Pattern Matching in Multiple Streams

CPM, 3–5 July 2012

Markus Jalsenius

Joint work with
Raphaël Clifford, Benjamin Sach and Ely Porat
Output **Match** or **No Match** before next symbol arrives.

We consider different notions of a match.
Problem

Stream

Pattern
Problem

Stream

A new symbol arrives in any one of the streams.
Output **Match** or **No Match** before the next symbol arrives (in any of the streams).

Pattern

<table>
<thead>
<tr>
<th>bdcbabbdacdcddcddcaad</th>
</tr>
</thead>
<tbody>
<tr>
<td>bdddcdbdcdacbabbbabbbbabcccbdbcbabdashcdcdcdcdcdccdaaa</td>
</tr>
<tr>
<td>ddddbdacdbababcbbbdcbbaaadccccdca</td>
</tr>
<tr>
<td>abddddyccdbdacabababcbbdcbdbabdashcdcdcdccdaa</td>
</tr>
<tr>
<td>cdddcdddbcdbdbdbabdashcdcdcdccdaa</td>
</tr>
<tr>
<td>abccbbdbdbdbdbabdashcdcdcdccdaa</td>
</tr>
<tr>
<td>ccdbdacababbbacaabcbdbbbdbdbbashcdcdcdccdaa</td>
</tr>
</tbody>
</table>
Problem

Stream

Pattern

bdcbabdadcddccdaad

bdddccbdacababbbacbbabcccbbdcbabdcabdcddccdaaccddcdaa

bacbbabcbddbdacdcddcddccddcbdacaabacccbbdbcbbaaacccddca

abccccbbddccbdcbabdacdcddccddccbdacabacccbbdbcbbaaacccddca

abcccbddccbdbbcbdbabdacdcddccddccbdacabacccbbdbcbbaaacccddca

abdacdcddcddcaaccddcbdcdbacbbdcbcababbaaabaabccbc

ccbdacababbaacaaabcbbdddcbbdbcbdbabdcacdcddcaaccddcccd
<table>
<thead>
<tr>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>bdddbcccdbdabdbababbbbabcccbbdbcbabbacdbcdcdcdcdcdccdaad</td>
</tr>
<tr>
<td>bdddbcccdbdabdbababbbbabcccbbdbcbabbacdbcdcdcdcdcdccdaad</td>
</tr>
<tr>
<td>bdddbcccdbdabdbababbbbabcccbbdbcbabbacdbcdcdcdcdcdccdaad</td>
</tr>
<tr>
<td>bdddbcccdbdabdbababbbbabcccbbdbcbabbacdbcdcdcdcdcdccdaad</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>bdddbcccdbdabdbababbbbabcccbbdbcbabbacdbcdcdcdcdcdccdaad</td>
</tr>
<tr>
<td>bdddbcccdbdabdbababbbbabcccbbdbcbabbacdbcdcdcdcdcdccdaad</td>
</tr>
<tr>
<td>bdddbcccdbdabdbababbbbabcccbbdbcbabbacdbcdcdcdcdcdccdaad</td>
</tr>
<tr>
<td>bdddbcccdbdabdbababbbbabcccbbdbcbabbacdbcdcdcdcdcdccdaad</td>
</tr>
</tbody>
</table>

Diagram:
- Stream of data blocks
- Matching pattern: `bdcdbabdbdadcddcdccdaad`
Problem

Read-only memory

Stream

Approach

Preprocess pattern, store the output in **read-only memory** that is shared across the streams.

Eqip each stream with **small working memory**.
Problem

Read-only memory

Pattern

Stream

Approach

Preprocess pattern, store the output in read-only memory that is shared across the streams.

Eqip each stream with small working memory.

Want fast outputs!
# Results

<table>
<thead>
<tr>
<th></th>
<th>Space</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 stream</td>
<td>1 stream (offline)</td>
</tr>
<tr>
<td></td>
<td>$s$ streams</td>
<td>$s$ streams (offline)</td>
</tr>
<tr>
<td>Exact matching</td>
<td>$O(m)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td></td>
<td>$O(m + s)$ words</td>
<td>$O(1)$</td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td>$k$-mismatch</td>
<td>$O(m)$</td>
<td>$O(n\sqrt{k \log k})$</td>
</tr>
<tr>
<td></td>
<td>$O(m + ks)$ words</td>
<td>$O(k)$</td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td>$k$-difference</td>
<td>$O(m)$</td>
<td>$O(nk)$</td>
</tr>
<tr>
<td>(edit distance)</td>
<td>$O(m)$</td>
<td>$O(k)$</td>
</tr>
<tr>
<td></td>
<td>$O(m + ks)$ words</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
</tbody>
</table>

