Sketches and streaming algorithms for string processing

Tatiana Starikovskaya
Streaming model

Objectives: real time & small space
When to use: stream data, big data
Classical approaches won’t work: to answer a question deterministically and exactly, we need to store the input in full.

Relaxations: randomisation + approximation

Tools: sketches (= lossy compression of the data) — capture essential properties of the data
Outline of today’s talk

- Part I: Exact pattern matching
- Part II: String similarity and approximate pattern matching
- Part III: Periodicity
Part I: Exact pattern matching
Exact pattern matching

- **Query** = “Is there an occurrence of $P$?”
- **Space** = total space used by the stream processor
- **Time** = time per position of $T$
Exact pattern matching

- **Query** = “Is there an occurrence of $P$?”
- **Space** = total space used by the stream processor
- **Time** = time per position of $T$
Exact pattern matching

Text $T$

$c\ a\ a\ b\ c\ a\ a\ a$

Pattern $P$

- **Query** = “Is there an occurrence of $P$?”
- **Space** = total space used by the stream processor
- **Time** = time per position of $T$
Exact pattern matching

- **Query** = “Is there an occurrence of $P$?”
- **Space** = total space used by the stream processor
- **Time** = time per position of $T$
Exact pattern matching

- **Query** = “Is there an occurrence of $P$?”
- **Space** = total space used by the stream processor
- **Time** = time per position of $T$
Karp–Rabin algorithm

Karp–Rabin fingerprint

\[ \varphi(s_1s_2 \ldots s_m) = \sum_{i=1}^{m} s_i \cdot r^{m-i} \mod p \]

where \( p \) is a prime and \( r \) is a random integer \( \in [0, p - 1] \)

It’s a good hash function:
\( S_1, S_2 \) are two strings of length \( m \), the prime \( p \) is large

- If \( S_1 = S_2 \), then \( \varphi(S_1) = \varphi(S_2) \)
- If \( S_1 \neq S_2 \), then \( \varphi(S_1) \neq \varphi(S_2) \) w.h.p.
Karp–Rabin algorithm

When a new character $t_i = a$ arrives:

1. Update $\varphi(t_{i-m+1} \ldots t_{i-1}t_i) = \sum_{j=1}^{m} t_{i-m+j} \cdot r^{m-j} \mod p$:

$$\varphi(t_{i-m+1} \ldots t_{i-1}t_i) = (\varphi(t_{i-m} \ldots t_{i-1}t_{i-1}) - t_{i-m} \cdot r^{m-1}) \cdot r + t_i \mod p$$

2. If $\varphi(t_{i-m+1} \ldots t_{i-1}t_i) = \varphi(P)$, output “YES”

We need $t_{i-m}$ to update the fingerprint ⇒ we must store $t_{i-m}, \ldots, t_{i-1}$
## Exact pattern matching

<table>
<thead>
<tr>
<th>Authors</th>
<th>Space ¹</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single pattern</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Karp &amp; Rabin, 1987</td>
<td>$\Theta(m)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Porat &amp; Porat, FOCS’09</td>
<td>$O(\log m)$</td>
<td>$O(\log m)$</td>
</tr>
<tr>
<td>Breslauer &amp; Galil, CPM’11</td>
<td>$O(\log m)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td><strong>Dictionary of $d$ patterns</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clifford, Fontaine, Porat</td>
<td>$O(d \log m)$</td>
<td>$O(\log \log (m + d))$</td>
</tr>
<tr>
<td>Sach, S., ESA’15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Golan &amp; Porat, ESA’17</td>
<td>$O(d \log m)$</td>
<td>$O(\log \log</td>
</tr>
<tr>
<td></td>
<td>$O(</td>
<td>\Sigma</td>
</tr>
</tbody>
</table>

