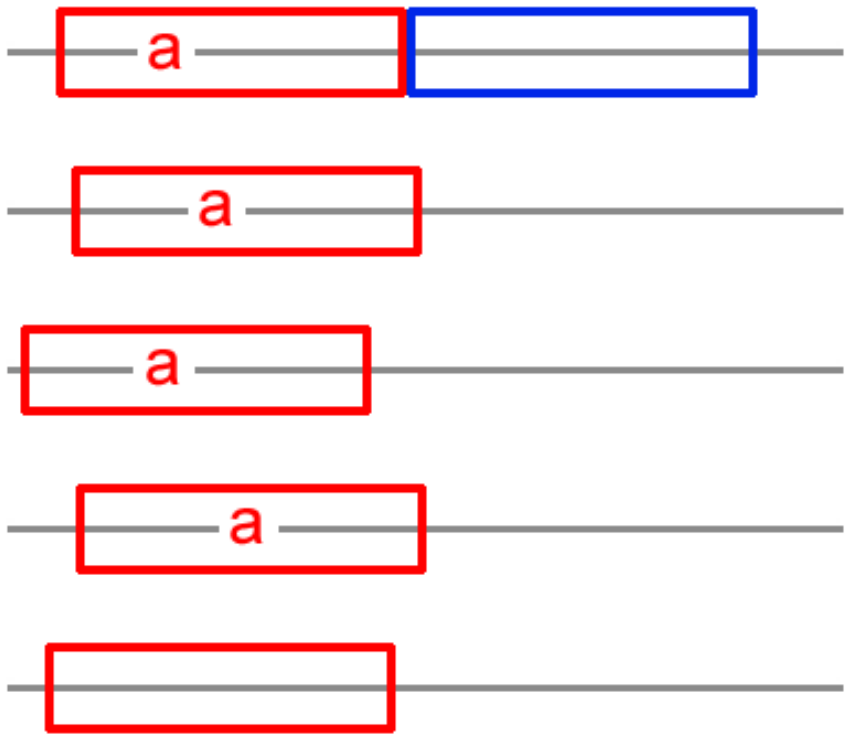
Example

Find a useful character.



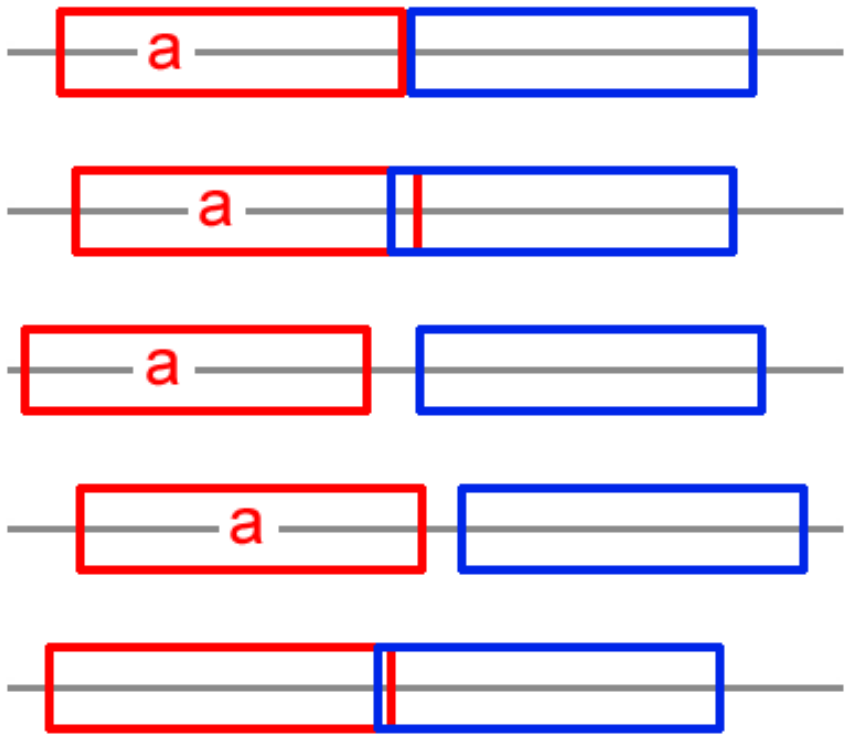
Example

Repeat for the next t-substring.