**Notation:**

$m =$ pattern length, $n =$ text length (when offline), $\Sigma =$ alphabet. We operate in the RAM model.
## Results

<table>
<thead>
<tr>
<th></th>
<th>Space</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 stream</td>
<td>s streams</td>
</tr>
<tr>
<td>Exact matching</td>
<td>$O(m)$ words</td>
<td>$O(k)$ words offline</td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td>$k$-mismatch</td>
<td>$O(m)$ words</td>
<td>$O(n \sqrt{k \log k})$ offline</td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td>$k$-difference</td>
<td>$O(m)$ words</td>
<td>$O(nk)$ offline</td>
</tr>
<tr>
<td>(edit distance)</td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
</tbody>
</table>

**Notation:**

$m = \text{pattern length}$, $n = \text{text length (when offline)}$, $\Sigma = \text{alphabet}$. We operate in the RAM model.
## Results

<table>
<thead>
<tr>
<th></th>
<th>Space</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 stream</td>
<td>s streams</td>
</tr>
<tr>
<td><strong>Exact matching</strong></td>
<td>$O(m)$</td>
<td>$O(m + s)$ words</td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td><strong>$k$-mismatch</strong></td>
<td>$O(m)$</td>
<td>$O(m + ks)$ words</td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td><strong>$k$-difference</strong> (edit distance)</td>
<td>$O(m)$</td>
<td>$O(m + ks)$ words</td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
</tbody>
</table>

**Notation:**

$m$ = pattern length, $n$ = text length (when offline), $\Sigma$ = alphabet. We operate in the RAM model.
### Results

<table>
<thead>
<tr>
<th></th>
<th>Space</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 stream</td>
<td>$s$ streams</td>
</tr>
<tr>
<td><strong>Exact matching</strong></td>
<td>$O(m)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td></td>
<td>$O(m + s)$ words</td>
<td>$O(1)$</td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td><strong>$k$-mismatch</strong></td>
<td>$O(m)$</td>
<td>$O(n \sqrt{k \log k})$</td>
</tr>
<tr>
<td></td>
<td>$O(m + ks)$ words</td>
<td>$O(k)$</td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td><strong>$k$-difference (edit distance)</strong></td>
<td>$O(m)$</td>
<td>$O(nk)$ offline</td>
</tr>
<tr>
<td></td>
<td>$O(m + ks)$ words</td>
<td>$O(k)$</td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
</tbody>
</table>

**Naive solution:** $O(ms)$ space

**Notation:**
- $m$ = pattern length
- $n$ = text length (when offline)
- $\Sigma$ = alphabet
- We operate in the RAM model.
# Results

<table>
<thead>
<tr>
<th></th>
<th>Space</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 stream</td>
<td>1 stream</td>
</tr>
<tr>
<td></td>
<td><em>s</em> streams</td>
<td><em>s</em> streams</td>
</tr>
<tr>
<td>Exact matching</td>
<td>$O(m)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td></td>
<td>$O(m + s)$ words</td>
<td>$O(k)$ offline</td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td>$k$-mismatch</td>
<td>$O(m)$</td>
<td>$O(k)$ offline</td>
</tr>
<tr>
<td></td>
<td>$O(m + ks)$ words</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td>$k$-difference</td>
<td>$O(m)$</td>
<td>$O(k)$ offline</td>
</tr>
<tr>
<td>(edit distance)</td>
<td>$O(m + ks)$ words</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td></td>
<td>Naive solution: $O(ms)$ space</td>
<td></td>
</tr>
</tbody>
</table>

**Notation:**

$m = \text{pattern length}, \ n = \text{text length (when offline)}, \ \Sigma = \text{alphabet. We operate in the RAM model.}$

**Read-only space:** Preprocessing time is roughly $O(m \log m)$

**Read/write space:**
## Results

<table>
<thead>
<tr>
<th></th>
<th>Space</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 stream</td>
<td>s streams</td>
</tr>
<tr>
<td><strong>Exact matching</strong></td>
<td>$O(m)$</td>
<td>$O(m + s)$ words</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Omega(m \log</td>
</tr>
<tr>
<td><strong>k-mismatch</strong></td>
<td>$O(m)$</td>
<td>$O(m + ks)$ words</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Omega(m \log</td>
</tr>
<tr>
<td><strong>k-difference</strong> (edit distance)</td>
<td>$O(m)$</td>
<td>$O(m + ks)$ words</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Omega(m \log</td>
</tr>
</tbody>
</table>