¹In words
Exact pattern matching

<table>
<thead>
<tr>
<th>Authors</th>
<th>Space $^1$</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single pattern</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Karp &amp; Rabin, 1987</td>
<td>$\Theta(m)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Porat &amp; Porat, FOCS’09 ★</td>
<td>$O(\log m)$</td>
<td>$O(\log m)$</td>
</tr>
<tr>
<td>Breslauer &amp; Galil, CPM’11</td>
<td>$O(\log m)$</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>

| Dictionary of $d$ patterns           |                       |       |
| Clifford, Fontaine, Porat Sach, S., ESA’15 | $O(d \log m)$ | $O(\log \log(m + d))$ |
| Golan & Porat, ESA’17                | $O(d \log m)$         | $O(\log \log|\Sigma|)$ |
|                                      | $O(|\Sigma|^{\varepsilon} d \log(m/\varepsilon))$ | $O(1/\varepsilon)$ |

$^1$In words
Fine and Wilf’s periodicity lemma
If a string \( Q \) has two periods of length \( p \) and \( q \) and \( p + q \leq |Q| \), then \( Q \) also has a period of length \( \gcd(p, q) \).

Corollary
If \( |Y| = 2|X| \), and \( Y \) contains \( \geq 3 \) occurrences of \( X \), they form an arithmetic progression with difference equal to the smallest period of \( X \).

Occurrences of \( X \) in \( Y \) can be stored in \( O(1) \) space.
text $T$

occurrences of $p_1$

occurrences of $p_1p_2$

occurrences of $p_1p_2p_3p_4$

occurrences of $P = p_1p_2 \ldots p_m$

for each character $t_i$ do
  if $t_i = p_1$ then push $i$ to level $0$
  for each $j = 0, \ldots, \log m - 1$
    $lp \leftarrow$ leftmost position in level $j$
    if $i - lp + 1 = 2^{j+1}$ then
      Pop $lp$ from level $j$
    if $\varphi(t_{lp} \ldots t_i) = \varphi(p_1 \ldots p_{2^{j+1}})$ then push $lp$ to level $j + 1$
for each character $t_i$ do
  if $t_i = p_1$ then push $i$ to level 0
for each $j = 0, \ldots, \log m - 1$
  $lp \leftarrow$ leftmost position in level $j$
  if $i - lp + 1 = 2^{j+1}$ then
    Pop $lp$ from level $j$
  if $\varphi(t_{lp} \ldots t_i) = \varphi(p_1 \ldots p_{2^{j+1}})$ then push $lp$ to level $j + 1$
text $T$  

occurrences of $p_1$  

occurrences of $p_1p_2$  

occurrences of $p_1p_2p_3p_4$  

occurrences of $P = p_1p_2 \ldots p_m$

for each character $t_i$ do

  if $t_i = p_1$ then push $i$ to level 0

for each $j = 0, \ldots, \log m - 1$

  $lp \leftarrow$ leftmost position in level $j$

  if $i - lp + 1 = 2^{j+1}$ then

    Pop $lp$ from level $j$

  if $\varphi(t_{lp} \ldots t_i) = \varphi(p_1 \ldots p_{2j+1})$ then push $lp$ to level $j + 1$

If $i$ is an occ. of $p_1$, push it to level 0
text $T$

\begin{itemize}
  \item occurrences of $p_1$
  \item occurrences of $p_1p_2$
  \item occurrences of $p_1p_2p_3p_4$
  \item \ldots
  \item occurrences of $P = p_1p_2\ldots p_m$
\end{itemize}

\textbf{for} each character $t_i$ \textbf{do}

\textbf{if} $t_i = p_1$ \textbf{then} push $i$ to level 0

\textbf{for} each $j = 0, \ldots, \log m - 1$

\begin{itemize}
  \item $lp \leftarrow$ leftmost position in level $j$
  \item \textbf{if} $i - lp + 1 = 2^{j+1}$ \textbf{then} Pop $lp$ from level $j$
  \item \textbf{if} $\varphi(t_{lp} \ldots t_i) = \varphi(p_1 \ldots p_{2^{j+1}})$ \textbf{then} push $lp$ to level $j + 1$
\end{itemize}

