Recuperació de la informació

• Modern Information Retrieval (1999)
  Ricardo-Baeza Yates and Berthier Ribeiro-Neto

• Flexible Pattern Matching in Strings (2002)
  Gonzalo Navarro and Mathieu Raffinot

• Algorithms on strings (2001)
  M. Crochemore, C. Hancart and T. Lecroq

• http://www-igm.univ-mlv.fr/~lecroq/string/index.html
String matching: definition of the problem (text, pattern)

- **Exact matching:** depends on what we have: text or patterns
  - The patterns ---» Data structures for the patterns
    - 1 pattern ---» The algorithm depends on $|p|$ and $|\Sigma|$  
    - $k$ patterns ---» The algorithm depends on $k$, $|p|$ and $|\Sigma|$  
  
- **Extensions**
- **Regular Expressions**
- **The text** ---» Data structure for the text (suffix tree, ...)

- **Approximate matching:**
  - Dynamic programming
  - Sequence alignment (pairwise and multiple)
  - Sequence assembly: hash algorithm

- **Probabilistic search:** Hidden Markov Models
Extended string matching

There are classes of characters represented by one symbol. For instance the IUPAC code for the DNA alphabet is:

R = \{G,A\} \quad Y = \{T,C\} \quad K = \{G,T\} \quad M = \{A,C\} \quad S = \{G,C\} \quad W = \{A,T\}
B = \{G,T,C\} \quad D = \{G,A,T\} \quad H = \{A,C,T\} \quad V = \{G,C,A\} \quad N = \{A,G,C,T\} \quad (any)

1. Classes of characters in the text.

   There are characters in the text that represent sets of symbols

2. Classes of characters in the pattern.

   There are characters in the pattern that represent sets of symbols
Extended alphabets

First part

Classes in the text
• How the comparison is made?

Text: over $2^{|\Sigma|}$

Pattern over $\Sigma$

From left to right: prefix

We need the operation: belongs to a set?

• Which is the next position of the window?

Text:

Pattern:

The window is shifted only one cell
When $|\Sigma| < \text{computer word}$

Every subset of $\Sigma$ is represented by a string of bits of length $|\Sigma|$.

For instance, given the DNA alphabet $\Sigma=\{A,C,G,T\}$:
$I(A)=(1,0,0,0), \ I(C)=(0,1,0,0), \ldots \ I(R)=I(G,A)=( \ , \ , \ )$
When $|\Sigma| < \text{computer word}$

Every subset of $\Sigma$ is represented by a string of bits of length $|\Sigma|$.

For instance, given the DNA alphabet $\Sigma=\{A,C,G,T\}$:

$I(A)=(1,0,0,0)$, $I(C)=(0,1,0,0)$, \ldots $I(R)=I(G,A)=(1,0,1,0)$, \ldots $I(N)=(\ ,\ ,\ ,\ )$
When $|\Sigma| < \text{computer word}$

Every subset of $\Sigma$ is represented by a string of bits of length $|\Sigma|$.

For instance, given the DNA alphabet $\Sigma=\{A,C,G,T\}$:

$I(A)=(1,0,0,0), \ I(C)=(0,1,0,0),... \ I(R)=I(G,A)=(1,0,1,0)\ldots I(N)=(1,1,1,1)$

Then the operation “A belongs to set $X$” is made with ...
Classes in the text: Brute force algorithm

When $|\Sigma| < \text{computer word}$

Every subset of $\Sigma$ is represented by a string of bits of length $|\Sigma|$.

For instance, given the DNA alphabet $\Sigma=\{A,C,G,T\}$:

$I(A)=(1,0,0,0)$, $I(C)=(0,1,0,0)$, ... $I(R)=I(G,A)=(1,0,1,0)$... $I(N)=(1,1,1,1)$

Then the operation “$A \text{ belongs to set } X$” is made with $I(A)$ and $I(X) > 0$

$$\begin{align*}
\text{G T A R T R N A G G A ...} & \\
\text{ATGTA} & \\
\text{ATGTA} & \text{I(A) & I(T)>0}
\end{align*}$$
When $|\Sigma| < \text{computer word}$

Every subset of $\Sigma$ is represented by a string of bits of length $|\Sigma|$.

