Streaming and property testing algorithms for string processing

Tatiana Starikovskaya

Based on joint work with:
R. Clifford, P. Gawrychowski, A. Fontaine, E. Porat, B. Sach
- Pattern matching has been studied for 40+ years
- More than 85 algorithms
- KMP algorithm uses $O(|P|)$ space and $O(|T|)$ time, and Aho-Corasick achieves similar bounds for dictionary matching
- We can’t do better: we must store a description of the pattern(s) and we must read the whole text
GAME OVER
Intrusion Detection Systems

- Large number of patterns
- Search patterns represent portions of known attack patterns and have length 1 – 30
- If only cache memory is used, the algorithm can benefit most from a high performance cache
Outline of today’s talk

Streaming model

- Exact pattern matching
- Approximate pattern matching (Hamming distance)
- Approximate pattern matching (edit distance)
- Preprocessing

Property testing model

- Exact pattern matching
Streaming model

We want to process the stream on-the-fly & in small space.
Part I: Exact pattern matching
Exact pattern matching

text $T$
\[
\begin{array}{ccccccc}
  & c & a & a & b & c & a \\
\end{array}
\]

pattern $P$
\[
\begin{array}{ccccccc}
  b & c & a & a & a & c \\
\end{array}
\]

- **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 $T$
Exact pattern matching

\[
\begin{array}{c}
\text{YES} \\
\text{Query} = \text{"Is there an occurrence of } P \text{"} \\
\text{Space} = \text{total space used by the stream processor} \\
\text{Time} = \text{time per position of } T
\end{array}
\]
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

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

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

It’s a good hash function

\( S_1, S_2 \) are two strings of length \( m \), the prime \( p \) is large

1. \( S_1 = S_2 \Rightarrow \varphi(S_1) = \varphi(S_2) \)

2. \( S_1 \neq S_2 \), lengths of \( S_1, S_2 \) are equal \( \Rightarrow \varphi(S_1) \neq \varphi(S_2) \) w.h.p.
Karp-Rabin algorithm

YES

When a new character \( t_i = a \) arrives:

1. Compute the fingerprint \( \varphi(t_{i-m+1} \ldots t_{i-1}t_i) \) in \( O(1) \) time

\[
\varphi(\text{caaacc}) = \left( (\varphi(\text{bcaaac}) - br^{m-1}) \cdot r + a \right) \mod p
\]

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 \( \Rightarrow \) we must store \( t_{i-m}, \ldots, t_{i-1} \)
Karp-Rabin algorithm

The K.-R. algorithm is a **streaming pattern matching algorithm** that uses $\Theta(m)$ space and $O(1)$ time per character of $T$. It finds all occurrences of $P$ in $T$ correctly w.h.p.
## 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, 2009</td>
<td>(O(\log m))</td>
<td>(O(\log m))</td>
</tr>
<tr>
<td>Breslauer &amp; Galil, 2011</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., 2015</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Golan &amp; Porat, 2017</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>Θ(m)</td>
<td>O(1)</td>
</tr>
<tr>
<td>Porat &amp; Porat, 2009 ★</td>
<td>O(log(m))</td>
<td>O(log(m))</td>
</tr>
<tr>
<td>Breslauer &amp; Galil, 2011</td>
<td>O(log(m))</td>
<td>O(1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Authors</th>
<th>Space ¹</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<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., 2015</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Golan &amp; Porat, 2017</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
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 $j+1$
text $T$

\[ t_i \]

occurrences of $p_1$

\[ \times \times \]

occurrences of $p_1p_2$

\[ \times \times \times \]

occurrences of $p_1p_2p_3p_4$

\[ \vdots \]

occurrences of $P = p_1p_2 \ldots p_m$

\[ \vdots \]

**for** each character $t_i$ **do**

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

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

    \[ lp \leftarrow \text{leftmost position in level } j \]

    **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
Porat & Porat, 2009 ★

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 $j + 1$
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 $j + 1$
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 $j + 1$
Porat & Porat, 2009 ★

text $T$ ----------------------------------- $t_i$

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 $j + 1$
text $T$ \overset{\longleftarrow}{\overset{\longleftarrow}{\overset{\longleftarrow}{\overset{\longleftarrow}{\overset{\longleftarrow}{\overset{\longleftarrow}{\overset{\longleftarrow}{t_i}}}}}}}}$

occurrences of $p_1$

occurrences of $p_1p_2$

occurrences of $p_1p_2p_3p_4$

occurrences of $P = p_1p_2 \ldots p_m$

**Lemma** If there are $\geq 3$ occurrences of a $2^j$-length string in a $2^{j+1}$-length string, the occurrences form a run

