Lecture 13
Approximate pattern matching (part two)

Benjamin Sach
Pattern matching with mismatches (Hamming distance)

**Input** A text string $T$ (length $n$) and a pattern string $P$ (length $m$)

<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>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>a</td>
<td>d</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>$P$</td>
<td>a</td>
<td>b</td>
<td>d</td>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Goal:** For all $i$, output, $\text{Ham}(i)$, the Hamming distance between $P$ and $T[i \ldots i + m - 1]$

*The Hamming distance is the number of (single character) mismatches...*  
i.e. the number of distinct $j$ such that $P[j] \neq T[i + j]
Pattern matching with mismatches (Hamming distance)

**Input** A text string $T$ (length $n$) and a pattern string $P$ (length $m$)

\[
\begin{array}{cccccccccccc}
T & a & b & c & d & a & b & a & a & d & a & c & a & a \\
\hline
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
\end{array}
\]

\[
\begin{array}{cccc}
P & a & b & d & a \\
\hline
\end{array}
\]

Ham(4) = 1

**Goal:** For all $i$, output, Ham($i$), the Hamming distance between $P$ and $T[i \ldots i + m - 1]$

*The Hamming distance is the number of (single character) mismatches...*

i.e. the number of distinct $j$ such that $P[j] \neq T[i + j]$
Pattern matching with mismatches (Hamming distance)

**Input** A text string $T$ (length $n$) and a pattern string $P$ (length $m$)

<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>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>a</td>
<td>d</td>
<td>a</td>
<td>c</td>
<td>a</td>
</tr>
</tbody>
</table>

$P$

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>b</td>
<td>d</td>
<td>a</td>
</tr>
</tbody>
</table>

$\text{Ham}(5) = 4$

**Goal:** For all $i$, output, $\text{Ham}(i)$, the Hamming distance between $P$ and $T[i \ldots i + m - 1]$

*The Hamming distance is the number of (single character) mismatches...*  
...i.e. the number of distinct $j$ such that $P[j] \neq T[i + j]$
Pattern matching with mismatches (Hamming distance)

**Input** A text string $T$ (length $n$) and a pattern string $P$ (length $m$)

$$
\begin{array}{cccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
T & a & b & c & d & a & b & a & \textcolor{red}{a} & d & a & c & a & a \\
\end{array}
$$

$$
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
P & a & b & d & a \\
\end{array}
$$

$$\text{Ham}(6) = 1$$

**Goal:** For all $i$, output, $\text{Ham}(i)$, the Hamming distance between $P$ and $T[i \ldots i + m - 1]$

*The Hamming distance is the number of (single character) mismatches...*  
i.e. the number of distinct $j$ such that $P[j] \neq T[i + j]$
Pattern matching with mismatches (Hamming distance)

**Input** A text string $T$ (length $n$) and a pattern string $P$ (length $m$)

|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|----|----|----|
| $T$ | a | b | c | d | a | b | a | a | d | a | c | a | a |
| $P$ | a | b | d | a |

**Goal:** For all $i$, output, $\text{Ham}(i)$, the Hamming distance between $P$ and $T[i \ldots i + m - 1]$

*The Hamming distance is the number of (single character) mismatches...*  
  i.e. the number of distinct $j$ such that $P[j] \neq T[i + j]$
Pattern matching with mismatches (Hamming distance)

**Input** A text string $T$ (length $n$) and a pattern string $P$ (length $m$)

<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>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>a</td>
<td>d</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>$P$</td>
<td>a</td>
<td>b</td>
<td>d</td>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Hamming distance is the number of (single character) mismatches...

i.e. the number of distinct $j$ such that $P[j] \neq T[i + j]$

**Goal:** For all $i$, output, $\text{Ham}(i)$, the Hamming distance between $P$ and $T[i \ldots i + m - 1]$

$\text{Ham}(8) = 3$
Pattern matching with mismatches (Hamming distance)

**Input** A text string $T$ (length $n$) and a pattern string $P$ (length $m$)

\[
\begin{array}{cccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
\hline
T & a & b & c & d & a & b & a & a & d & a & c & a & a \\
P & a & b & d & a \\
\end{array}
\]

Ham(8) = 3

**Goal:** For all $i$, output, Ham(i), the Hamming distance between $P$ and $T[i \ldots i + m - 1]$

*The Hamming distance is the number of (single character) mismatches...*

i.e. the number of distinct $j$ such that $P[j] \neq T[i + j]$

• We saw an $O(n|\Sigma| \log m)$ time algorithm last lecture...
Pattern matching with mismatches (Hamming distance)

**Input** A text string $T$ (length $n$) and a pattern string $P$ (length $m$)

![Text and Pattern Strings](image)

**Goal:** For all $i$, output, $\text{Ham}(i)$, the Hamming distance between $P$ and $T[i \ldots i + m - 1]$

*The Hamming distance is the number of (single character) mismatches... i.e. the number of distinct $j$ such that $P[j] \neq T[i + j]*

- We saw an $O(n|\Sigma| \log m)$ time algorithm last lecture... *where $|\Sigma|$ is the number of alphabet symbols*
Hamming distance - considering symbols separately

Imagine that the alphabet contains only a small number of different symbols, which we consider individually...

\[ T = \begin{array}{cccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
\hline
a & b & c & d & a & b & a & a & d & a & c & a & a \\
\end{array} \]

\[ P = \begin{array}{cccc}
0 & 1 & 2 & 3 \\
\hline
a & b & d & a \\
\end{array} \]

Imagine that the alphabet contains only a small number of different symbols, which we consider individually...
Hamming distance - considering symbols separately

Imagine that the alphabet contains only a small number of different symbols, which we consider individually...

Replace all $a$ symbols with 1 and everything else with 0
Hamming distance - considering symbols separately

Imagine that the alphabet contains only a small number of different symbols, which we consider individually...

Replace all $a$ symbols with 1 and everything else with 0.
Hamming distance - considering symbols separately

Imagine that the alphabet contains only a small number of different symbols, which we consider individually...

\[
\begin{array}{cccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
\hline
T & \text{a} & \text{b} & \text{c} & \text{d} & \text{a} & \text{b} & \text{a} & \text{a} & \text{d} & \text{a} & \text{c} & \text{a} & \text{a} \\
\hline
P & \text{a} & \text{b} & \text{d} & \text{a} \\
\hline
\end{array}
\]

Replace all \text{a} symbols with 1 and everything else with 0.
Hamming distance - considering symbols separately

Imagine that the alphabet contains only a small number of different symbols, which we consider individually...

Replace all *a* symbols with 1 and everything else with 0.
Hamming distance - considering symbols separately

Imagine that the alphabet contains only a small number of different symbols, which we consider individually...

\[
\begin{array}{cccccccccccc}
\hline
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
\hline
\end{array}
\]

\[
T \quad \begin{array}{cccccccccccc}
1 & b & c & d & 1 & b & 1 & 1 & d & 1 & c & 1 & 1 \\
\end{array}
\]

\[
P \quad \begin{array}{cccccccccccc}
1 & b & d & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccccccc}
\hline
m \\
\hline
\end{array}
\]

Replace all \(a\) symbols with 1 and everything else with 0.
Hamming distance - considering symbols separately

Imagine that the alphabet contains only a small number of different symbols, which we consider individually...

Replace all $a$ symbols with 1 and everything else with 0.
Hamming distance - considering symbols separately

Imagine that the alphabet contains only a small number of different symbols, which we consider individually...

Replace all $a$ symbols with 1 and everything else with 0
Hamming distance - considering symbols separately

Imagine that the alphabet contains only a small number of different symbols, which we consider individually...

\[ T_a \]

\[ P_a \]

Replace all \( \alpha \) symbols with 1 and everything else with 0

We denote these new strings \( T_a \) and \( P_a \) and consider...
Hamming distance - considering symbols separately

Imagine that the alphabet contains only a small number of different symbols, which we consider individually...