**Naive solution:** $O(ms)$ space

**Notation:**

$m =$ pattern length, $n =$ text length (when offline), $\Sigma =$ alphabet. We operate in the RAM model.
<table>
<thead>
<tr>
<th></th>
<th>1 stream</th>
<th>(s) streams</th>
<th>1 stream</th>
<th>(s) streams</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exact matching</strong></td>
<td>(O(m))</td>
<td>(O(m+s)) words&lt;br&gt;(\Omega(m \log</td>
<td>\Sigma</td>
<td>+ s)) bits</td>
</tr>
<tr>
<td><strong>(k)-mismatch</strong></td>
<td>(O(m))</td>
<td>(O(m+ks)) words&lt;br&gt;(\Omega(m \log</td>
<td>\Sigma</td>
<td>+ ks)) bits</td>
</tr>
<tr>
<td><strong>(k)-difference</strong> (edit distance)</td>
<td>(O(m))</td>
<td>(O(m+ks)) words&lt;br&gt;(\Omega(m \log</td>
<td>\Sigma</td>
<td>+ ks)) bits</td>
</tr>
</tbody>
</table>

**Notation:**

\(m\) = pattern length, \(n\) = text length (when offline), \(\Sigma\) = alphabet. We operate in the RAM model.
## Results

### Space

<table>
<thead>
<tr>
<th></th>
<th>1 stream</th>
<th>$s$ streams</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exact matching</strong></td>
<td>$O(m)$</td>
<td>$O(m + s)$ words</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Omega(m \log</td>
</tr>
<tr>
<td><strong>$k$-mismatch</strong></td>
<td>$O(m)$</td>
<td>$O(m + ks)$ words</td>
</tr>
<tr>
<td><strong>$k$-difference</strong> (edit distance)</td>
<td>$O(m)$</td>
<td>$O(m + ks)$ words</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Omega(m \log</td>
</tr>
</tbody>
</table>

### Time

<table>
<thead>
<tr>
<th></th>
<th>1 stream</th>
<th>$s$ streams</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exact matching</strong></td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td><strong>$k$-mismatch</strong></td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td><strong>$k$-difference</strong> (edit distance)</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>

---


Also, for one stream: $O(\sqrt{k \log k + \log m})$


**Notation:**

$m =$ pattern length, $n =$ text length (when offline), $\Sigma =$ alphabet. We operate in the RAM model.
## Results

<table>
<thead>
<tr>
<th></th>
<th>Space</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 stream</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>s streams</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Exact matching</strong></td>
<td>(O(m))</td>
<td>(O(1))</td>
</tr>
<tr>
<td></td>
<td>(O(m + s)) words</td>
<td>(O(1))</td>
</tr>
<tr>
<td></td>
<td>(\Omega(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td><strong>k-mismatch</strong></td>
<td>(O(m))</td>
<td>(O(n \sqrt{k \log k})) offline</td>
</tr>
<tr>
<td></td>
<td>(O(m + ks)) words</td>
<td>(O(k))</td>
</tr>
<tr>
<td></td>
<td>(\Omega(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td><strong>k-difference</strong></td>
<td>(O(m))</td>
<td>(O(nk)) offline</td>
</tr>
<tr>
<td>(edit distance)</td>
<td>(O(m + ks)) words</td>
<td>(O(k))</td>
</tr>
<tr>
<td></td>
<td>(\Omega(m \log</td>
<td>\Sigma</td>
</tr>
</tbody>
</table>


**Notation:**

\(m\) = pattern length, \(n\) = text length (when offline), \(\Sigma\) = alphabet. We operate in the RAM model.
## Results

<table>
<thead>
<tr>
<th></th>
<th>Space</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 stream</td>
<td>s streams</td>
</tr>
<tr>
<td><strong>Exact matching</strong></td>
<td>$O(m)$</td>
<td>$O(m+s)$ words</td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td><strong>$k$-mismatch</strong></td>
<td>$O(m)$</td>
<td>$O(m+ks)$ words</td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td><strong>$k$-difference (edit distance)</strong></td>
<td>$O(m)$</td>
<td>$O(m+ks)$ words</td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td><strong>$L_1$, $L_2$, Hamming distances, convolution/cross-correlation</strong></td>
<td></td>
<td>$\Omega(ms)$ bits</td>
</tr>
</tbody>
</table>