For instance, given the DNA alphabet $\Sigma$={$A,C,G,T$}:

$I(A)=$(1,0,0,0), $I(C)=$(0,1,0,0),... $I(R)=$$I(G,A)=$(1,0,1,0)...$I(N)=$(1,1,1,1)

Then the operation “$A \text{ belongs to set } X$” is made with $I(A)$ and $I(X)>0$
When $|\Sigma| < \text{computer word}$

Every subset of $\Sigma$ is represented by a string of bits of length $|\Sigma|$.

For instance, given the DNA alphabet $\Sigma=${A,C,G,T}:

$I(A)=(1,0,0,0)$, $I(C)=(0,1,0,0)$, ..., $I(R)=I(G,A)=(1,0,1,0)$...$I(N)=(1,1,1,1)$

Then the operation “A belongs to set $X$” is made with $I(A)$ and $I(X) > 0$

```
G T A R T R N A G G A ...
ATGTA
ATGTA
ATGTA
...
```

Which is the cost?
Experimental efficiency (Navarro & Raffinot)

BNDM : Backward Nondeterministic Dawg Matching
BOM : Backward Oracle Matching
Classes in the text: Horspool algorithm

• How the comparison is made?

Text:

Pattern:

← Suffix search

• Which is the next position of the window?

Text:

Pattern:

Shift until the next occurrence of “a” (or “t”, “r”, …) in the pattern:

We need a shift table with the extended alphabet.
Given the pattern ATGTA

- The shift table is:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>R</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>?</td>
</tr>
</tbody>
</table>
Given the pattern ATGTA

• The shift table is:

<table>
<thead>
<tr>
<th>Letter</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>R</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>?</td>
</tr>
</tbody>
</table>
Classes in the text: Horspool example

Given the pattern ATGTA

- The shift table is:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>R</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>1</td>
</tr>
</tbody>
</table>

text: G T A R T R N A A G G A ...

ATGTA

ATGTA

ATGTA

ATGTA
Given the pattern ATGTA

• The shift table is:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>R</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>1</td>
</tr>
</tbody>
</table>

text : GTA RTRNAAGGA ...
Experimental efficiency (Navarro & Raffinot)

BNDM : Backward Nondeterministic Dawg Matching
BOM : Backward Oracle Matching

Classes in the text
Search for suffixes of T that are factors of the pattern

How the comparison is made?

Text:

Pattern:

\[ D_2 = 1\, 0\, 0\, 0\, 1\, 0\, 0 \]

\[ D_3 = D_2 \ll 1 \& B(x) \]

Once the next character x is read

\[ B(x): \text{mask of } x \text{ in the pattern } P. \]

For instance, if \( B(x) = (0\, 0\, 1\, 1\, 0\, 0\, 0) \)

\[ D = (0\, 0\, 0\, 1\, 0\, 0\, 0) \& (0\, 0\, 1\, 1\, 0\, 0\, 0) = (0\, 0\, 0\, 1\, 0\, 0\, 0) \]

Which is the next position of the window?

 Depends on the value of the leftmost bit of D
Given the pattern ATGTA

• The masks of bits are

\[
\begin{align*}
B(A) &= (1 \ 0 \ 0 \ 0 \ 1) \\
B(C) &= (0 \ 0 \ 0 \ 0 \ 0) \\
B(G) &= (0 \ 0 \ 1 \ 0 \ 0) \\
B(T) &= (0 \ 1 \ 0 \ 1 \ 0)
\end{align*}
\]
Given the pattern ATGTA

• The masks of bits are

\[
\begin{align*}
B(A) &= (10001) \\
B(C) &= (00000) \\
B(G) &= (00100) \\
B(T) &= (01010)
\end{align*}
\]

\[
B(R) = (10101) \\
B(N) = (\ )
\]
Given the pattern ATGTA

• The masks of bits are

<table>
<thead>
<tr>
<th>Letter</th>
<th>Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(1 0 0 0 1)</td>
</tr>
<tr>
<td>R</td>
<td>(1 0 1 0 1)</td>
</tr>
<tr>
<td>C</td>
<td>(0 0 0 0 0)</td>
</tr>
<tr>
<td>G</td>
<td>(0 0 1 0 0)</td>
</tr>
<tr>
<td>N</td>
<td>(1 1 1 1 1)</td>
</tr>
<tr>
<td>T</td>
<td>(0 1 0 1 0)</td>
</tr>
</tbody>
</table>