For each level we store:

- The leftmost and the second leftmost positions $l_p, l_p'$
- The fingerprints of $t_1t_2 \ldots t_{l_p}, t_{l_p+1} \ldots t_{l_p'},$ and $t_1 \ldots t_i$
Porat & Porat, 2009

For each level we need:

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

**Theorem** Porat & Porat algorithm is a streaming pattern matching algorithm that uses $O(\log m)$ space and $O(\log m)$ time per character
Part II: Approximate pattern matching
Approximate pattern matching

\[ \text{dist}(P,T) \]

Text \( T \):

```
| c | a | a | b | c | a | a | a | c | a |
```

Pattern \( P \):

```
| b | c | a | a | a | a |
```

- **Query** = “Distance between \( P \) and \( T \)”
- **Distance**: Hamming, edit, …
Approximate pattern matching (Hamming distance)

Any streaming algorithm for computing **exact** Hamming distances must use $\Omega(m)$ space

By **Yao’s minimax principle** it suffices to consider deterministic algorithms on “hard” distribution of the inputs

```
<table>
<thead>
<tr>
<th>text $T$</th>
<th>1 0 1 1 0 0</th>
<th>0 0 0 0 0 0 0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>pattern $P$</td>
<td>0 0 0 0 0 0</td>
<td>$T[1, m]$ is random</td>
</tr>
</tbody>
</table>
```

After reading $T[m]$, the algorithm cannot go back and read one of the letters $T[1], T[2], \ldots, T[m]$, but can restore $T[1, m]$

Therefore, it stores a full description of $T[1, m] \Rightarrow \Omega(m)$ space by information-theoretic ideas
Approximate pattern matching (Hamming distance)

Any streaming algorithm for computing **exact** Hamming distances must use $\Omega(m)$ space.

By **Yao’s minimax principle** it suffices to consider deterministic algorithms on “hard” distribution of the inputs.

$$\text{dist}(P, T) = 3$$

**text** $T$

\[
\begin{array}{cccccccc}
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

$T[1, m]$ is random

**pattern** $P$

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

After reading $T[m]$, the algorithm cannot go back and read one of the letters $T[1], T[2], \ldots, T[m]$, but can restore $T[1, m]$.

Therefore, it stores a full description of $T[1, m] \Rightarrow \Omega(m)$ space by information-theoretic ideas.
Approximate pattern matching (Hamming distance)

Any streaming algorithm for computing **exact** Hamming distances must use $\Omega(m)$ space

By **Yao’s minimax principle** it suffices to consider deterministic algorithms on “hard” distribution of the inputs

$$\text{dist}(P, T) = 2, \ T[1] = 3 - 2$$

The text $T$

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

is random

The pattern $P$

| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

After reading $T[m]$, the algorithm cannot go back and read one of the letters $T[1], T[2], \ldots, T[m]$, but can restore $T[1, m]$

Therefore, it stores a full description of $T[1, m] \Rightarrow \Omega(m)$ space by information-theoretic ideas
Approximate pattern matching (Hamming distance)

Any streaming algorithm for computing exact Hamming distances must use $\Omega(m)$ space.

By Yao’s minimax principle it suffices to consider deterministic algorithms on “hard” distribution of the inputs.

$$\text{dist}(P,T) = 2, \ T[2] = 2 - 2$$

After reading $T[m]$, the algorithm cannot go back and read one of the letters $T[1], T[2], \ldots, T[m]$, but can restore $T[1, m]$.

Therefore, it stores a full description of $T[1, m] \Rightarrow \Omega(m)$ space by information-theoretic ideas.
## 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 $\leq k$</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Porat &amp; Porat, 2009</td>
<td>$\tilde{O}(k^3)$</td>
<td>$\tilde{O}(k^2)$</td>
</tr>
<tr>
<td>Clifford, Fontaine, Porat,</td>
<td>$\tilde{O}(k^2)$</td>
<td>$\tilde{O}(\sqrt{k})$</td>
</tr>
<tr>
<td>Sach, S., 2016</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clifford, Kociumaka, Porat,</td>
<td>$O(k \log \frac{m}{k})$</td>
<td>$O(k \log^3 m \log \frac{m}{k})$</td>
</tr>
<tr>
<td>2018</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Single pattern, $(1+\varepsilon)$-approx.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clifford, S., 2016</td>
<td>$O(\varepsilon^{-5} \sqrt{m} \log^4 m)$</td>
<td>$O(\varepsilon^{-4} \log^3 m)$</td>
</tr>
</tbody>
</table>

$^2$In words
Approximate pattern matching (Hamming distance)