**Notation:**

$m =$ pattern length, $n =$ text length (when offline), $\Sigma =$ alphabet. We operate in the RAM model.
## Results

<table>
<thead>
<tr>
<th></th>
<th>Space</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 stream</td>
<td>1 stream</td>
</tr>
<tr>
<td></td>
<td>1 stream</td>
<td>1 stream</td>
</tr>
<tr>
<td>Exact matching</td>
<td>$O(m)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td></td>
<td>$O(m + s)$ words</td>
<td>$O(1)$</td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td>$k$-mismatch</td>
<td>$O(m)$</td>
<td>$O(n\sqrt{k \log k})$ offline</td>
</tr>
<tr>
<td></td>
<td>$O(m + ks)$ words</td>
<td>$O(k)$ offline</td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td>$k$-difference</td>
<td>$O(m)$</td>
<td>$O(nk)$ offline</td>
</tr>
<tr>
<td>(edit distance)</td>
<td>$O(m + ks)$ words</td>
<td>$O(k)$ offline</td>
</tr>
<tr>
<td></td>
<td>$\Omega(m \log</td>
<td>\Sigma</td>
</tr>
<tr>
<td>$L_1$, $L_2$, Hamming distances, convolution/cross-correlation</td>
<td>$\Omega(ms)$ bits</td>
<td></td>
</tr>
</tbody>
</table>

Randmosied bounds are open!

### Notation:

$m =$ pattern length, $n =$ text length (when offline), $\Sigma =$ alphabet. We operate in the RAM model.
Exact matching

$O(1)$ amortised (e.g. KMP)
$O(1)$ unamortised (e.g. Galil 1981)
Exact matching

$O(1)$ amortised (e.g. KMP)
$O(1)$ unamortised (e.g. Galil 1981)

⚠️ Buffering the text $\implies O(ms)$ space
Exact matching

Simple modification of KMP

Pattern

```
 a a b a a c a a b a a c a a d a a a c a a a b a a
```
Exact matching

Simple modification of KMP

Pattern

Prefix table

\[ O(m + s) \]

\[ O(1) \]
Exact matching

Simple modification of KMP

Pattern

Prefix table

Stream
Exact matching

Simple modification of KMP

Pattern

Prefix table

Stream
Simple modification of KMP
Exact matching

Simple modification of KMP

Prefix table

Stream

Pattern
Exact matching

Simple modification of KMP

Pattern

Prefix table

Stream

Shift pattern 9 steps

\[ O(m+s) \quad \text{and} \quad O(1) \]
Simple modification of KMP

Prefix table

Pattern

Exact matching

$O(m + s)$  $O(1)$
Exact matching

Simple modification of KMP

For each position, the shift is found in \( O(1) \) time through static perfect hashing.
Simple modification of KMP

Total number of elements to store is at most $m$.

Exact matching

Simple modification of KMP

Storing the hash tables: $O(m)$ space.
Each stream has a pointer into the pattern: $O(1)$ space per stream.
Time per symbol: $O(1)$.

Total number of elements to store is at most $m$.

Preparation for $k$-mismatch/difference

Encoding the stream in terms of the pattern.

Pattern

<table>
<thead>
<tr>
<th>a</th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>a</th>
<th>c</th>
<th>a</th>
<th>a</th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>a</th>
<th>c</th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Stream

b a a c c a a a b a a c a b a a
Encoding the stream in terms of the pattern.

Pattern: 

\[
\begin{array}{ccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10
\end{array}
\]

Stream: 

\[
\begin{array}{ccccccccccc}
\text{b} & \text{a} & \text{a} & \text{c} & \text{c} & \text{a} & \text{a} & \text{a} & \text{b} & \text{a} & \text{a} & \text{c} & \text{a} & \text{b} & \text{a} & \text{a}
\end{array}
\]

(2, 5)
Preparation for $k$-mismatch/difference

Encoding the stream in terms of the pattern.

Pattern

<table>
<thead>
<tr>
<th>a</th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>a</th>
<th>a</th>
<th>c</th>
<th>a</th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>
Preparation for $k$-mismatch/difference

Encoding the stream in terms of the pattern.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>a</th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>a</th>
<th>c</th>
<th>a</th>
<th>a</th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(2, 5) \hspace{1cm} (5, 7) \hspace{1cm} (1, 6)
Preparation for \( k \)-mismatch/difference

Encoding the stream in terms of the pattern.

