Pattern matching

- A basic tool for information retrieval
  - find all positions that a pattern occurs in a text

\[ t = b \ b \ a \ a \ a \ b \ b \ a \ a \ a \ b \ b \ a \ a \ a \ b \ a \ a \ a \ b \ b \ b \ b \ b \ b \ b \ b \ a \ a \]

\[ p = a \ a \ b \]

- Many applications
  - Text search
  - Genome analysis
  - Data compression
Permuted pattern matching problem [Katsura+, SOFSEM2013]

- A variant of the pattern matching
  - On multi-track strings

\[ T = \begin{pmatrix}
  bbaaabbbaaabbaaaababbbbbbbaa \\
  aaaaabbbaaaabbaaababaabbbbbaaabbba \\
  aabbbabbaaaababbaaaabbbbaaab
\end{pmatrix} \]

\[ P = \begin{pmatrix}
  ab \\
  ba \\
  aa
\end{pmatrix} \]

Permuted-match at position \( i \):
all strings of \( P \) occur at \( i \) in \( T \) in any order

- Applications
  - Multi-sensor data
  - Polyphonic music data
Indexing structure

- Preprocessing a text in order to support fast search
  - Suffix tree [Weigner, ’73]
  - Suffix array [Manber and Mayers, ’93]
  - Position heap [Ehrenfeucht+, ’11]

\[ t = a \ b \ a \ a \ b \ b \ a \ b \ b \ c \ \$ \]
\[ p = a \ b \ b \]

Suffix tree of \( t \)

Position heap of \( t \)
## Contributions

- Propose **new indexing structures** for multi-track strings

<table>
<thead>
<tr>
<th>Indexing structure</th>
<th>Matching time</th>
<th>Space</th>
<th>Construction time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-Track Suffix Tree [Katsura+, SOFSEM2013]</td>
<td>$O( mN \log</td>
<td>\Sigma</td>
<td>+ occ )$</td>
</tr>
<tr>
<td>new Multi-Track Position Heap</td>
<td>$O( m^2N \log</td>
<td>\Sigma</td>
<td>+ occ )$</td>
</tr>
<tr>
<td>new Contracted Multi-Track Position Heap</td>
<td>$O( m^2N^2 \log</td>
<td>\Sigma</td>
<td>+ occ )$</td>
</tr>
</tbody>
</table>

- **fast**
- **memory efficient**

```
text
T = (bbababbbaaababbbaaabbbaabbbaaababbbbbbabbaaababbababbaaababbbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaaabbbbabbaa... | m |
| pattern P = (bab | a a ) N | $|\Sigma|$: alphabet size | $occ$: the number of occurrences of the pattern

```

---

$n$, $m$, $N$, $\Sigma$, and $occ$ represent the number of occurrences of the pattern.