If $i$ is an occ. of $p_1$, push it to level 0
text $T$

occurrences of $p_1$

occurrences of $p_1p_2$

occurrences of $p_1p_2p_3p_4$

occurrences of $P = p_1p_2 \ldots p_m$

For each character $t_i$ do
  if $t_i = p_1$ then push $i$ to level 0
  for each $j = 0, \ldots, \log m - 1$
    $lp \leftarrow$ leftmost position in level $j$
    if $i - lp + 1 = 2^{j+1}$ then
      Pop $lp$ from level $j$
      if $\varphi(t_{lp} \ldots t_i) = \varphi(p_1 \ldots p_{2j+1})$ then push $lp$ to level $j + 1$

If $lp$ is an occ. of $p_1p_2$, promote it
text $T$
\begin{itemize}
\item \text{occurrences of $p_1$}
\item \text{occurrences of $p_1p_2$}
\item \text{occurrences of $p_1p_2p_3p_4$}
\item \text{occurrences of $P = p_1p_2 \ldots p_m$}
\end{itemize}

\begin{algorithm}
for each character $t_i$ do
  if $t_i = p_1$ then push $i$ to level 0
  for each $j = 0, \ldots, \log m - 1$
    $lp \leftarrow$ leftmost position in level $j$
    if $i - lp + 1 = 2^{j+1}$ then
      Pop $lp$ from level $j$
    if $\varphi(t_{lp} \ldots t_i) = \varphi(p_1 \ldots p_{2^{j+1}})$ then push $lp$ to level $j + 1$
\end{algorithm}

If $lp$ is an occ. of $p_1p_2$, promote it
In level $j$, we store occurrences of $p_1p_2\ldots p_j$ in $T[i - 2^{j+1} + 1, i]$. They form an arithmetic progression. We store:

- Number of occurrences
- The leftmost and the second leftmost positions $lp, lp'$
- The fingerprints $\varphi(t_1t_2\ldots t_{lp}), \varphi(t_{lp+1}\ldots t_{lp'}), \varphi(t_1\ldots t_i)$
text $T$ \hfill $t_i$
\hline
occurrences of $p_1$
\hline
occurrences of $p_1p_2$
\hline
occurrences of $p_1p_2p_3p_4$
\hline
occurrences of $P = p_1p_2 \ldots p_m$
\hline

For each level we need:

- $O(1)$ space
- $O(1)$ time for updating and extracting $\varphi(t_l p \ldots t_i)$

In total, the algorithm uses $O(\log m)$ space and $O(\log m)$ time per character
Part II: String similarity and approximate pattern matching
String similarity

Given two streams $S_1, S_2$ ($S_1$ arrives before $S_2$), compute the distance between them.

Approximate pattern matching

$\text{dist}(P, T)$

$T$: c a a b c a a a a c a

$P$: b c a a a a c

pattern $P$
String similarity (Hamming distance)

Johnson & Lindenstrauss, 1984: one can compute $(1 + \varepsilon)$-approximation of the Hamming distance between two streams using $O(\varepsilon^{-2} \log n)$ space and $O(\varepsilon^{-2} \log n)$ time per character.

Porat & Lipsky, 2007: one can decide if the Hamming distance between two streams is $\leq k$ using $O(k \log n)$ space and $O(\log n)$ time per character.
Approximate pattern matching (Hamming distance)

<table>
<thead>
<tr>
<th>Authors</th>
<th>Space</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single pattern, only distances ≤ k</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Porat &amp; Porat, FOCS’09</td>
<td>$\tilde{O}(k^3)$</td>
<td>$\tilde{O}(k^2)$</td>
</tr>
<tr>
<td>Clifford, Fontaine, Porat, Sach, S., SODA’16</td>
<td>$\tilde{O}(k^2)$</td>
<td>$\tilde{O}(\sqrt{k})$</td>
</tr>
<tr>
<td>Golan, Kopelowitz, Porat, ICALP’18</td>
<td>$\tilde{O}(k)$</td>
<td>$\tilde{O}(k)$</td>
</tr>
<tr>
<td>Clifford, Kociumaka, Porat, SODA’19</td>
<td>$\tilde{O}(k)$</td>
<td>$\tilde{O}(\sqrt{k})$</td>
</tr>
</tbody>
</table>

| Single pattern, $(1 + \varepsilon)$-approx. | | |
| Clifford, S., ICALP’16 | $O(\varepsilon^{-5} \sqrt{m \log^4 m})$ | $O(\varepsilon^{-4} \log^3 m)$ |

| Dictionary of $d$ patterns, only distances ≤ k | | |
| Gawrychowski, S. (submitted) | $\tilde{O}(kd \log^k d)$ | $\tilde{O}(k \log^k d + \text{occ})$ |