Encoding is not necessarily unique.
Preparation for $k$-mismatch/difference

Encoding the stream in terms of the pattern.

Pattern: $a a b a a b a c a a b a a c$

Encoding is not necessarily unique.

**Greedy construction**

Extend pair if possible...
Preparation for \( k \)-mismatch/difference

Encoding the stream in terms of the pattern.

<table>
<thead>
<tr>
<th>Pattern</th>
<th>a</th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>a</th>
<th>c</th>
<th>a</th>
<th>c</th>
<th>a</th>
<th>a</th>
<th>b</th>
<th>a</th>
<th>a</th>
<th>c</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoding</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Greedy construction**

Extend pair if possible...
Preparation for $k$-mismatch/difference

Encoding the stream in terms of the pattern.

Encoding is not necessarily unique.

**Greedy construction**

Extend pair if possible...
Preparation for $k$-mismatch/difference

Encoding the stream in terms of the pattern.

Pattern: $\text{a a b a a a c a a a b a c}$

Encoding is not necessarily unique.

**Greedy construction**

Extend pair if possible... ...if not, start a new pair.
Preparation for \( k \)-mismatch/difference

Encoding the stream in terms of the pattern.

![Pattern and Encoding Diagram]

**Greedy construction**

Extend pair if possible... ...if not, start a new pair.

Encoding is not necessarily unique.
Preparation for $k$-mismatch/difference

Encoding the stream in terms of the pattern.

<table>
<thead>
<tr>
<th>Pattern</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>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>b</td>
</tr>
</tbody>
</table>

(2, 5) (5, 7) (1, 6) (2, 7) (0, 1)

b a a c c a a a b a a c a b a a c a a a a a

Encoding is not necessarily unique.

**Greedy construction**

Extend pair if possible... ...if not, start a new pair.
Preparation for $k$-mismatch/difference

Encoding the stream in terms of the pattern.

Pattern: $a a b a a c a a a b a a c a b a a c a a a a a$

Encoding is not necessarily unique.

**Greedy construction**

Extend pair if possible... ...if not, start a new pair.

Results in a minimal length encoding.

Takes $O(1)$ time per symbol (using suffix tree of pattern).
Preparation for $k$-mismatch/difference

Encoding the stream in terms of the pattern.

Any stream/pattern LCE query can be answered through at most three self-LCE queries on the pattern.
Preparation for $k$-mismatch/difference

Encoding the stream in terms of the pattern.

Any stream/pattern LCE query can be answered through at most three self-LCE queries on the pattern.
Preparation for $k$-mismatch/difference

Encoding the stream in terms of the pattern.

Any stream/pattern LCE query can be answered through at most three self-LCE queries on the pattern.
Preparation for $k$-mismatch/difference

Encoding the stream in terms of the pattern.

Any stream/pattern LCE query can be answered through at most three self-LCE queries on the pattern.
Preparation for \( k \)-mismatch/difference

Encoding the stream in terms of the pattern.

Any stream/pattern \textbf{LCE} query can be answered through at most \textbf{three} self-LCE queries on the pattern.
Preparation for $k$-mismatch/difference

Encoding the stream in terms of the pattern.

Any stream/pattern LCE query can be answered through at most three self-LCE queries on the pattern.
Preparation for \( k \)-mismatch/difference

Encoding the stream in terms of the pattern.

Any stream/pattern \textbf{LCE} query can be answered through at most \textbf{three} self-LCE queries on the pattern.
Preparation for $k$-mismatch/difference

Encoding the stream in terms of the pattern.

Any stream/pattern LCE query can be answered through at most three self-LCE queries on the pattern.
Preparation for $k$-mismatch/difference

Encoding the stream in terms of the pattern.

Pattern

<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>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td>a</td>
</tr>
</tbody>
</table>

Any stream/pattern **LCE** query can be answered through at most **three** self-LCE queries on the pattern.