• text: GTARTRNAGGAGC...
Classes in the text : BNDM example

Given the pattern ATGTA

• The masks of bits are

\[
\begin{align*}
B(A) &= (10001) \\
B(R) &= (10101) \\
B(C) &= (00000) \\
B(G) &= (00100) \\
B(N) &= (11111) \\
B(T) &= (01010)
\end{align*}
\]

• text : G T A R T R N A G G A C G ...  

\[
\text{ATGTA}
\]

\[
\begin{align*}
D_1 &= (01010) \\
D_2 &= (10100) \& (10101) = (10100) \\
D_2 &= (01000) \& (10001) = (00000) \\
\end{align*}
\]

\[
\begin{align*}
D_1 &= (10001) \\
D_2 &= (00010) \& (11111) = (00010) \\
D_3 &= (00100) \& (10101) = (00100) \\
D_4 &= (01000) \& (01010) = (01000) \\
D_5 &= (10000) \& (10101) = (10000)
\end{align*}
\]
Given the pattern ATGTA

- The masks of bits are

$B(A) = (1 0 0 0 1)$  $B(R) = (1 0 1 0 1)$
$B(C) = (0 0 0 0 0)$
$B(G) = (0 0 1 0 0)$  $B(N) = (1 1 1 1 1)$
$B(T) = (0 1 0 1 0)$

- text: G T A R T R N A G G A C G ... ATGTA

$D_1 = (0 1 0 1 0)$
$D_2 = (1 0 1 0 0) \& (1 0 1 0 1) = (1 0 1 0 0)$
$D_2 = (0 1 0 0 0) \& (1 0 0 0 1) = (0 0 0 0 0)$

$D_1 = (1 0 0 0 1)$
$D_2 = (0 0 0 1 0) \& (1 1 1 1 1) = (0 0 0 1 0)$
$D_3 = (0 0 1 0 0) \& (1 0 1 0 1) = (0 0 1 0 0)$
$D_4 = (0 1 0 0 0) \& (0 1 0 1 0) = (0 1 0 0 0)$
$D_5 = (1 0 0 0 0) \& (1 0 1 0 1) = (1 0 0 0 0)$
Experimental efficiency (Navarro & Raffinot)

BNDM : Backward Nondeterministic Dawg Matching
BOM : Backward Oracle Matching
BOM algorithm (Backward Oracle Matching)

• How the comparison is made?

Text:

Pattern: Automata: Factor Oracle

Check if the suffix is a factor

• Which is the next position of the window?

The position determined by the last character of the text with a transition in the automata
The we build the AFO of the inverse pattern of ATGTATG

... and we try to find... : G T A R T R N A A T G...

ATGATATG

It’s not possible any improvement!
• How the comparison is made?

Text:

Patterns:

Trie of all inverse patterns

• Which is the next position of the window?

?  

By suffixes
Set Horspool algorithm

Search for ATGTATG, TATG, ATAAT, ATGTG

1. Construct the trie of GTATGTA, GTAT, TAATA and GTGTA

2. Determine $l_{\text{min}} = 4$

3. Determine the shift table

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4 (l_{\text{min}})</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

4. Find the patterns
Classes in the text: Set Horspool

Search for the patterns
ATGTATG, TATG, ATAAT, ATGTG

text: ARTGNCTATGTGACA…

It’s not possible any improvement!
Multiple string matching

Wu-Manber

SBOM

| Σ |

8

Wu-Manber

Ad AC

SBOM

(5 strings)

(10 strings)

(100 strings)

(1000 strings)
• How the comparison is made?

Text:

Pattern: Automata: Factor Oracle (Inverse patterns of length $l_{min}$)

Check if the suffix is a factor of any pattern

• Which is the next position of the window?

The position determined by the last character of the text with a transition in the automata
Search for the patterns ATGTATG, TAATG, TAATAAT i AATGTG

It’s not possible any improvement!
### Extended alphabets

#### Classes in the:

<table>
<thead>
<tr>
<th></th>
<th>text</th>
<th>pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horspool</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>BNDM</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>BOM</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Set-Horspool</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>SBOM</td>
<td>✗</td>
<td></td>
</tr>
</tbody>
</table>
Second part

Classes in the pattern
Classes in the pattern: Brute force algorithm

- How the comparison is made?