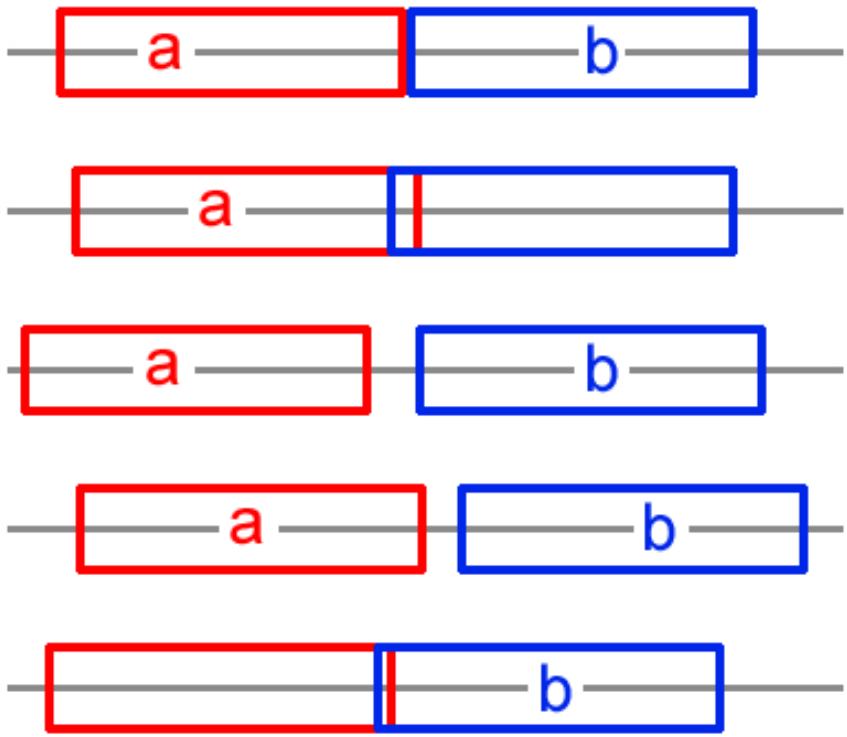
Example

Repeat for the next t-substring.



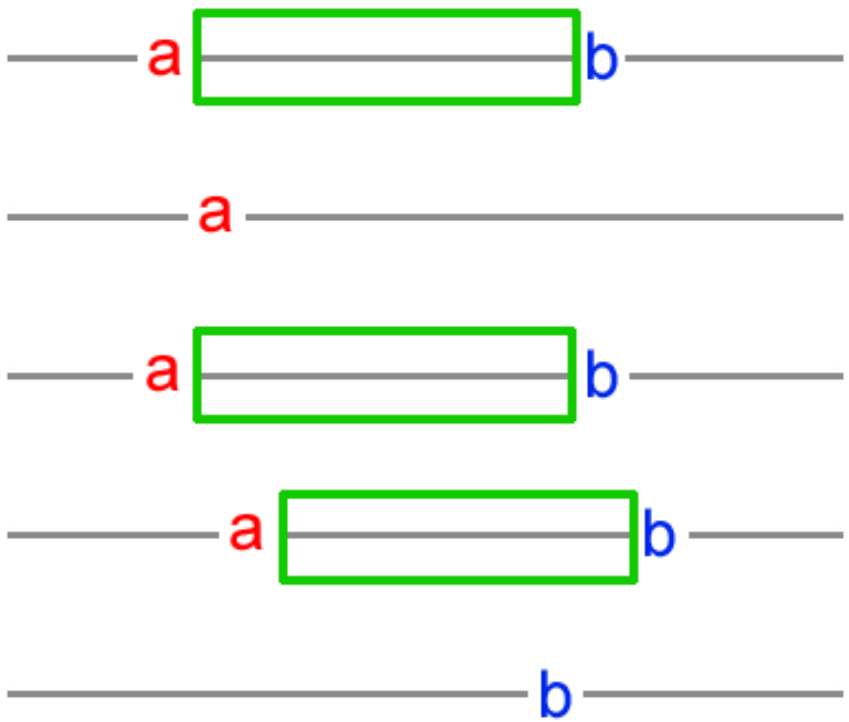
Example

Repeat for the next t-substring.



Example

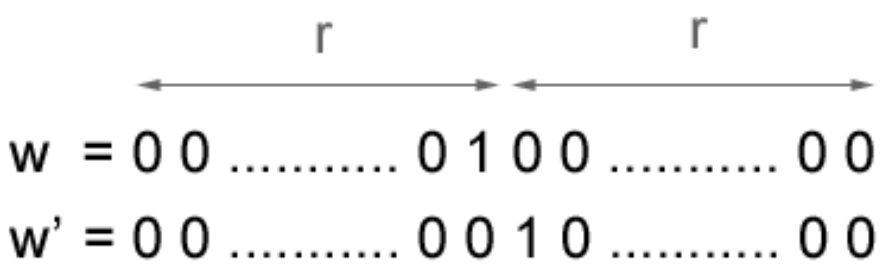
Consider parts of traces between useful characters.