Preprocess pattern to support LCE queries in constant time.
$k$-mismatch

\[ O(m+ks) \quad O(k) \]

Pattern: c c c a a b a a a a b a a c b a a a a
$k$-mismatch

LCE query
(‘kangaroo jumping’)

Pattern

\[
\begin{array}{cccccccccccccccccccc}
\text{Pattern} & c & c & c & c & a & a & b & a & a & a & a & a & b & a & a & a & c & b & a & a & a \\
\end{array}
\]
$k$-mismatch

LCE query
(’kangaroo jumping’)

Pattern

\[
\begin{array}{ccccccccccccccccccc}
\text{b} & \text{a} & \text{a} & \text{c} & \text{c} & \text{a} & \text{a} & \text{a} & \text{b} & \text{a} & \text{a} & \text{c} & \text{a} & \text{b} & \text{a} & \text{a} & \text{c} & \text{a} & \text{a} & \text{a} \\
\text{Pattern} & \text{c} & \text{c} & \text{c} & \text{a} & \text{a} & \text{b} & \text{a} & \text{a} & \text{a} & \text{a} & \text{b} & \text{a} & \text{a} & \text{c} & \text{b} & \text{a} & \text{a} & \text{a} \\
\end{array}
\]
**$k$-mismatch**

LCE query

(’kangaroo jumping’)

Pattern

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

```
Pattern
```

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

$O(m+ks)$  $O(k)$
**$k$-mismatch**

LCE query
(’kangaroo jumping’)

- Each jump spans at most three pairs in stream encoding.
- A mismatch could be in its own pair.

$\implies$ Only store the last $4(k + 1)$ pairs of each stream.

$O(m + ks)$ space ($m$ for pattern/pattern LCE queries), $O(k)$ time per symbol.
$k$-difference

Edit operations: **insert, delete, mismatch**.
Report smallest edit distance between pattern and suffixes of stream if $k$ or less.

Stream $T$

Pattern $P$

![Stream and Pattern Diagram](image-url)
**k-difference**

Edit operations: **insert, delete, mismatch**.
Report smallest edit distance between pattern and suffixes of stream if $k$ or less.

**Dynamic programming**

$D[j, i] =$ the minimum of all $k$-bounded edit distances between pattern prefix $P[0 \ldots j]$ and all suffixes of $T[0 \ldots i]$. Thus, we want $D[m-1, i]$ as symbol $i$ arrives.
**$k$-difference**

Edit operations: **insert, delete, mismatch**.

Report smallest edit distance between pattern and suffixes of stream if $k$ or less.

Dynamic programming

$$D[j, i] = \min \begin{cases} 
D[j, i-1] + 1 & \text{(insert)} \\
D[j-1, i] + 1 & \text{(delete)} \\
D[j-1, i-1] + 1 - \text{eq}(i, j) & \text{(mismatch)} \\
k + 1 & \text{($k$-bounded)} 
\end{cases}$$
**k-difference**

Edit operations: **insert, delete, mismatch**.
Report smallest edit distance between pattern and suffixes of stream if \( k \) or less.

**Dynamic programming**

\[
D[j, i] = \min \begin{cases} 
D[j, i - 1] + 1 & \text{(insert)} \\
D[j - 1, i] + 1 & \text{(delete)} \\
D[j - 1, i - 1] + 1 - \text{eq}(i, j) & \text{(mismatch)} \\
k + 1 & \text{(k-bounded)}
\end{cases}
\]

**Question**

How do we compute this fast for a stream with small working memory?
$k$-difference

Dynamic programming table

Pattern

0

$\ldots$

$m - 1$

Stream

$\ldots$

$m$

$\ldots$

$k$

$k$
Dynamic programming table

Pattern

Stream

Compute $D[m - 1, i]$ for $i$ in this interval.
Start work here.
**$k$-difference**

Dynamic programming table

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Stream</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0$</td>
<td>$m-1$</td>
</tr>
</tbody>
</table>

- **Compute the $i$-values using offline method of Landau-Vishkin 1988.**
  - **Ingredient:** LCE
  - **Table space:** $O(k)$
  - **Time:** $O(k^2)$
    - ($O(k)$ per symbol)

- **Compute $D[m-1, i]$ for $i$ in this interval.**
  - Start work here.
**k-difference**

Dynamic programming table

Compute the $\text{yellow}$-values using offline method of Landau-Vishkin 1988.

Ingredient: LCE

Table space: $O(k)$

Time: $O(k^2)$ ($O(k)$ per symbol)

These values can be set to some constant

Compute the $\text{orange}$-values directly using the recurrence.

Space: $O(k)$

Time: $O(k^2)$ $O(k)$ per symbol

These values can be set to some constant

Compute $D[m-1, i]$ for $i$ in this interval.

Start work here.

Pattern

Stream

$m$

$m-1$

Start work here.
Dynamic programming table

Pattern

Compute the \( k \)-values using offline method of Landau-Vishkin 1988.