Text: \( \overline{\Sigma} \)
Pattern: \( \overline{2^{\mid \Sigma \mid}} \)

From left to right: prefix

We need the operation: belongs to a set?

- Which is the next position of the window?

Text:
Pattern:

The window is shifted only one cell
When $|\Sigma| < \text{ computer word word} $

Every subset is represented by a string of bits of length $|\Sigma|$. For instance, given the DNA alphabet $\Sigma = \{A,C,G,T\}$:

$I(A) = (1,0,0,0)$, $I(C) = (0,1,0,0)$, ..., $I(R) = (1,0,1,0)$, ..., $I(N) = (1,1,1,1)$

Then the operation “$A \text{ belongs to set } X$” is made with $I(A)$ and $I(X) > 0$

$G \ T \ A \ C \ T \ A \ G \ A \ G \ G \ A \ C \ G \ T \ A \ T \ G \ T \ A \ C \ T \ G \ ...

I(T) \text{ and } I(R) > 0

I(A) \text{ and } I(R) > 0

I(T) \text{ and } I(T) > 0

I(C) \text{ and } I(N) > 0

I(A) \text{ and } I(T) > 0

...
Experimental efficiency (Navarro & Raffinot)

BNDM : Backward Nondeterministic Dawg Matching
BOM : Backward Oracle Matching
Classes in the pattern: Horspool algorithm

- **How the comparison is made?**

  Text:
  
  Pattern:
  
  Suffix search

- **Which is the next position of the window?**

  Text:
  
  Pattern:

  Shift until the next occurrence of “a” in the pattern:

  We need a preprocessing phase to construct the shift table.
Given the pattern ATNTR

• The shift table is:
Given the pattern ATNTR

- The shift table is:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
</tr>
</tbody>
</table>
Given the pattern ATNTR

- The shift table is:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
</tr>
</tbody>
</table>
Given the pattern ATNTR

- The shift table is:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td></td>
</tr>
</tbody>
</table>
Classes in the pattern: Horspool example

Given the pattern ATNTR

- The shift table is:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>1</td>
</tr>
</tbody>
</table>

Text: G T A C T A G A T A T G A G ...

ATNTR
ATNTR
ATNTR
ATNTR
ATNTR
Given the pattern ATNTR

• The shift table is:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

text: GTA C T A G AT A T G A G ...

Shorter shifts!
Classes in the text

Experimental efficiency (Navarro & Raffinot)

BNDM : Backward Nondeterministic Dawg Matching
BOM : Backward Oracle Matching

Long. pattern

\[ |\Sigma| \]

Horspool

BNDM

BOM
• How the comparison is made?

Search for suffixes of T that are factors of the pattern

Text:

Pattern:

...that is denoted as

\[ D_2 = 1 \circled{0} 0 \ 0 \ 0 \ 1 \ 0 \ 0 \]

Once the next character \( x \) is read

\[ D_3 = D_2 \ll 1 \& B(x) \]

\( B(x) \): mask of \( x \) in the pattern \( P \).

For instance, if \( B(x) = (0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0) \)

\[ D = (0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0) \& (0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0) = (0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0) \]

• Which is the next position of the window?

Depends on the value of the leftmost bit of \( D \)
Classes in the pattern : BNDM example

Given the pattern ATNTR

- The masks of bits of symbols are

  \[ B(A) = ( \quad ) \]
  \[ B(C) = ( \quad ) \]
  \[ B(G) = ( \quad ) \]
  \[ B(T) = ( \quad ) \]
Classes in the pattern: BNDM example

Given the pattern ATNTR

• The masks of bits of symbols are

B(A) = (1 0 1 0 1)
B(C) = (               )
B(G) = (               )
B(T) = (               )
Classes in the pattern : BNDM example

Given the pattern ATNTR

• The masks of bits of symbols are

B(A) = (1 0 1 0 1)
B(C) = (0 0 1 0 0)
B(G) = (               )
B(T) = (               )
Given the pattern ATNTR