$^2$In words
If $\text{HAM}(P,T) > k$, output "NO"

Otherwise, output $\text{HAM}(P,T)$

There is a streaming algorithm that uses $\tilde{O}(k^3)$ space and $\tilde{O}(k^2)$ time per character of the text
From 1 mismatch to exact pattern matching

```
string_1
a b a a c b a b a a b b
```

```
string_2
a b a c c b a b a a a b
```

- Is HAM \((string_1, string_2)\) = 1?
From 1 mismatch to exact pattern matching

- Is $\text{HAM}(\text{string}_1, \text{string}_2) = 1$?
- Partition the strings into substrings of $q$ colors
- One mismatch $\Rightarrow$ one pair of substrings does not match
- **Hope:** If there are $\geq 2$ mismatches, they will end up in substrings of different colors $\Rightarrow$ at least 2 pairs of substrings do not match
From 1 mismatch to exact pattern matching

For each prime $q \in [\log m, \log^2 m]$:
- Partition $string_1$ into $q$ equi-spaced substrings
- Partition $string_2$ into $q$ equi-spaced substrings

In total: $O(\log m)$ primes, and for each prime there are $O(\log^2 m)$ pairs of substrings
From 1 mismatch to exact pattern matching

Lemma There are $\geq 2$ mismatches $\times_1, \times_2 \Rightarrow$ there exists a prime $q$ such that at least two pairs of substrings do not match

- $\times_1, \times_2$ in the same pair $\Leftrightarrow \times_1 - \times_2 = 0 \pmod{q}$
- $m \geq \times_1 - \times_2$ cannot be a multiple of $\log m$ distinct primes
From 1 mismatch to exact pattern matching

Is $\text{HAM}(P, T) = 1$?

for each position of the text $T$ do
    for each prime $q$ in $[\log m, \log^2 m]$ do
        $h \leftarrow$ number of (substream, subpattern) that mismatch
        if $h = 0$ or $h > 1$ return “NO”
    return “YES”
From 1 mismatch to exact pattern matching

Compute number of mismatching pairs

\[ \text{for each prime } q \text{ in } [\log m, \log^2 m] \text{ do } \]
\[ \text{for each (substream, subpattern) do } \]
\[ \text{run streaming exact pattern matching } \]
From 1 mismatch to exact pattern matching

text $T$

pattern $P$

Space = $O\left(\frac{\log m}{\# \text{ of primes}} \cdot \frac{\log^2 m}{\# \text{ of substr.}} \cdot \frac{\log^2 m}{\# \text{ of subpatterns}} \cdot \log m\right)$

Time = $O\left(\frac{\log m}{\# \text{ of primes}} \cdot \frac{\log^2 m}{\# \text{ of substr.}} \cdot \frac{\log^2 m}{\# \text{ of subpatterns}}\right)$

In general: $\tilde{O}(k^3)$ space, $\tilde{O}(k^2)$ time
(same as for $k = 1$ but take more subpatterns)
String similarity (edit distance)

Chakraborty, Goldenberg, Koucky, STOC’16: small distortion embedding from edit to Hamming distance

Belazzougui, Zhang, FOCS’16: embedding-based sketches for computing the edit distance exactly given that it is $\leq k$
Embedding from edit to Hamming distance

Pick $3m$ random functions $h_j : \{0, 1\} \rightarrow \{0, 1\}$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>3n</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Copy letters of $S$ to $\mu(S)$:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th></th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>S :</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>$\mu(S)$ :</td>
<td></td>
<td>...</td>
<td></td>
<td>text position = 1, j = 1</td>
<td></td>
</tr>
</tbody>
</table>