Result 2: Lower Bound

Reconstructing a random binary string of length n requires $\Omega(\log^2 n)$ traces for constant $p > 0$.

We introduce two specific binary strings of length $2r$ where $r = O(\log n)$:



Lower Bound: Proof Idea

- Distinguishing w and w' with probability greater than $1 - \delta$ requires $\Omega(r \log(1/\delta))$ traces.
- Each of w and w' occur $n^{\Omega(1)}$ times in a random binary string of length n.
- Thus δ needs to be inversely polynomial in n, otherwise one of the occurrences of w will be confused with an occurrence of w'.
- Therefore, number of traces required is $\Omega(\log^2 n)$.

Reconstructing Arbitrary Strings

p = deletion probability

m = number of traces

p	$O(1/\sqrt{n})$	constant	$\leq 1 - c\sqrt{\frac{\log n}{n}}$
m	$O(n \cdot polylog(n))$ Batu et al.[1]	$O(\exp(\sqrt{n} \cdot polylog(n)))$ Holenstein et al.[2]	$O(\exp(\sqrt{n} \cdot polylog(n)))$

We show that for deletion probability $p \leq 1 - c\sqrt{\frac{\log n}{n}}$, number of traces sufficient to reconstruct x is exponential in $\sqrt{n} \cdot polylog(n)$.

Previous result by Holenstein et al.[2] showed that the same number of traces was sufficient when traces were of length $\Theta(n)$. We show that reconstruction is still possible for trace length $\Theta(\sqrt{n \log n})$.

Result 3: Arbitrary Strings Reconstruction

- Scott[3] shows that you can uniquely determine x from $\langle n_1, n_2, ..., n_k \rangle$, where n_i is the number of subsequences of x of length i ending with 1, if $k = c\sqrt{n \log n}$.
- If we have sufficiently many traces of length greater than k, we can sample subsequences of length i and determine n_i.
- Using Chernoff bound we show that to obtain all of n_i whp we need $\exp(\sqrt{n} \cdot polylog(n))$ traces.

Other Results

Deletions, Insertions, and Substitutions

Consider the following error model:
deletion probability: $p = c/\log n$
insertion probability: $q = c/\log n$
substitution probability: $\alpha = c$

Viswanathan and Swaminathan[4] show that $O(\log n)$ traces are sufficient to reconstruct random x for some small c.

We give a simpler algorithm for this result.

- Split the first trace into t-substrings for $t = O(\log n)$.
- For each t-substring in the first trace, find matches in other traces with Hamming distance at least $3\alpha t$.
- For each set of matches take the mode of each of the t character positions to obtain substrings of x.

Other Results

Relationship between Random and Arbitrary String Reconstruction

Result 1 shows that for an alphabet of size $\Theta(\log n)$ reconstructing arbitrary strings with $f(n)$ traces allows us to reconstruct random strings with $f(\log n)$ traces.

We show that the reverse is also true.

To reconstruct an arbitrary string x of length n:

- Pick random strings a and b of length $O(2^n)$.
- To each trace of x add a trace of a on the left and a trace of b on the right.
- Use random string reconstruction algorithm on padded traces.
- Extract the part corresponding to the original string.

Summary and Open Problems

For random strings and constant deletion probability:

- Sub-polynomial number of traces is sufficient for alphabet size $\Theta(\log n)$. *(Can we reduce the alphabet back to binary while keeping the same number of traces?)*
- Super-logarithmic number of traces is necessary for binary x. *(Can we merge the gap between the upper and lower bounds?)*

For arbitrary strings $\exp(\sqrt{n} \cdot \mathit{polylog}(n))$ traces of length $\Theta(\sqrt{n \log n})$ are sufficient. *(Is polynomial number of traces sufficient for arbitrary strings for constant and larger deletion probabilities?)*

Can current research be extended to other error models? E.g., insertions, substitutions, switching two characters, etc.

Thank you for your attention!

Bibliography

[1] T. Batu, S. Kannan, S. Khanna, and A. McGregor. Reconstructing strings from random traces. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 910-918, 2004.

[2] T. Holenstein, M. Mitzenmacher, R. Panigrahy, and U. Wieder. Trace reconstruction with constant deletion probability and related results. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 389-398, 2008.

[3] A. D. Scott. Reconstructing sequences. *Discrete Mathematics*, 175(1-3):231-238, 1997.

[4] K. Viswanathan and R. Swaminathan. Improved string reconstruction over insertion-deletion channels. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 399-408, 2008.