<table>
<thead>
<tr>
<th>Authors</th>
<th>Space</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Porat &amp; Porat, 2009 ★</td>
<td>$\tilde{O}(k^3)$</td>
<td>$\tilde{O}(k^2)$</td>
</tr>
<tr>
<td>Clifford, Fontaine, Porat, Sach, S., 2016</td>
<td>$\tilde{O}(k^2)$</td>
<td>$\tilde{O}(\sqrt{k})$</td>
</tr>
<tr>
<td>Clifford, Kociumaka, Porat, 2018</td>
<td>$O(k \log \frac{m}{k})$</td>
<td>$O(k \log^3 m \log \frac{m}{k})$</td>
</tr>
</tbody>
</table>

Single pattern, (1 + $\varepsilon$)-approx.

<table>
<thead>
<tr>
<th>Authors</th>
<th>Space</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clifford, S., 2016</td>
<td>$O(\varepsilon^{-5} \sqrt{m} \log^4 m)$</td>
<td>$O(\varepsilon^{-4} \log^3 m)$</td>
</tr>
</tbody>
</table>

$^2$In words
$dist(P,T)$

text $T$

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

pattern $P$

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

- If $\text{HAM}(P,T) > k$, output “NO”
- Otherwise, output $\text{HAM}(P,T)$
From 1 mismatch to exact pattern matching

\[ \text{string}_1 \]

\[ \text{string}_2 \]

- Is HAM (\text{string}_1, \text{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 $\text{string}_1$ into $q$ equi-spaced substrings
- Partition $\text{string}_2$ into $q$ equi-spaced substrings

In total: $O(\log m)$ primes, and for each prime there are $O(\log^2 m)$ pairs of substrings
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

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

text $T$

pattern $P$

**Complexity**

**Space** = $O(\log m \cdot \log^2 m \cdot \log^2 m \cdot \log m)$

# of primes  # of substr.  # of subpatterns

**Time** = $O(\log m \cdot \log^2 m \cdot \log^2 m)$

# of primes  # of substr.  # of subpatterns
Approximate pattern matching (Hamming distance)

Porat & Porat, 2009
\(\tilde{O}(k^3)\) space, \(\tilde{O}(k^2)\) time
Same as for \(k = 1\) but take more primes

Clifford, Fontaine, Porat, Sach, S., 2016
\(\tilde{O}(k^2)\) space, \(\tilde{O}(\sqrt{k})\) time
We can take fewer primes if we choose them at random + periodicity to improve time

Clifford, Kociumaka, Porat, 2018
\(O(k \log \frac{m}{k})\) space, \(O(k \log^3 m \log \frac{m}{k})\) time
New encoding for mismatch information + periodicity + exponentially growing prefixes
Approximate pattern matching (edit distance)

$ED(P,T)$

**text $T$**

c a a b c a a a a c a

**pattern $P$**

b c a a a a c

$ED(P,S)$ = minimum number of insertions, deletions, and replacements that transform $P$ into $S$

Example: $P = aaac$, $S = abacab$, edit distance = 2

- If $ED(P,T) > k$, output “NO”
- Otherwise, output $ED(P,T)$
Approximate pattern matching (edit distance)

\[ ED(P, T) \]

Text \( T \):

```
| c a a b c a a a c a |
```

Pattern \( P \):

```
| b c a a a a c |
```

\[ ED(P, S) = \text{minimum number of insertions, deletions, and replacements that transform } P \text{ into } S \]

Example: \( P = \text{aaac}, S = \text{abacab}, \text{edit distance} = 2 \)

- Hybrid dynamic programming: \( \mathcal{O}(m) \) space, \( \mathcal{O}(k) \) time
- S., 2017: \( \mathcal{O}(\sqrt{m} \cdot \text{poly}(k, \log m)) \) space, \( \mathcal{O}(\sqrt{m} \cdot \text{poly}(k, \log m)) \) time
Embedding from edit to Hamming distance

Chakraborty, Goldenberg, Koucky, 2016

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

<table>
<thead>
<tr>
<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>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 $S'$:

$S : 0 \ 1 \ 0 \ ... \ 0$

$S' : ... \ text \ position = 1, j = 1$

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

Chakraborty, Goldenberg, Koucky, 2016

Pick $3n$ 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 $S'$:

$S$: 0 1 0 ... 0
$S'$: 0 ... 0

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

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

Chakraborty, Goldenberg, Koucky, 2016

Pick $3n$ random functions $h_j : \{0, 1\} \to \{0, 1\}$

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

Copy letters of $S$ to $S'$:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>\ldots</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$ :</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>\ldots</td>
<td>0</td>
</tr>
<tr>
<td>$S'$ :</td>
<td>0</td>
<td>\ldots</td>
<td>0</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