Ingredient: LCE

Table space: \( O(k) \)

Time: \( O(k^2) \) (\( O(k) \) per symbol)

Compute the \( k \)-values directly using the recurrence.

Space: \( O(k) \)

Time: \( O(k^2) \) \( O(k) \) per symbol

Use the recurrence.

Space: \( O(k) \)

Time: \( O(k) \) per symbol

Compute \( D[m-1, i] \) for \( i \) in this interval.

Start work here.

These values can be set to some constant
Dynamic programming table

Compute the \(k\)-values using offline method of Landau-Vishkin 1988.

Ingredient: LCE

Table space: \(O(k)\)

Time: \(O(k^2)\)

\((O(k)\) per symbol)
**Dynamic programming table**

**Pattern**

<table>
<thead>
<tr>
<th>0</th>
<th>Compute the (k)-values using offline method of Landau-Vishkin 1988.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(m - 1)</td>
<td>Compute the (k)-values directly using the recurrence.</td>
</tr>
</tbody>
</table>

**Ingredient:** LCE

**Table space:** \(O(k)\)

**Time:** \(O(k^2)\) \((O(k)\) per symbol)  

These values can be set to some constant.

**Stream**

- **Space:** \(O(k)\)  
  - **Time:** \(O(k^2)\) \((O(k)\) per symbol)

Use the recurrence.

- **Space:** \(O(k)\)  
  - **Time:** \(O(k)\) \(\text{per symbol}\)

Compute \(D[m-1, i]\) for \(i\) in this interval. Start work here.

**k-difference**

\[
O(m+ks) \quad O(k)
\]
**k-difference**

**Dynamic programming table**

Compute the \(-\) values using offline method of Landau-Vishkin 1988.

**Ingredient:** LCE

**Table space:** \(O(k)\)

**Time:** \(O(k^2)\) \((O(k) \text{ per symbol})\)

These values can be set to some constant

Compute the \(D\)-values directly using the recurrence.

**Space:** \(O(k)\)

**Time:** \(O(k^2)\) \(O(k)\) \(O(k)\) per symbol

Use the recurrence.

**Space:** \(O(k)\)

**Time:** \(O(k)\) per symbol

Compute \(D[m-1, i]\) for \(i\) in this interval.

Start work here.
**k-difference**

### Dynamic programming table

#### Pattern

- **0**
- **Compute** the \( k \)-values using offline method of Landau-Vishkin 1988.
- **Ingredient:** LCE
- **Table space:** \( O(k) \)
- **Time:** \( O(k^2) \) \((O(k)\) per symbol)

#### Stream

- **Compute** the \( k \)-values directly using the recurrence.
- **Space:** \( O(k) \)
- **Time:** \( O(k^2) \) \(O(k)\) per symbol

- **Run two** dynamic programming processes in parallel to cover every symbol in the stream.

- **Compute** \( D[m-1, i] \) for \( i \) in this interval.
- **Start work here.**

These values can be set to some constant.
Dynamic programming table

Pattern

Shared $O(m)$ space for LCE queries.

Each process: $O(k)$ space and $O(k)$ time per arriving symbol.

Multiple streams: $O(m + ks)$ space and $O(k)$ time.

Run two dynamic programming processes in parallel to cover every symbol in the stream.

These values can be set to some constant

Space: $O(k)$
Time: $O(k)$
per symbol

Compute $D[m-1, i]$ for $i$ in this interval.

Start work here.
One-way communication complexity

The equality problem

Is my string equal to Alice’s?

001010101000101101
\(n\) bits

001010101010101001
\(n\) bits
One-way communication complexity

The equality problem

Is my string equal to Alice’s?

Alice must send $n$ bits.

001010101000101101
$n$ bits

001010101010101001
$n$ bits
One-way communication complexity

The indexing problem

What’s the bit at position \( i \) of Alice’s string?

001010101000101101

\( n \) bits

Index \( i \)
One-way communication complexity

The indexing problem

What’s the bit at position $i$ of Alice’s string?

Alice must send $n$ bits.

001010101000101101
$n$ bits

Index $i$
Space lower bound \((k\text{-mismatch/difference})\)

Part 1 – the equality problem

Has pattern \(P\) over alphabet \(\Sigma\)
\[ = m \log |\Sigma| \text{ bits.} \]

Bit string \(T\) of length 
\[ m \log |\Sigma|. \]
Space lower bound ($k$-mismatch/difference)

Part 1 – the equality problem

**Step 1**
Sends internal state of pattern matching machine on $P$.

Has pattern $P$ over alphabet $\Sigma = m \log |\Sigma|$ bits.