Replace all $a$ symbols with 1 and everything else with 0

We denote these new strings $T_a$ and $P_a$ and consider...
Hamming distance - considering symbols seperately

Imagine that the alphabet contains only a small number of different symbols, which we consider individually...

\[
\begin{array}{cccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
\hline
T_a & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\
P_a & 1 & 0 & 0 & 1 \\
\end{array}
\]

Replace all \( a \) symbols with 1 and everything else with 0

We denote these new strings \( T_a \) and \( P_a \) and consider...

\[
(T_a \otimes P_a)[i] = \sum_{j=0}^{m-1} P_a[j] T_a[i + j]
\]

1 iff \( P[j] = T[i + j] = a \)
Hamming distance - considering symbols separately

Imagine that the alphabet contains only a small number of different symbols, which we consider individually...

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_a$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$P_a$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Replace all $a$ symbols with 1 and everything else with 0

We denote these new strings $T_a$ and $P_a$ and consider...

$$(T_a \otimes P_a)[i] = \sum_{j=0}^{m-1} P_a[j] T_a[i+j]$$

1 iff $P[j]=T[i+j]=a$

(1 × 1) + (2 × 2) + (1 × 1) = 6
Hamming distance - considering symbols separately

Imagine that the alphabet contains only a small number of different symbols, which we consider individually...

<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>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>T_a</strong></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>P_a</strong></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Replace all a symbols with 1 and everything else with 0

We denote these new strings $T_a$ and $P_a$ and consider...

\[
(T_a \otimes P_a)[i] = \sum_{j=0}^{m-1} P_a[j]T_a[i + j]
\]

1 iff $P[j]=T[i+j]=a$

\[
\begin{array}{cccccccc}
\ldots & 2 & 1 & 2 & 1 & 2 & 1 & 3 & \ldots \\
\times + \times + \times \\
\begin{array}{cccc}
1 & 2 & 1 & \end{array}
\end{array}
\]

\[ (1 \times 1) + (2 \times 2) + (1 \times 1) = 6 \]

This is the number of matching as at the i-th alignment.
Hamming distance - considering symbols separately

Imagine that the alphabet contains only a small number of different symbols, which we consider individually...

<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>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ T_a \]

\[ P_a \]

Replace all \( a \) symbols with 1 and everything else with 0

We denote these new strings \( T_a \) and \( P_a \) and consider...

\[(T_a \otimes P_a)[i] = \sum_{j=0}^{m-1} P_a[j]T_a[i+j] \]

1 iff \( P[j] = T[i+j] = a \)

\( (1 \times 1) + (2 \times 2) + (1 \times 1) = 6 \)

This is the number of matching \( a \)s at the i-th alignment.
Hamming distance - considering symbols seperately

Imagine that the alphabet contains only a small number of different symbols, which we consider individually...

\[
\begin{array}{ccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
\hline
n \\
T_a & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\
P_a & 1 & 0 & 0 & 1 \\
\hline
m \\
\end{array}
\]

Replace all \(a\) symbols with 1 and everything else with 0

We denote these new strings \(T_a\) and \(P_a\) and consider...

\[
(T_a \otimes P_a)[i] = \sum_{j=0}^{m-1} P_a[j]T_a[i + j]
\]

\[
1 \text{ iff } P[j]=T[i+j]=a
\]

This is the number of matching \(a\)s at the \(i\)-th alignment.

which we can compute (for all \(i\)) in \(O(n \log m)\) time via cross-correlations
Hamming distance - considering symbols separately

We saw how to find all matches with a single symbol in $O(n \log m)$ time.

Let $\Sigma$ denote the set of alphabet symbols and $|\Sigma|$ be its size.

**Algorithm Summary**

1. Construct $T_{\sigma}$ and $P_{\sigma}$ for every symbol $\sigma$ in $\Sigma$.
2. Compute $T_{\sigma} \otimes P_{\sigma}$ (for every symbol $\sigma$ in $\Sigma$).
3. For every $i$, compute,

   $$ \text{Ham}(i) = m - \sum_{\sigma \in \Sigma} (T_{\sigma} \otimes P_{\sigma})[i]. $$
Hamming distance - considering symbols separately

We saw how to find all matches with a single symbol in $O(n \log m)$ time.

Let $\Sigma$ denote the set of alphabet symbols and $|\Sigma|$ be its size.

**Algorithm Summary**

- Construct $T_\sigma$ and $P_\sigma$ for every symbol $\sigma$ in $\Sigma$.
- Compute $T_\sigma \otimes P_\sigma$ (for every symbol $\sigma$ in $\Sigma$).
- For every $i$, compute,

$$\text{Ham}(i) = m - \sum_{\sigma \in \Sigma} (T_\sigma \otimes P_\sigma)[i].$$

matches involving $\sigma$
Hamming distance - considering symbols separately

We saw how to find all matches with a single symbol in $O(n \log m)$ time.

Let $\Sigma$ denote the set of alphabet symbols and $|\Sigma|$ be its size.

Algorithm Summary

Construct $T_\sigma$ and $P_\sigma$ for every symbol $\sigma$ in $\Sigma$.
Compute $T_\sigma \otimes P_\sigma$ (for every symbol $\sigma$ in $\Sigma$).
For every $i$, compute,

$$\text{Ham}(i) = m - \sum_{\sigma \in \Sigma} (T_\sigma \otimes P_\sigma)[i].$$

all matches
Hamming distance - considering symbols separately

We saw how to find all matches with a single symbol in $O(n \log m)$ time

Let $\Sigma$ denote the set of alphabet symbols and $|\Sigma|$ be its size

Algorithm Summary

Construct $T_{\sigma}$ and $P_{\sigma}$ for every symbol $\sigma$ in $\Sigma$
Compute $T_{\sigma} \otimes P_{\sigma}$ (for every symbol $\sigma$ in $\Sigma$)

For every $i$, compute,

$$\text{Ham}(i) = m - \sum_{\sigma \in \Sigma} (T_{\sigma} \otimes P_{\sigma})[i].$$

mismatches = $m$ - matches
Hamming distance - considering symbols separately

We saw how to find all matches with a single symbol in $O(n \log m)$ time.

Let $\Sigma$ denote the set of alphabet symbols and $|\Sigma|$ be its size.

**Algorithm Summary**

- Construct $T_\sigma$ and $P_\sigma$ for every symbol $\sigma$ in $\Sigma$.
- Compute $T_\sigma \otimes P_\sigma$ (for every symbol $\sigma$ in $\Sigma$).
- For every $i$, compute,

$$\text{Ham}(i) = m - \sum_{\sigma \in \Sigma} (T_\sigma \otimes P_\sigma)[i].$$
Hamming distance - considering symbols seperately

We saw how to find all matches with a single symbol in $O(n \log m)$ time

Let $\Sigma$ denote the set of alphabet symbols and $|\Sigma|$ be its size

Algorithm Summary

Construct $T_\sigma$ and $P_\sigma$ for every symbol $\sigma$ in $\Sigma$ \((O(n|\Sigma|)\) time)  
Compute $T_\sigma \otimes P_\sigma$ (for every symbol $\sigma$ in $\Sigma$)  
For every $i$, compute,

$$\text{Ham}(i) = m - \sum_{\sigma \in \Sigma} (T_\sigma \otimes P_\sigma)[i].$$
Hamming distance - considering symbols separately

We saw how to find all matches with a single symbol in $O(n \log m)$ time

Let $\Sigma$ denote the set of alphabet symbols and $|\Sigma|$ be its size

Algorithm Summary

Construct $T_\sigma$ and $P_\sigma$ for every symbol $\sigma$ in $\Sigma$ \( (O(n|\Sigma|) \text{ time}) \)
Compute $T_\sigma \otimes P_\sigma$ (for every symbol $\sigma$ in $\Sigma$) \( (O(n|\Sigma| \log m) \text{ time}) \)
For every $i$, compute,

$$\text{Ham}(i) = m - \sum_{\sigma \in \Sigma} (T_\sigma \otimes P_\sigma)[i].$$
Hamming distance - considering symbols separately

We saw how to find all matches with a single symbol in $O(n \log m)$ time

Let $\Sigma$ denote the set of alphabet symbols and $|\Sigma|$ be its size

Algorithm Summary

Construct $T_\sigma$ and $P_\sigma$ for every symbol $\sigma$ in $\Sigma$ \hspace{1cm} (O(n|\Sigma|) time)

Compute $T_\sigma \otimes P_\sigma$ (for every symbol $\sigma$ in $\Sigma$) \hspace{1cm} (O(n|\Sigma| \log m) time)