Chakraborty, Goldenberg, Koucky, 2016

Pick $3n$ random functions $h_j : \{0, 1\} \to \{0, 1\}$

<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>
</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>

Copy letters of $S$ to $S'$:

<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>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$S'$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</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

Chakraborty, Goldenberg, Koucky, 2016

Pick $3n$ 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>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Copy letters of $S$ to $S'$:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th></th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$: 0</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$S'$: 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 = 1, $j = 2$**
Embedding from edit to Hamming distance

Chakraborty, Goldenberg, Koucky, 2016

Pick $3n$ 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 $S'$:

$S : \begin{array}{cccccccc} 1 & 2 & 3 & \ldots & n \\ 0 & 1 & 0 & \ldots & 0 \end{array}$
$S' : \begin{array}{cccccccc} 1 & 2 & 3 & \ldots & n \\ 0 & 0 & \ldots & 0 \end{array}$

text position = 1, $j = 2$

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

Chakraborty, Goldenberg, Koucky, 2016

Pick $3n$ random functions $h_j : \{0, 1\} \to \{0, 1\}$

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

Copy letters of $S$ to $S'$:

<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>0</td>
</tr>
<tr>
<td>$S'$:</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>0</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

Chakraborty, Goldenberg, Koucky, 2016

Pick $3n$ 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>
</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 $S'$:

<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>$S'$ :</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

Chakraborty, Goldenberg, Koucky, 2016

Pick $3n$ 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>1</td>
</tr>
</tbody>
</table>

Copy letters of $S$ to $S'$:

<table>
<thead>
<tr>
<th></th>
<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>...</td>
<td>0</td>
</tr>
<tr>
<td>$S'$ : 0</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>0</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

Chakraborty, Goldenberg, Koucky, 2016

Pick $3n$ random functions $h_j : \{0, 1\} \to \{0, 1\}$

<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 $S'$:

<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>0</td>
</tr>
<tr>
<td>$S'$:</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>...</td>
</tr>
</tbody>
</table>

If $ED(S, T) = k$, then $k/2 \leq HD(S', T') \leq O(k^2)$ w/ prob. 0.99
Embedding from edit to Hamming distance

Chakraborty, Goldenberg, Koucky, 2016

Pick $3n$ 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>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Copy letters of $S$ to $S'$:

<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>$S'$ :</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

text position = 2, $j = 3$

Belazzougui, Zhang, 2016

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

\[ B \approx \sqrt{m} \]

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: Preprocessing
Preprocessing for pattern matching

Can we preprocess the patterns in a streaming way? If yes, do we need to read them several times? How much space do we need?

Periodicity — Ergün, Jowhari, Saglam, 2010
- Periodic patterns: $O(\log m)$ space, $O(\log m)$ time
- Non-periodic patterns: $\Omega(m)$ space
- 2 passes (periodic and non-periodic patterns): $O(\log m)$ space, $O(\log m)$ time

Periodicity with mismatches — Ergün et al., 2017
- Periodic patterns: $O(k^4 \log^9 n)$ space
- 2-pass algorithm for non-periodic patterns, lower bounds
Part IV: Property testing model
Pattern matching

Is $T$ free from occurrences of $P$?

Same question when $T$ and $P$ are of dimension $d \geq 2$
Property testing model

If Sherlock wants to solve the problem fast, he can only query a few characters of $T$
Property testing model

**Task:** develop an ultra-efficient randomised algorithm to decide whether $T$ is free from occurrences of $P$

**We must**

- accept, if $T$ is $\varepsilon_1$-close to being $P$-free
- reject, if $T$ is $\varepsilon_2$-far from being $P$-free
- accept or reject otherwise

$\varepsilon_1$-close = we can fix $\leq \varepsilon_1 n$ characters of $T$ so that the property is satisfied

$\varepsilon_2$-far = we must fix $\geq \varepsilon_2 n$ characters of $T$ so that the property is satisfied
Property testing model

Task: develop an ultra-efficient randomised algorithm to decide whether $T$ is free from occurrences of $P$

We must

- accept, if $T$ is $\varepsilon_1$-close to being $P$-free
- reject, if $T$ is $\varepsilon_2$-far from being $P$-free
- accept or reject otherwise

Ben-Eliezer, Korman, Reichman, 2017

There is an algorithm which queries $O(\varepsilon^{-1})$ letters of $T$ and distinguishes between $\varepsilon/2$-close and $\varepsilon$-far (for almost all patterns)
Summary of today’s talk

It’s all about **pattern matching**

Randomisation and approximation $\Rightarrow$ more efficient algorithms

Many open questions

Thank you!