1. Copy $S[i]$. If $h_j(S[i]) = 1$, move to the right;
2. $j = j + 1$. 
Embedding from edit to Hamming distance

Pick $3m$ random functions $h_j : \{0, 1\} \rightarrow \{0, 1\}$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>3n</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Copy letters of $S$ to $\mu(S)$:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$:</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$\mu(S)$:</td>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

text position = 1, $j = 1$

1. Copy $S[i]$. If $h_j(S[i]) = 1$, move to the right;
2. $j = j + 1$. 

Embedding from edit to Hamming distance

Pick $3m$ random functions $h_j : \{0, 1\} \rightarrow \{0, 1\}$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

... ... 0 1

Copy letters of $S$ to $\mu(S)$:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$:</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\mu(S)$:</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

text position = 1, $j = 1$

1. Copy $S[i]$. If $h_j(S[i]) = 1$, move to the right;
2. $j = j + 1$. 
Embedding from edit to Hamming distance

Pick $3m$ random functions $h_j : \{0, 1\} \rightarrow \{0, 1\}$

|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

... ... ... ... 0

... 1

Copy letters of $S$ to $\mu(S)$:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>...</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>$\mu(S)$</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Copy $S[i]$. If $h_j(S[i]) = 1$, move to the right;
2. $j = j + 1$. 

text position = 1, $j = 2$
Embedding from edit to Hamming distance

Pick $3m$ random functions $h_j : \{0, 1\} \rightarrow \{0, 1\}$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th></th>
<th>3n</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Copy letters of $S$ to $\mu(S)$:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$:</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$\mu(S)$:</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>text position = 1, $j = 2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Copy $S[i]$. If $h_j(S[i]) = 1$, move to the right;
2. $j = j + 1$. 

Embedding from edit to Hamming distance

Pick 3m random functions \( h_j : \{0, 1\} \rightarrow \{0, 1\} \)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

... ... 3n

Copy letters of \( S \) to \( \mu(S) \):

\[
S : \begin{array}{llll}
0 & 1 & 0 & \cdots & 0 \\
\end{array}
\]

\[
\mu(S) : \begin{array}{llll}
0 & 0 & \cdots & 0 \\
\end{array}
\]

1. Copy \( S[i] \). If \( h_j(S[i]) = 1 \), move to the right;
2. \( j = j + 1 \).
Embedding from edit to Hamming distance

Pick $3m$ random functions $h_j : \{0, 1\} \rightarrow \{0, 1\}$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>...</th>
<th>3n</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Copy letters of $S$ to $\mu(S)$:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>...</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$:</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$\mu(S)$:</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Copy $S[i]$. If $h_j(S[i]) = 1$, move to the right;
2. $j = j + 1$. 

text position = 2, $j = 3$
Embedding from edit to Hamming distance

Pick $3m$ random functions $h_j : \{0, 1\} \rightarrow \{0, 1\}$

1 2 3 4 5 6 7 8

| 0 | 0 1 1 0 1 1 0 0 |
| 1 | 1 1 1 1 0 1 0 1 |

... ... 3n

| 0 |
| 1 |

Copy letters of $S$ to $\mu(S)$:

| 1 2 3 n |
| S: 0 1 0 ... 0 |
| $\mu(S)$: 0 0 1 |

1. Copy $S[i]$. If $h_j(S[i]) = 1$, move to the right;
2. $j = j + 1$. 

text position = 2, $j = 3$
Embedding from edit to Hamming distance

Pick $3m$ random functions $h_j : \{0, 1\} \rightarrow \{0, 1\}$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>3n</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

... 0

... 1

Copy letters of $S$ to $\mu(S)$:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$:</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>$\mu(S)$:</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

text position = 2, $j = 3$

1. Copy $S[i]$. If $h_j(S[i]) = 1$, move to the right;
2. $j = j + 1$. 
Embedding from edit to Hamming distance

Pick $3m$ random functions $h_j : \{0, 1\} \rightarrow \{0, 1\}$

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>3n</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

... 0

... 1

Copy letters of $S$ to $\mu(S)$:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>...</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>S : 0</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>$\mu(S)$ : 0</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

text position = 2, $j = 3$

When the length of $\mu(S)$ reaches $3n$, stop. If the length of $\mu(S) < 3n$, append with zeros.