For every $i$, compute,

$$\text{Ham}(i) = m - \sum_{\sigma \in \Sigma} (T_\sigma \otimes P_\sigma)[i].$$ \hspace{1cm} (O(n|\Sigma|) time)
Hamming distance - considering symbols separately

We saw how to find all matches with a single symbol in $O(n \log m)$ time

Let $\Sigma$ denote the set of alphabet symbols and $|\Sigma|$ be its size

**Algorithm Summary**

1. Construct $T_{\sigma}$ and $P_{\sigma}$ for every symbol $\sigma$ in $\Sigma$ ($O(n|\Sigma|)$ time)
2. Compute $T_{\sigma} \otimes P_{\sigma}$ (for every symbol $\sigma$ in $\Sigma$) ($O(n|\Sigma| \log m)$ time)
3. For every $i$, compute,

$$\text{Ham}(i) = m - \sum_{\sigma \in \Sigma} (T_{\sigma} \otimes P_{\sigma})[i].$$

This takes $O(n|\Sigma| \log m)$ total time (and $O(n)$ space)
Hamming distance - considering symbols seperately

We saw how to find all matches with a single symbol in $O(n \log m)$ time

Let $\Sigma$ denote the set of alphabet symbols and $|\Sigma|$ be its size

Algorithm Summary

Construct $T_\sigma$ and $P_\sigma$ for every symbol $\sigma$ in $\Sigma$ \hspace{1cm} (O(n|\Sigma|) time)

Compute $T_\sigma \otimes P_\sigma$ (for every symbol $\sigma$ in $\Sigma$) \hspace{1cm} (O(n|\Sigma| \log m) time)

For every $i$, compute,

$$\text{Ham}(i) = m - \sum_{\sigma \in \Sigma} (T_\sigma \otimes P_\sigma)[i].$$ \hspace{1cm} (O(n|\Sigma|) time)

This takes $O(n|\Sigma| \log m)$ total time (and $O(n)$ space)

However, $|\Sigma|$ could be as big as $m$...
Hamming distance - considering symbols seperately

We saw how to find all matches with a single symbol in $O(n \log m)$ time

Let $\Sigma$ denote the set of alphabet symbols and $|\Sigma|$ be its size

Algorithm Summary

Construct $T_\sigma$ and $P_\sigma$ for every symbol $\sigma$ in $\Sigma$ \((O(n|\Sigma|) \text{ time})\)

Compute $T_\sigma \otimes P_\sigma$ (for every symbol $\sigma$ in $\Sigma$) \((O(n|\Sigma| \log m) \text{ time})\)

For every $i$, compute,

$$\text{Ham}(i) = m - \sum_{\sigma \in \Sigma} (T_\sigma \otimes P_\sigma)[i].$$  \((O(n|\Sigma|) \text{ time})\)

This takes $O(n|\Sigma| \log m)$ total time (and $O(n)$ space)

However, $|\Sigma|$ could be as big as $m$...

what should we do instead?
The frequent/infrequent symbols trick

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{m}$ times in $P$. 
The frequent/infrequent symbols trick

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{m}$ times in $P$.

$$P = \begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
| & | & | & | & | & | & | & |
m = 9 \\
\hline
a & b & b & a & c & a & d & b & d
\end{array}$$
The frequent/infrequent symbols trick

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{m}$ times in $P$.

\[P = a\ b\ b\ a\ c\ a\ d\ b\ d\]

$m = 9$

$a$ is frequent
The frequent/infrequent symbols trick

**Definition:** A symbol is *frequent* if it occurs at least \( \sqrt{m} \) times in \( P \).

\[
P = \begin{array}{cccccccc}
\text{①} & \text{②} & \text{③} & \text{④} & \text{⑤} & \text{⑥} & \text{⑦} & \text{⑧} \\
\text{a} & \text{b} & \text{b} & \text{a} & \text{c} & \text{a} & \text{d} & \text{b} & \text{d} \\
\end{array}
m = 9
\]

\( a \) is frequent, \( b \) is frequent
The frequent/infrequent symbols trick

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{m}$ times in $P$.

\[ P \]
\[ a \ b \ b \ a \ c \ a \ d \ b \ d \]

$a$ is frequent, $b$ is frequent, $c$ and $d$ are not frequent
The frequent/infrequent symbols trick

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{m}$ times in $P$.

\[ P = \begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
a & b & b & a & c & a & d & b & d \\
m = 9 \\
\end{array} \]

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are not frequent

**Step 1:** Count all matches involving frequent symbols.
The frequent/infrequent symbols trick

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{m}$ times in $P$.

\[
\begin{array}{cccccccccc}
\text{0} & \text{1} & \text{2} & \text{3} & \text{4} & \text{5} & \text{6} & \text{7} & \text{8} \\
\hline
m = 9 \\
P \quad a & b & b & a & c & a & d & b & d
\end{array}
\]

$a$ is frequent, $b$ is frequent, $c$ and $d$ are not frequent

**Step 1:** Count all matches involving frequent symbols.

Consider each frequent symbol separately in $O(n \log m)$ time (per symbol).
The frequent/infrequent symbols trick

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{m}$ times in $P$.

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
\end{array}
\]

\[
P = \begin{array}{cccccccc}
a & b & b & a & c & a & d & b & d \\
\end{array}
\]

\[
m = 9
\]

$a$ is frequent, $b$ is frequent
$c$ and $d$ are not frequent

**Step 1:** Count all matches involving frequent symbols.
Consider each frequent symbol separately in $O(n \log m)$ time (per symbol).

using cross-correlations
The frequent/infrequent symbols trick

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{m}$ times in $P$.

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
P & a & b & b & a & c & a & d & b & d \\
\end{array}
\]

$m = 9$

$a$ is frequent, $b$ is frequent
$c$ and $d$ are not frequent

**Step 1:** Count all matches involving frequent symbols.
Consider each frequent symbol separately in $O(n \log m)$ time (per symbol).

*using cross-correlations*

---

*How many frequent symbols can there be?*
The frequent/infrequent symbols trick

**Definition:** A symbol is \textit{frequent} if it occurs at least $\sqrt{m}$ times in $P$.

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
\end{array}
\]

\[
P = \begin{bmatrix}
a & b & b & a & c & a & d & b & d \\
\end{bmatrix}
\]

\[
a \text{ is frequent, } b \text{ is frequent} \\
c \text{ and } d \text{ are not frequent}
\]

**Step 1:** Count all matches involving frequent symbols.

Consider each frequent symbol separately in $O(n \log m)$ time (per symbol).

\textit{using cross-correlations}

---

**How many frequent symbols can there be?**

Assume that there at least $(\sqrt{m} + 1)$ freq. symbols
The frequent/infrequent symbols trick

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{m}$ times in $P$.

```
<table>
<thead>
<tr>
<th>0   1   2   3   4   5   6   7   8</th>
</tr>
</thead>
<tbody>
<tr>
<td>m = 9</td>
</tr>
<tr>
<td>-------------------------------------</td>
</tr>
<tr>
<td>P a b b a c a d b d</td>
</tr>
</tbody>
</table>
```

$a$ is frequent, $b$ is frequent
$c$ and $d$ are not frequent

**Step 1:** Count all matches involving frequent symbols.

Consider each frequent symbol separately in $O(n \log m)$ time (per symbol). *using cross-correlations*

*How many frequent symbols can there be?*

Assume that there at least $(\sqrt{m} + 1)$ freq. symbols each occurs at least $\sqrt{m}$ times...
The frequent/infrequent symbols trick

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{m}$ times in $P$.

\[
\begin{array}{cccccccc}
| & | & | & | & | & | & | & | \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
P & a & b & b & a & c & a & d & b & d \\
\end{array}
\]

$m = 9$

$a$ is frequent, $b$ is frequent
$c$ and $d$ are not frequent

**Step 1:** Count all matches involving frequent symbols.

Consider each frequent symbol separately in $O(n \log m)$ time (per symbol).

*using cross-correlations*

*How many frequent symbols can there be?*

Assume that there at least $(\sqrt{m} + 1)$ freq. symbols

each occurs at least $\sqrt{m}$ times... $(\sqrt{m} + 1)\sqrt{m} > m$
**The frequent/infrequent symbols trick**

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{m}$ times in $P$.