Bit string $T$ of length $m \log |\Sigma|$.
Space lower bound \((k\text{-mismatch/difference})\)

Part 1 – the equality problem

**Step 1**
Sends internal state of pattern matching machine on \(P\).

Has pattern \(P\) over alphabet \(\Sigma\) 

\[
= m \log |\Sigma| \text{ bits.}
\]

**Step 2**
Bob feeds \(T\) into one stream to determine if \(P = T\).
Space lower bound ($k$-mismatch/difference)

Part 1 – the equality problem

**Step 1**
Sends internal state of pattern matching machine on $P$.

Has pattern $P$ over alphabet $\Sigma = m \log |\Sigma|$ bits.

Bit string $T$ of length $m \log |\Sigma|$.

**Step 2**
Bob feeds $T$ into one stream to determine if $P = T$.

**Conclusion:** Space must be $\Omega(m \log |\Sigma|)$ bits.
Space lower bound ($k$-mismatch/difference)

Part 2 – the indexing problem

Has $k_s$-length bit string  

Index $i$
Space lower bound ($k$-mismatch/difference)

Part 2 – the indexing problem

Has $k$s-length bit string

**Step 1**
Pattern matching machine:
Alphabet $\Sigma = \{0, 1\}$.
$P = 00 \cdots 0$ ($m$ zeros).
Feeds in $k$ bits into each stream.

Index $i$
Space lower bound ($k$-mismatch/difference)

Part 2 – the indexing problem

**Step 1**
Pattern matching machine:
Alphabet $\Sigma = \{0, 1\}$.
$P = 00 \cdots 0$ ($m$ zeros).
Feeds in $k$ bits into each stream.

**Step 2**
Sends internal state

Has $ks$-length bit string

Index $i$
Space lower bound ($k$-mismatch/difference)

Part 2 – the indexing problem

**Step 2**
Sends internal state

Has $k$s-length bit string

**Step 1**
Pattern matching machine:
Alphabet $\Sigma = \{0, 1\}$.
$P = 00 \cdots 0$ ($m$ zeros).
Feeds in $k$ bits into each stream.

**Step 3**
Bob feeds 0s into the appropriate stream, takes the outputted distance, feeds in another 0 and compare the two distances. This reveals the bit at position $i$. 
Space lower bound ($k$-mismatch/difference)

Part 2 – the indexing problem

**Step 1**
Pattern matching machine:
Alphabet $\Sigma = \{0, 1\}$.
$P = 00 \cdots 0$ ($m$ zeros).
Feeds in $k$ bits into each stream.

**Step 2**

**Conclusion:** Space must be $\Omega(ks)$ bits.

Combining Parts 1 and 2: $\Omega(m \log |\Sigma| + ks)$ bits of space.

**Step 3**
Bob feeds 0s into the appropriate stream, takes the outputted distance, feeds in another 0 and compare the two distances. This reveals the bit at position $i$. Has $ks$-length bit string Index $i$
Space lower bound ($k$-mismatch/difference)

Part 2 – the indexing problem

**Step 2**

**Conclusion:** Space must be $\Omega(ks)$ bits.

Combining Parts 1 and 2: $\Omega(m \log |\Sigma| + ks)$ bits of space.

The bounds $\Omega(m \log |\Sigma| + s)$ for **exact matching** and $\Omega(ms)$ for $L_1$, $L_2$, **Hamming distance** and **convolution** are obtained similarly.

Pattern matching machine: Bob feeds 0s into the appropriate stream, takes the outputted distance, feeds in another 0 and compare the two distances. This reveals the bit at position $i$.

Alphabet $\Sigma = \{0, 1\}$.

$P = 00 \cdots 0$ ($m$ zeros).

Feeds in $k$ bits into each stream.
Open problems

- Close the gap for $k$-mismatch:
  Our $O(k)$ time versus $O(\sqrt{k \log k})$ offline.

  Potentially exponential gap for constant size alphabets:
  Our $O(k)$ time versus $O(\log^2 m)$ in a single stream.

- Randomised space lower bound for $k$-mismatch/difference is $O(\log m + ks)$ and $O(\log m + s)$ for exact matching. Can we get (near) matching upper bounds?

- Conjecture: for every multiple-streams algorithm, there is an equivalent (time and space) one with read-only space that is independent of $s$ (like our bounds).
Thank you!