**Theorem.** If $ED(S, T) = k$, then $k/2 \leq HD(\mu(S), \mu(T)) \leq \mathcal{O}(k^2)$ with probability 0.99.
Embedding from edit to Hamming distance

Pick $3m$ random functions $h_j : \{0, 1\} \rightarrow \{0, 1\}$

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>...</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>1</td>
</tr>
</tbody>
</table>

Copy letters of $S$ to $\mu(S)$:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>S : 0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\mu(S)$ : 0</td>
<td>0</td>
<td>1</td>
<td>...</td>
</tr>
</tbody>
</table>

text position = 2, $j = 3$

Belazzougui, Zhang, FOCS'16

- Embedding + streaming alg’m for $k^2$-mismatch ⇒ a good estimate for edit distance
- If $ED(S, T) \leq k$, $\tilde{O}(k^2)$ embeddings + streaming alg’m for $k^2$-mismatch ⇒ exact value!
Approximate pattern matching (edit distance)

Given a pattern $P$ and a text $T$, the edit distance $ED(P, T)$ is defined as:

- If $ED(P, S) > k$, output "NO"
- Otherwise, output $ED(P, S)$

Algorithms:

- Hybrid dynamic programming: $O(m)$ space, $O(k)$ time
- S., 2017: $O(\sqrt{m} \cdot poly(k, \log m))$ space, $O(\sqrt{m} \cdot poly(k, \log m))$ time
Approximate pattern matching (edit distance)

Starting from each block $i$, run Belazzougui & Zhang, 2016

$$ED[j] = \min_{i \in [r-k, r+k]} ED(P[1, B - i], T_1) + ED(P[B - i + 1, m], T_2)$$

We compute $ED(P[1, B - i], T_1)$ while reading $T_1$ using dynamic programming, then encode the distances to restore later.
Part III: Periodicity
Periodicity

For each prefix of the input stream, compute its period

Motivation:

- Detecting anomalies in streams
- Preprocessing for pattern matching
Periodicity

**Exact periods**
Ergün et al., APPROX-RANDOM’10
- Periodic streams: $O(\log n)$ space, $O(\log n)$ time
- Non-periodic streams: $\Omega(m)$ space

**Approximate periods** (Hamming distance $\leq k$)
Ergün et al., APPROX-RANDOM’17
- Periods of length $< n/2$: $\tilde{O}(k^4)$ space
- All periods: $\Omega(n)$ space

**Approximate periods** ($\leq k$ wildcards)
Ergün et al., CSR’18
- Periods of length $< n/2$: $\tilde{O}(k^3)$ space
- All periods: $\Omega(n)$ space
Exact periods

We will show how to compute the period if it is $\leq n/4$

Lemma The period is equal to $\rho$ iff $T[1, n - \rho] = T[\rho, n]$

Lemma The only candidate for the period, $\rho$ is the first occurrence of $T[1, n/2]$ in $T$
Exact periods

We will show how to compute the period if it is $\leq n/4$

\[
\begin{align*}
T[1, n/2] & \quad T[1, n/2] \\
T[1, n/2] & \quad T[1, n/2] \\
\end{align*}
\]

Algorithm:

- Use the exact pattern matching algorithm to find the first occurrence $\rho$ of $T[1, n/2]$ (happens when $T[\rho + n/2 - 1]$ arrives)
- Memorize $\varphi(T[1, n - \rho])$ (we have $n - \rho \geq \rho + n/2$)
- If $\varphi(T[1, n - \rho]) = \varphi(T[\rho, n])$, then $\rho$ is the period w.h.p
Summary of today’s talk

Streaming algorithms:

- String similarity for Hamming and edit distances
- Exact pattern matching — $O(\log m)$ space, $O(1)$ time
- $k$-mismatch (Hamming distance) — $\tilde{O}(k)$ space, $\tilde{O}(\sqrt{k})$ time
- $k$-mismatch (edit distance) — $\tilde{O}(\sqrt{m} \ \text{polylog} \ k)$ space, $\tilde{O}(\sqrt{m} \ \text{polylog} \ k)$ time
- Periodicity

Thank you!