- **Step 1:** Count all matches involving frequent symbols.
  - Consider each frequent symbol separately in $O(n \log m)$ time (per symbol).

- How many frequent symbols can there be?
  - Assume that there at least $(\sqrt{m} + 1)$ freq. symbols
  - each occurs at least $\sqrt{m}$ times...

  $(\sqrt{m} + 1)\sqrt{m} > m$  
  Contradiction!
The frequent/infrequent symbols trick

**Definition:** A symbol is *frequent* if it occurs at least \( \sqrt{m} \) times in \( P \).

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
m = 9 \\
P & a & b & b & a & c & a & d & b & d \\
\end{array}
\]

- \( a \) is frequent, \( b \) is frequent
- \( c \) and \( d \) are not frequent

**Step 1:** Count all matches involving frequent symbols.

Consider each frequent symbol separately in \( O(n \log m) \) time (per symbol).

*using cross-correlations*

---

**How many frequent symbols can there be?**

Assume that there at least \( (\sqrt{m} + 1) \) freq. symbols

- each occurs at least \( \sqrt{m} \) times... \( (\sqrt{m} + 1)\sqrt{m} > m \) Contradiction!

*so there are at most \( \sqrt{m} \) frequent symbols*
The frequent/infrequent symbols trick

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{m}$ times in $P$.

![Symbol occurrences](image.png)

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are not frequent

**Step 1:** Count all matches involving frequent symbols.

Consider each frequent symbol separately in $O(n \log m)$ time (per symbol).

*using cross-correlations*

**How many frequent symbols can there be?**

Assume that there at least $(\sqrt{m} + 1)$ freq. symbols

- each occurs at least $\sqrt{m}$ times...
- $(\sqrt{m} + 1)\sqrt{m} > m$ Contradiction!

So there are at most $\sqrt{m}$ frequent symbols

*So Step 1 takes $O(n\sqrt{m} \log m)$ time.*
The infrequent/frequent symbols trick

Definition: A symbol is *infrequent* if it occurs fewer than \( \sqrt{m} \) times in \( P \).

\[
\begin{array}{cccccccccccc}
T & a & d & b & a & c & c & c & d & a & d & c & d & c & d & a & c \\
\hline
P & a & b & b & a & c & a & d & b & d \\
\hline
m = 9
\end{array}
\]

Every symbol is either frequent or infrequent

- \( a \) is frequent, \( b \) is frequent
- \( c \) and \( d \) are infrequent
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

| $T$ | a | d | b | a | c | c | c | d | a | d | c | d | c | d | a | c |
|-----|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $P$ | a | b | b | a | c | a | d | b | d | a | b | b | a | c | a |

Every symbol is either frequent or infrequent

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent.

$a$ is frequent, $b$ is frequent, $c$ and $d$ are infrequent.

| $T$ | a | d | b | a | c | c | c | d | a | d | c | d | c | d | a | c |
|-----|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $P$ | a | b | b | a | c | a | d | b | d |
| $A$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros.

Make a single pass through $T$...
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent.

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros.

Make a single pass through $T$...
The infrequent/frequent symbols trick

Definition: A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent.

- $a$ is frequent,
- $b$ is frequent,
- $c$ and $d$ are infrequent.

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros.

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than \( \sqrt{m} \) times in \( P \).

\[
\begin{array}{ccccccccccc}
T & a & d & b & a & c & c & c & d & a & d & c & d & c & d & a & c \\
\hline
P & a & b & b & a & c & a & d & b & d \\
A & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Every symbol is either frequent or infrequent

- \( a \) is frequent,
- \( b \) is frequent,
- \( c \) and \( d \) are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array \( A \) of length \( (n - m + 1) \) - which is initially all zeros

Make a single pass through \( T \)...

For each character \( T[k] \), (where \( 0 \leq k < n \))

If \( T[k] \) is infrequent...
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros.

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

$T = \text{d b a c c c d a d c d c d a c}$

$P = \text{a b b a c a d b d}$

$A = \text{0 0 0 0 0 0 0 0}$

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$ . . .

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent . . .

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one
The infrequent/frequent symbols trick

Definition: A symbol is *infrequent* if it occurs fewer than\( \sqrt{m} \) times in \( P \).

\[
\begin{array}{l}
T = d b a c c c d a d c d c d a c \\
P = a b b a c a d b b d \\
A = 0 0 0 0 0 0 0 0 0
\end{array}
\]

Every symbol is either frequent or infrequent

\( a \) is frequent, \( b \) is frequent
\( c \) and \( d \) are infrequent

Step 2: Count all matches involving infrequent symbols.

Construct an array \( A \) of length \((n - m + 1)\) - which is initially all zeros

\[
\begin{align*}
\text{Make a single pass through } T \ldots \\
\text{For each character } T[k], \text{ (where } 0 \leq k < n) \\
\text{If } T[k] \text{ is infrequent} \ldots \\
\text{For all } j \text{ such that } T[k] = P[j], \\
\text{Increase } A[k - j] \text{ by one} \\
\text{except when } (k - j) < 0
\end{align*}
\]
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent.  

- $a$ is frequent, $b$ is frequent  
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$ . . .

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent . . .

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one  

*except when* $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

![Diagram with symbols T and A]

Every symbol is either frequent or infrequent

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

*except when $(k - j) < 0$*
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

<table>
<thead>
<tr>
<th>$T$</th>
<th>$d$</th>
<th>$b$</th>
<th>$a$</th>
<th>$c$</th>
<th>$c$</th>
<th>$c$</th>
<th>$d$</th>
<th>$a$</th>
<th>$d$</th>
<th>$c$</th>
<th>$d$</th>
<th>$c$</th>
<th>$d$</th>
<th>$a$</th>
<th>$c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$</td>
<td>$a$</td>
<td>$b$</td>
<td>$b$</td>
<td>$a$</td>
<td>$c$</td>
<td>$a$</td>
<td>$d$</td>
<td>$b$</td>
<td>$d$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A$</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>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

- If $T[k]$ is infrequent...
  - For all $j$ such that $T[k] = P[j]$, Increase $A[k - j]$ by one
    - except when $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

### Step 2: Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...
For each character $T[k]$, (where $0 \leq k < n$)
If $T[k]$ is infrequent...
For all $j$ such that $T[k] = P[j]$,
Increase $A[k - j]$ by one

*except when $(k - j) < 0$*
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

|   | $a$ | $d$ | $b$ | $a$ | $c$ | $c$ | $c$ | $d$ | $a$ | $d$ | $c$ | $d$ | $a$ | $c$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | $\times$ | $\times$ | $a$ | $b$ | $\times$ | $a$ | $c$ | $c$ | $c$ | $d$ | $a$ | $d$ | $c$ | $d$ | $a$ | $c$ |
| $P$ | $a$ | $b$ | $b$ | $a$ | $\times$ | $c$ | $a$ | $d$ | $b$ | $d$ |
| $A$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$

Increase $A[k - j]$ by one

*except when $(k - j) < 0$*
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

\[
\begin{array}{cccccccccccc}
T & & & & & & & & & & & & \\
\times & d & \times & & a & c & c & c & d & a & d & c & d & c & d & a & c \\
\end{array}
\]

\[
\begin{array}{cccccccccccc}
P & & & & & & & & & & & & \\
a & b & b & a & c & a & d & b & d \\
\end{array}
\]

\[
\begin{array}{ccccccccccc}
A & & & & & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Every symbol is either frequent or infrequent

- $a$ is frequent
- $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

*except when* $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

$a$ is frequent, $b$ is frequent

$c$ and $d$ are infrequent

---

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

*except when $(k - j) < 0$*
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$…

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent…

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

*except when* $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent. $a$ is frequent, $b$ is frequent, $c$ and $d$ are infrequent.

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

except when $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent.

- $a$ is frequent,
- $b$ is frequent,
- $c$ and $d$ are infrequent.

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros.