• The masks of bits of symbols are

  B(A) = (1 0 1 0 1)
  B(C) = (0 0 1 0 0)
  B(G) = (0 0 1 0 1)
  B(T) = (     )
Classes in the pattern : BNDM example

Given the pattern ATNTR

- The masks of bits of symbols are

  \[ B(A) = (11011) \]
  \[ B(C) = (00100) \]
  \[ B(G) = (00101) \]
  \[ B(T) = (01110) \]

- text : \[ C G T A C T A G A G G A C G T A T G T A C T G \ldots \]

\[ D_1 = (01110) \]
\[ D_2 = (11100) \land (00100) = (00100) \]
\[ D_3 = (01000) \land (10101) = (00000) \]
\[ D_1 = (00101) \]
\[ D_2 = (01010) \land (00101) = (00000) \]
\[ D_1 = (10101) \]
\[ D_2 = (01010) \land (01110) = (01010) \]
\[ D_3 = (10100) \land (00101) = (00100) \]
\[ D_4 = (01000) \land (00100) = (00000) \]
\[ D_1 = (10101) \]
\[ D_2 = (01010) \land (01110) = (01010) \]
\[ D_3 = (10100) \land (00101) = (00100) \]
\[ D_4 = (01000) \land (00100) = (00000) \]
Experimental efficiency (Navarro & Raffinot)

BNDM : Backward Nondeterministic Dawg Matching
BOM : Backward Oracle Matching

<table>
<thead>
<tr>
<th>Σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>32</td>
</tr>
<tr>
<td>64</td>
</tr>
<tr>
<td>128</td>
</tr>
<tr>
<td>256</td>
</tr>
</tbody>
</table>

Long. pattern
BOM algorithm (Backward Oracle Matching)

- How the comparison is made?

Text:

Pattern: Automata: Factor Oracle

Check if the suffix is a factor

- Which is the next position of the window?

The position determined by the last character of the text with a transition in the automata
• Given the pattern ATGTATG, the AFO is

\[
\begin{align*}
&\text{G} \quad \text{T} \quad \text{A} \quad \text{T} \quad \text{G} \quad \text{T} \quad \text{A} \\
&\quad \text{T} \quad \text{A} \quad \text{G} \quad \text{T} \\
&\quad \text{A} \quad \text{T} 
\end{align*}
\]

but for the pattern ATNTRTG?

We should apply the SBOM algorithm!
Set Horspool algorithm

- **How the comparison is made?**

  **Text:**
  
  By suffixes

  **Patterns:**
  
  Trie of all inverse patterns

- **Which is the next position of the window?**

  We shift until a is aligned with the first a in the trie not longer than $l_{min}$, or $l_{min}$
Set Horspool algorithm

Search for ATNTARG, RTGR, NTTNAR, ATRTG

1. Construct the trie of the 46 possible inverse patterns

2. Determine $l_{min}=4$

3. Determine the shift table

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>1</td>
</tr>
</tbody>
</table>

4. Find the patterns
Multiple string matching

Wu-Manber

SBOM

Ad AC

SBOM

SBOM

(5 strings)

(10 strings)

(100 strings)

(1000 strings)
SBOM algorithm

- How the comparison is made?

Text:

Pattern: Automata: Factor Oracle (Inverse patterns of length $l_{\text{min}}$)

Check if the suffix is a factor of any pattern

- Which is the next position of the window?

The position determined by the last character of the text with a transition in the automata
Given the patterns
ATGNARG, TRATR, TAATAAT i ANTNTGR

the Automata Factor Oracle of all 21 possible patterns is built …
Multiple string matching

Wu-Manber

SBOM

|Σ|

Ad AC

Wu-Manber

SBOM

(5 strings)

(10 strings)

(100 strings)

(1000 strings)
## Extended alphabets

### Classes in the:

<table>
<thead>
<tr>
<th></th>
<th>text</th>
<th>pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horspool</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>BNDM</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>BOM</td>
<td>✗</td>
<td>≈</td>
</tr>
<tr>
<td>Set-Horspool</td>
<td>✗</td>
<td>≈</td>
</tr>
<tr>
<td>SBOM</td>
<td>✗</td>
<td>≈</td>
</tr>
</tbody>
</table>