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

- If $T[k]$ is infrequent...
  - For all $j$ such that $T[k] = P[j]$,
    - Increase $A[k - j]$ by one
    - *except when* $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than \( \sqrt{m} \) times in \( P \).

\[
k = 4
\]

\[
T \hspace{1cm} a \hspace{0.5cm} d \hspace{0.5cm} \times \hspace{0.5cm} \times \hspace{0.5cm} c \hspace{0.5cm} c \hspace{0.5cm} c \hspace{0.5cm} d \hspace{0.5cm} a \hspace{0.5cm} d \hspace{0.5cm} c \hspace{0.5cm} d \hspace{0.5cm} c \hspace{0.5cm} d \hspace{0.5cm} a \hspace{0.5cm} c
\]

\[
P \hspace{1cm} a \hspace{0.5cm} b \hspace{0.5cm} b \hspace{0.5cm} a \hspace{0.5cm} c \hspace{0.5cm} a \hspace{0.5cm} d \hspace{0.5cm} b \hspace{0.5cm} d
\]

\[
A \hspace{1cm} 1 \hspace{0.5cm} 0 \hspace{0.5cm} 0 \hspace{0.5cm} 0 \hspace{0.5cm} 0 \hspace{0.5cm} 0 \hspace{0.5cm} 0 \hspace{0.5cm} 0 \hspace{0.5cm} 0
\]

Every symbol is either frequent or infrequent

- \( a \) is frequent, \( b \) is frequent
- \( c \) and \( d \) are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array \( A \) of length \( (n - m + 1) \) - which is initially all zeros.

Make a single pass through \( T \)...

For each character \( T[k] \), (where \( 0 \leq k < n \))

If \( T[k] \) is infrequent...

For all \( j \) such that \( T[k] = P[j] \),

Increase \( A[k - j] \) by one

*except when \( (k - j) < 0 \).*
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent.

\(a\) is frequent, \(b\) is frequent, \(c\) and \(d\) are infrequent.

<table>
<thead>
<tr>
<th></th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(T)</td>
<td>(\times)</td>
<td>(\times)</td>
<td>(\times)</td>
<td>(c)</td>
</tr>
<tr>
<td>(P)</td>
<td>(a)</td>
<td>(b)</td>
<td>(b)</td>
<td>(a)</td>
</tr>
<tr>
<td>(A)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Step 2:** Count all matches involving infrequent symbols.

Construct an array \(A\) of length \((n - m + 1)\) - which is initially all zeros

Make a single pass through \(T\)...

For each character \(T[k]\), (where \(0 \leq k < n\))

If \(T[k]\) is infrequent...

For all \(j\) such that \(T[k] = P[j]\),

Increase \(A[k - j]\) by one

*except when \((k - j) < 0\)*
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than \( \sqrt{m} \) times in \( P \).

Every symbol is either frequent or infrequent

- \( a \) is frequent, \( b \) is frequent
- \( c \) and \( d \) are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array \( A \) of length \((n - m + 1)\) - which is initially all zeros

Make a single pass through \( T \)

For each character \( T[k] \), (where \( 0 \leq k < n \))

If \( T[k] \) is infrequent...

For all \( j \) such that \( T[k] = P[j] \),

Increase \( A[k - j] \) by one

*except when \((k - j) < 0)*
**The infrequent/frequent symbols trick**

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$…

For each character $T[k]$, (where $0 \leq k < n$)

- If $T[k]$ is infrequent…
  - For all $j$ such that $T[k] = P[j]$,
    - Increase $A[k - j]$ by one
  - except when $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than \(\sqrt{m}\) times in \(P\).

\[ k = 5 \]

Every symbol is either frequent or infrequent

- \(a\) is frequent
- \(b\) is frequent
- \(c\) and \(d\) are infrequent

<table>
<thead>
<tr>
<th>(T)</th>
<th>(a)</th>
<th>(d)</th>
<th>(x)</th>
<th>(c)</th>
<th>(c)</th>
<th>(c)</th>
<th>(d)</th>
<th>(a)</th>
<th>(d)</th>
<th>(c)</th>
<th>(d)</th>
<th>(a)</th>
<th>(c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P)</td>
<td>(a)</td>
<td>(b)</td>
<td>(b)</td>
<td>(a)</td>
<td>(c)</td>
<td>(a)</td>
<td>(d)</td>
<td>(b)</td>
<td>(d)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(A)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Step 2:** Count all matches involving infrequent symbols.

Construct an array \(A\) of length \((n - m + 1)\) - which is initially all zeros

Make a single pass through \(T\)...

For each character \(T[k]\), (where \(0 \leq k < n\))

If \(T[k]\) is infrequent...

For all \(j\) such that \(T[k] = P[j]\),

Increase \(A[k - j]\) by one

except when \((k - j) < 0\)
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

$k = 5$

Every symbol is either frequent or infrequent

$a$ is frequent, $b$ is frequent
$c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,
Increase $A[k - j]$ by one

*except when* $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than \( \sqrt{m} \) times in \( P \).

Every symbol is either frequent or infrequent

\( a \) is frequent, \( b \) is frequent

\( c \) and \( d \) are infrequent

---

**Step 2:** Count all matches involving infrequent symbols.

Construct an array \( A \) of length \( (n - m + 1) \) - which is initially all zeros

Make a single pass through \( T \)...

For each character \( T[k] \), (where \( 0 \leq k < n \))

If \( T[k] \) is infrequent...

For all \( j \) such that \( T[k] = P[j] \),

Increase \( A[k - j] \) by one

except when \( (k - j) < 0 \)
The infrequent/frequent symbols trick

Definition: A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

$a$ is frequent, $b$ is frequent
$c$ and $d$ are infrequent

Step 2: Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j],$

Increase $A[k - j]$ by one

except when $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than \( \sqrt{m} \) times in \( P \).

Every symbol is either frequent or infrequent

\( a \) is frequent, \( b \) is frequent, \( c \) and \( d \) are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array \( A \) of length \( (n - m + 1) \) - which is initially all zeros

Make a single pass through \( T \)...

For each character \( T[k] \), (where \( 0 \leq k < n \))

If \( T[k] \) is infrequent...

For all \( j \) such that \( T[k] = P[j] \),

Increase \( A[k - j] \) by one

except when \( (k - j) < 0 \)
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

```
T   a d x c c c d a d c d c d a c
P   a b b a c a d b d
A   1 1 0 0 0 0 0 0
```

Every symbol is either frequent or infrequent

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$

Increase $A[k - j]$ by one

except when $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than \( \sqrt{m} \) times in \( P \).

Every symbol is either frequent or infrequent

\( a \) is frequent, \( b \) is frequent
\( c \) and \( d \) are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array \( A \) of length \((n - m + 1)\) - which is initially all zeros

Make a single pass through \( T \)...

For each character \( T[k] \), (where \( 0 \leq k < n \))

If \( T[k] \) is infrequent...

For all \( j \) such that \( T[k] = P[j] \),

Increase \( A[k - j] \) by one

except when \((k - j) < 0\)
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

Step 2: Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

- If $T[k]$ is infrequent...
  - For all $j$ such that $T[k] = P[j]$,
    - Increase $A[k - j]$ by one
  - except when $(k - j) < 0$
**The infrequent/frequent symbols trick**

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent.

$a$ is frequent, $b$ is frequent, $c$ and $d$ are infrequent.

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros.

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$, 

Increase $A[k - j]$ by one 

*except when* $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

$a$ is frequent, $b$ is frequent,
$c$ and $d$ are infrequent

Step 2: Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros.

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,
Increase $A[k - j]$ by one

*except when* $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent:
- $a$ is frequent,
- $b$ is frequent,
- $c$ and $d$ are infrequent.

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros.

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one except when $(k - j) < 0$.
**The infrequent/frequent symbols trick**

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent.

- $a$ is frequent,
- $b$ is frequent,
- $c$ and $d$ are infrequent.

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros.

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

- If $T[k]$ is infrequent...

  For all $j$ such that $T[k] = P[j]$,
  
  Increase $A[k - j]$ by one
  
  *except when* $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than \( \sqrt{m} \) times in \( P \).

Every symbol is either frequent or infrequent. 

\( a \) is frequent, \( b \) is frequent 
\( c \) and \( d \) are infrequent

Step 2: Count all matches involving infrequent symbols.

Construct an array \( A \) of length \( (n - m + 1) \) - which is initially all zeros

Make a single pass through \( T \)...

For each character \( T[k] \), (where \( 0 \leq k < n \))

If \( T[k] \) is infrequent...

For all \( j \) such that \( T[k] = P[j] \),

Increase \( A[k - j] \) by one

except when \( (k - j) < 0 \)
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

$a$ is frequent, $b$ is frequent
$c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$…

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent…

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

*except when $(k - j) < 0$*
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

$a$ is frequent, $b$ is frequent
$c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

except when $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than \( \sqrt{m} \) times in \( P \).

Every symbol is either frequent or infrequent.

\( a \) is frequent, \( b \) is frequent, \( c \) and \( d \) are infrequent.

**Step 2:** Count all matches involving infrequent symbols.

Construct an array \( A \) of length \( (n - m + 1) \) - which is initially all zeros.

Make a single pass through \( T \)...

For each character \( T[k] \), (where \( 0 \leq k < n \))

If \( T[k] \) is infrequent...

For all \( j \) such that \( T[k] = P[j] \),

Increase \( A[k - j] \) by one

except when \( (k - j) < 0 \)
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

except when $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

$a$ is frequent, $b$ is frequent
$c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j],$

Increase $A[k - j]$ by one

except when $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$, increase $A[k - j]$ by one

*except when* $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

$a$ is frequent, $b$ is frequent
$c$ and $d$ are infrequent

Step 2: Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

except when $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

\(k = 7\)

\[
T \begin{bmatrix}
\text{a} & \text{d} & \text{a} & \text{c} & \text{c} & \text{c} & \text{c} & \text{d} & \text{d} & \text{a} & \text{c}
\end{bmatrix}
\]

\[
P \begin{bmatrix}
\text{a} & \text{b} & \text{b} & \text{a} & \text{c} & \text{a} & \text{d} & \text{b} & \text{d}
\end{bmatrix}
\]

\[
A \begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$, 
Increase $A[k - j]$ by one 

except when $(k - j) < 0$

\(a\) is frequent, \(b\) is frequent 
\(c\) and \(d\) are infrequent
**The infrequent/frequent symbols trick**

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

- Make a single pass through $T$...
- For each character $T[k]$, (where $0 \leq k < n$)
  - If $T[k]$ is infrequent...
    - For all $j$ such that $T[k] = P[j]$,
      - Increase $A[k - j]$ by one

*except when $(k - j) < 0$*
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

$a$ is frequent, $b$ is frequent

c and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$…

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent…

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

*except when* $(k - j) < 0$
**The infrequent/frequent symbols trick**

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent. 

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

- If $T[k]$ is infrequent...
  - For all $j$ such that $T[k] = P[j]$,
    - Increase $A[k - j]$ by one

*except when $(k - j) < 0$*
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent.

\[ a \text{ is frequent, } b \text{ is frequent, } c \text{ and } d \text{ are infrequent} \]

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros.

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j],$

Increase $A[k - j]$ by one

except when $(k - j) < 0 $
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

- $a$ is frequent,
- $b$ is frequent,
- $c$ and $d$ are infrequent.

### Step 2: Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros.

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

- If $T[k]$ is infrequent...
  - For all $j$ such that $T[k] = P[j]$,
  - Increase $A[k - j]$ by one
    except when $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

* $a$ is frequent, $b$ is frequent
* $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

except when $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

<table>
<thead>
<tr>
<th>$T$</th>
<th>$d$</th>
<th>$c$</th>
<th>$c$</th>
<th>$c$</th>
<th>$d$</th>
<th>$c$</th>
<th>$d$</th>
<th>$c$</th>
<th>$d$</th>
<th>$a$</th>
<th>$c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$</td>
<td>$a$</td>
<td>$b$</td>
<td>$b$</td>
<td>$a$</td>
<td>$c$</td>
<td>$a$</td>
<td>$d$</td>
<td>$b$</td>
<td>$d$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A$</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Every symbol is either frequent or infrequent

$a$ is frequent, $b$ is frequent
$c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$ . . .

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent . . .

For all $j$ such that $T[k] = P[j]$, Increase $A[k - j]$ by one

except when $(k - j) < 0$
**The infrequent/frequent symbols trick**

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

$a$ is frequent, $b$ is frequent

$c$ and $d$ are infrequent

$T$: 

```
| a | d | b | x | a | c | c | c | d | x | d | c | d | c | d | a | c |
```

$P$: 

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

$A$: 

```
| 1 | 2 | 1 | 0 | 0 | 0 | 0 | 0 |
```

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

*except when $(k - j) < 0$*
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than \(\sqrt{m}\) times in \(P\).

Every symbol is either frequent or infrequent

\(a\) is frequent, \(b\) is frequent
\(c\) and \(d\) are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array \(A\) of length \((n - m + 1)\) - which is initially all zeros

Make a single pass through \(T\)...

For each character \(T[k]\), (where \(0 \leq k < n\))

If \(T[k]\) is infrequent...

For all \(j\) such that \(T[k] = P[j]\),

Increase \(A[k - j]\) by one

except when \((k - j) < 0\)
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent.

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

except when $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

$a$ is frequent, $b$ is frequent
$c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

except when $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent.

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros.

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

- If $T[k]$ is infrequent...
  - For all $j$ such that $T[k] = P[j]$, increase $A[k - j]$ by one
  - except when $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

$a$ is frequent, $b$ is frequent
$c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

except when $(k - j) < 0
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

![Diagram showing symbols and matching counts]

Every symbol is either frequent or infrequent

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

- Make a single pass through $T$...
- For each character $T[k]$, (where $0 \leq k < n$)
  - If $T[k]$ is infrequent...
    - For all $j$ such that $T[k] = P[j]$, Increase $A[k - j]$ by one
      - *except when* $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

\[\begin{align*}
T &= \text{[\textcolor{red}{a, d} \times \textcolor{green}{b} \times \textcolor{green}{c} \textcolor{green}{c} \textcolor{green}{c} \textcolor{green}{d} \times \textcolor{red}{d} \text{ d c d d a c}]} \\
A &= [1 \ 3 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0] \\
P &= [\textcolor{green}{a} \ b \ b \ a \ \textcolor{red}{c} \ \textcolor{green}{a} \ d \ b \ d]
\end{align*}\]

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

except when $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

$$T = \underline{a} \; d \; \underline{b} \; a \; c \; c \; c \; d \; \underline{a} \; d \; c \; d \; c \; d \; a \; c$$

Every symbol is either frequent or infrequent

$\underline{a}$ is frequent, $\underline{b}$ is frequent
$c$ and $d$ are infrequent

$$P = a \; b \; b \; a \; c \; a \; d \; b \; d$$

$$A = \begin{bmatrix} 1 & 3 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

except when $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

$a$ is frequent, $b$ is frequent
$c$ and $d$ are infrequent

Step 2: Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

except when $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than \( \sqrt{m} \) times in \( P \).

\[
A = \begin{bmatrix} 1 & 3 & 1 & 2 & 0 & 2 & 1 & 1 \end{bmatrix}
\]

\[
T = \begin{bmatrix} a & d & b & \times & a & c & c & c & d & x & d & c & d & c & d & a & c \end{bmatrix}
\]

\[
P = \begin{bmatrix} a & b & b & a & c & a & d & b & d \end{bmatrix}
\]

\[
A[k - j] \text{ except when } (k - j) < 0
\]

Every symbol is either frequent or infrequent.

\( a \) is frequent, \( b \) is frequent,
\( c \) and \( d \) are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array \( A \) of length \((n - m + 1)\) - which is initially all zeros

Make a single pass through \( T \)...

For each character \( T[k] \), (where \( 0 \leq k < n \))

If \( T[k] \) is infrequent...

For all \( j \) such that \( T[k] = P[j] \),

Increase \( A[k - j] \) by one

What is \( A[i] \)?
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

- $a$ is frequent
- $b$ is frequent
- $c$ and $d$ are infrequent

---

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$, Increase $A[k - j]$ by one

*except when $(k - j) < 0$*
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

$a$ is frequent, $b$ is frequent

$c$ and $d$ are infrequent

Step 2: Count all matches involving infrequent symbols

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$, $A[k - j]$ by one

*except when* $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than \( \sqrt{m} \) times in \( P \).

Every symbol is either frequent or infrequent

\[
T \quad a \quad d \quad b \quad a \quad c \quad c \quad c \quad d \quad a \quad d \quad c \quad d \quad c \quad d \quad a \quad c
\]

\[
P \quad a \quad b \quad b \quad a \quad c \quad a \quad d \quad b \quad d
\]

\[
A \quad 1 \quad 3 \quad 1 \quad 2 \quad 0 \quad 2 \quad 1 \quad 1
\]

**Step 2:** Count all matches involving infrequent symbols.

Construct an array \( A \) of length \( (n - m + 1) \) - which is initially all zeros

Make a single pass through \( T \)...

For each character \( T[k] \), (where \( 0 \leq k < n \))

If \( T[k] \) is infrequent...

For all \( j \) such that \( T[k] = P[j] \),

Increase \( A[k - j] \) by one

except when \( (k - j) < 0 \)
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

$a$ is frequent, $b$ is frequent
$c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

except when $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

- $a$ is frequent, $b$ is frequent
- $c$ and $d$ are infrequent

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$, increase $A[k - j]$ by one

*except when* $(k - j) < 0$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent

$a$ is frequent, $b$ is frequent
$c$ and $d$ are infrequent

Step 2: Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

*except when* $(k - j) < 0$

How quick is Step 2?

$O(n)$ time
The infrequent/frequent symbols trick

Definition: A symbol is \textit{infrequent} if it occurs fewer than $\sqrt{m}$ times in $P$.

Step 2: Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros.

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$, 

Increase $A[k - j]$ by one

except when $(k - j) < 0$

Every symbol is either frequent or infrequent

$a$ is frequent, $b$ is frequent 
$c$ and $d$ are infrequent

How quick is Step 2?

$O(n)$ time

store a list for each infrequent symbol

each list has length less than $\sqrt{m}$
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent.

How quick is Step 2?

$O(n)$ time

- Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros
- Make a single pass through $T$...
  - For each character $T[k]$, (where $0 \leq k < n$)
    - If $T[k]$ is infrequent...
      - For all $j$ such that $T[k] = P[j]$, increase $A[k - j]$ by one
        - except when $(k - j) < 0$

- Each list has length less than $\sqrt{m}$
- Store a list for each infrequent symbol
- $O(n\sqrt{m})$ time
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

Every symbol is either frequent or infrequent.

$T$:
```
- a d x a c c c d x d c d c d a c
```

$P$:
```
a b b a c a d b d
```

$A$:
```
1 3 1 2 0 2 1 1
```

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros.

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$,

Increase $A[k - j]$ by one

*except when $(k - j) < 0$*
The infrequent/frequent symbols trick

**Definition:** A symbol is *infrequent* if it occurs fewer than $\sqrt{m}$ times in $P$.

\[
\begin{array}{cccccccc}
T & a & d & b & a & c & c & c & d & d & c & c & d & a & c \\
A & 1 & 3 & 1 & 2 & 0 & 2 & 1 & 1
\end{array}
\]

Every symbol is either frequent or infrequent:
- $a$ is frequent,
- $b$ is frequent,
- $c$ and $d$ are infrequent.

**Step 2:** Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which is initially all zeros.

Make a single pass through $T$...

For each character $T[k]$, (where $0 \leq k < n$)

If $T[k]$ is infrequent...

For all $j$ such that $T[k] = P[j]$, 

Increase $A[k - j]$ by one

except when $(k - j) < 0$

$O(n\sqrt{m})$ total time
Pattern matching with mismatches: putting it all together

Algorithm summary
Algorithm summary

**Step 0:** Classify each symbol as frequent or infrequent ($O(m \log m)$ time)
Pattern matching with mismatches: putting it all together

Algorithm summary

Step 0: Classify each symbol as frequent or infrequent ($O(m \log m)$ time)

Step 1: Count all matches involving frequent symbols. ($O(n \sqrt{m} \log m)$ time)
Pattern matching with mismatches: putting it all together

Algorithm summary

**Step 0:** Classify each symbol as frequent or infrequent \((O(m \log m)\) time)

**Step 1:** Count all matches involving frequent symbols. \((O(n\sqrt{m} \log m)\) time)

**Step 2:** Count all matches involving infrequent symbols. \((O(n\sqrt{m})\) time)
Pattern matching with mismatches: putting it all together

Algorithm summary

**Step 0:** Classify each symbol as frequent or infrequent (\(O(m \log m)\) time)

**Step 1:** Count all matches involving frequent symbols. (\(O(n\sqrt{m} \log m)\) time)

**Step 2:** Count all matches involving infrequent symbols. (\(O(n\sqrt{m})\) time)
Pattern matching with mismatches: putting it all together

Algorithm summary

Step 0: Classify each symbol as frequent or infrequent ($O(m \log m)$ time)

Step 1: Count all matches involving frequent symbols. ($O(n \sqrt{m} \log m)$ time)

Step 2: Count all matches involving infrequent symbols. ($O(n \sqrt{m})$ time)

at any alignment
the number of mismatches is just $m$ minus the total number of matches
Algorithm summary

**Step 0:** Classify each symbol as frequent or infrequent ($O(m \log m)$ time)

**Step 1:** Count all matches involving frequent symbols. ($O(n\sqrt{m} \log m)$ time)

**Step 2:** Count all matches involving infrequent symbols. ($O(n\sqrt{m})$ time)

at any alignment
the number of mismatches is just $m$ minus the total number of matches

Overall, we obtain a time complexity of $O(n\sqrt{m} \log m)$. 
Pattern matching with mismatches: putting it all together

Algorithm summary

**Step 0:** Classify each symbol as frequent or infrequent ($O(m \log m)$ time)

**Step 1:** Count all matches involving frequent symbols. ($O(n\sqrt{m \log m})$ time)

**Step 2:** Count all matches involving infrequent symbols. ($O(n\sqrt{m})$ time)

at any alignment

the number of mismatches is just $m$ minus the total number of matches

Overall, we obtain a time complexity of $O(n\sqrt{m \log m})$.

Notice that Step 1 takes longer than Step 2...
Pattern matching with mismatches: putting it all together

Algorithm summary

**Step 0:** Classify each symbol as frequent or infrequent \((O(m \log m)\) time)

**Step 1:** Count all matches involving frequent symbols. \((O(n \sqrt{m \log m})\) time)

**Step 2:** Count all matches involving infrequent symbols. \((O(n \sqrt{m \log m})\) time)

*at any alignment*

the number of mismatches is just \(m\) minus the total number of matches

**Overall,** we obtain a time complexity of \(O(n \sqrt{m \log m})\).

(by changing the definition of *frequent* to be at least \(\sqrt{m \log m}\) occurrences.)
Pattern matching with few mismatches ($k$-mismatch)

**Input** A text string $T$ (length $n$), a pattern string $P$ (length $m$) and a positive integer $k$

```
T
| a | b | c | d | a | b | a | a | d | a | a | a | a |
```
```
P
| a | b | d | a |
```

**Goal:** For all $i$, output,

$$\text{Ham}_k(i) = \begin{cases} 
\text{Ham}(i) & \text{if Ham}(i) \leq k \\
X & \text{if Ham}(i) > k 
\end{cases}$$

Output the number of mismatches... unless it's more than $k$
(we interpret the output $X$ to mean “too many mismatches”)
Pattern matching with few mismatches ($k$-mismatch)

**Input** A text string $T$ (length $n$), a pattern string $P$ (length $m$) and a positive integer $k$

$$
\begin{array}{cccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12
\end{array}
\begin{array}{cccccccccccc}
T & | & a & b & c & d & a & b & a & a & d & a & a & a & a & a
\end{array}
\begin{array}{cccccccccccc}
P & \boxed{a & b & d & a}
\end{array}
\begin{array}{cccccccccccc}
& n & m
\end{array}

$k = 2$

**Goal:** For all $i$, output,

$$
Ham_k(i) = \begin{cases} 
\text{Ham}(i) & \text{if } \text{Ham}(i) \leq k \\
X & \text{if } \text{Ham}(i) > k 
\end{cases}
$$

Output the number of mismatches... unless it's more than $k$

(we interpret the output $X$ to mean “too many mismatches”)
Pattern matching with few mismatches ($k$-mismatch)

**Input** A text string $T$ (length $n$), a pattern string $P$ (length $m$) and a positive integer $k$

<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>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>a</td>
<td>d</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>n</td>
</tr>
<tr>
<td>$P$</td>
<td>a</td>
<td>b</td>
<td>d</td>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>m</td>
</tr>
</tbody>
</table>

$k = 2$

$	ext{Ham}_k(4) = 1$

**Goal:** For all $i$, output,

$$\text{Ham}_k(i) = \begin{cases} \text{Ham}(i) & \text{if Ham}(i) \leq k \\ X & \text{if Ham}(i) > k \end{cases}$$

*Output the number of mismatches... unless it's more than $k$*

*(we interpret the output $X$ to mean “too many mismatches”)*
Pattern matching with few mismatches ($k$-mismatch)

**Input** A text string $T$ (length $n$), a pattern string $P$ (length $m$) and a positive integer $k$

```
       0  1  2  3  4  5  6  7  8  9 10 11 12
      ___  ___  ___  ___  ___  ___  ___  ___  ___  ___  ___  ___  ___
     n
 T      a b c d a b a a d a a a a
 P      a b d a
```

$k = 2$

**Goal:** For all $i$, output,

$$\text{Ham}_k(i) = \begin{cases} 
\text{Ham}(i) & \text{if Ham}(i) \leq k \\
X & \text{if Ham}(i) > k
\end{cases}$$

Output the number of mismatches... unless it's more than $k$

(we interpret the output $X$ to mean “too many mismatches”)
Pattern matching with few mismatches ($k$-mismatch)

**Input** A text string $T$ (length $n$), a pattern string $P$ (length $m$) and a positive integer $k$

**Goal:** For all $i$, output,

$$
\text{Ham}_k(i) = \begin{cases} 
\text{Ham}(i) & \text{if Ham}(i) \leq k \\
X & \text{if Ham}(i) > k
\end{cases}
$$

Output the number of mismatches... unless it's more than $k$

(we interpret the output $X$ to mean “too many mismatches”)

**Example:**

Input:

<table>
<thead>
<tr>
<th>$T$</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>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>a</td>
<td>d</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$P$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a</td>
<td>b</td>
<td>d</td>
<td>a</td>
<td></td>
</tr>
</tbody>
</table>

$Ham_2(6) = 1$
Pattern matching with few mismatches ($k$-mismatch)

**Input** A text string $T$ (length $n$), a pattern string $P$ (length $m$) and a positive integer $k$

\[
\begin{array}{cccccccccccc}
\hline
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
\hline
T & a & b & c & d & a & b & a & a & d & a & a & a \\
\hline
P & a & b & d & a \\
\hline
\end{array}
\]

$k = 2$

\[
\text{Ham}_k(7) = 2
\]

**Goal:** For all $i$, output,

\[
\text{Ham}_k(i) = \begin{cases} 
\text{Ham}(i) & \text{if } \text{Ham}(i) \leq k \\
X & \text{if } \text{Ham}(i) > k 
\end{cases}
\]

*Output the number of mismatches . . . unless it's more than $k$ (we interpret the output $X$ to mean "too many mismatches")*
Pattern matching with few mismatches ($k$-mismatch)

**Input** A text string $T$ (length $n$), a pattern string $P$ (length $m$) and a positive integer $k$

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></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>11</td>
<td>12</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>$T$</td>
<td>$a$</td>
<td>$b$</td>
<td>$c$</td>
<td>$d$</td>
<td>$a$</td>
<td>$b$</td>
<td>$a$</td>
<td>$a$</td>
<td>$d$</td>
<td>$a$</td>
<td>$a$</td>
<td>$a$</td>
</tr>
</tbody>
</table>

$P$

$\begin{array}{ccc}
\color{red}{a} & \color{red}{b} & \color{red}{d} & \color{red}{a} \\
\end{array}$

$k = 2$

Ham$_k(8) = X$

**Goal:** For all $i$, output,

$$
\text{Ham}_k(i) = \begin{cases} 
\text{Ham}(i) & \text{if } \text{Ham}(i) \leq k \\
X & \text{if } \text{Ham}(i) > k 
\end{cases}
$$

*Output the number of mismatches... unless it's more than $k$*

*(we interpret the output $X$ to mean “too many mismatches”)*
Pattern matching with few mismatches ($k$-mismatch)

**Input** A text string $T$ (length $n$), a pattern string $P$ (length $m$) and a positive integer $k$

**Goal:** For all $i$, output,

$$\text{Ham}_k(i) = \begin{cases} \text{Ham}(i) & \text{if Ham}(i) \leq k \\ X & \text{if Ham}(i) > k \end{cases}$$

Output the number of mismatches... unless it's more than $k$
(we interpret the output $X$ to mean “too many mismatches”)

- We could use the $O(n\sqrt{m\log m})$ time algorithm for Hamming distance...
Pattern matching with few mismatches (\(k\)-mismatch)

**Input** A text string \(T\) (length \(n\)), a pattern string \(P\) (length \(m\)) and a positive integer \(k\)

\[
\begin{array}{cccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
\hline
T & a & b & c & d & a & b & a & a & d & a & a & a \\
\hline
P & a & b & d & a \\
\hline
\end{array}
\]

\(k = 2\)

\(\text{Ham}_k(8) = X\)

**Goal:** For all \(i\), output,

\[
\text{Ham}_k(i) = \begin{cases} 
\text{Ham}(i) & \text{if Ham}(i) \leq k \\
X & \text{if Ham}(i) > k 
\end{cases}
\]

*Output the number of mismatches... unless it’s more than \(k\)*

(we interpret the output \(X\) to mean “too many mismatches”)

- We could use the \(O(n\sqrt{m\log m})\) time algorithm for Hamming distance...

  *but when \(k\) is small we can do much better*
LCP - the Longest Common Prefix

For any pair of locations $i$ in $T$ and $j$ in $P$, $\text{LCP}(i, j)$ is the largest $d$ such that

$$T[i \ldots i + d - 1] = P[j \ldots j + d - 1]$$

*it's the furthest you can go before hitting a mismatch*
LCP - the Longest Common Prefix

For any pair of locations \( i \) in \( T \) and \( j \) in \( P \), \( \text{LCP}(i, j) \) is the largest \( d \) such that

\[
T[i \ldots i + d - 1] = P[j \ldots j + d - 1]
\]

\emph{it's the furthest you can go before hitting a mismatch}
LCP - the Longest Common Prefix

For any pair of locations $i$ in $T$ and $j$ in $P$, $\text{LCP}(i, j)$ is the largest $d$ such that

$$T[i \ldots i + d - 1] = P[j \ldots j + d - 1]$$

*it's the furthest you can go before hitting a mismatch*
LCP - the Longest Common Prefix

For any pair of locations $i$ in $T$ and $j$ in $P$, $\text{LCP}(i, j)$ is the largest $d$ such that

$$T[i \ldots i + d - 1] = P[j \ldots j + d - 1]$$

it's the furthest you can go before hitting a mismatch
LCP - the Longest Common Prefix

For any pair of locations $i$ in $T$ and $j$ in $P$, $\text{LCP}(i, j)$ is the largest $d$ such that

$$T[i \ldots i + d - 1] = P[j \ldots j + d - 1]$$

*it's the furthest you can go before hitting a mismatch*
For any pair of locations $i$ in $T$ and $j$ in $P$, $LCP(i, j)$ is the largest $d$ such that

$$T[i \ldots i + d - 1] = P[j \ldots j + d - 1]$$


doesn’t change the furthest you can go before hitting a mismatch

**WARNING** This is the definition of LCP between two strings $P$ and $T$
LCP - the Longest Common Prefix

For any pair of locations $i$ in $T$ and $j$ in $P$, LCP($i, j$) is the largest $d$ such that

$$T[i \ldots i + d - 1] = P[j \ldots j + d - 1]$$

it's the furthest you can go before hitting a mismatch

**WARNING** This is the definition of LCP between two strings $P$ and $T$ (previously we saw LCP on a single string)
For any pair of locations \( i \) in \( T \) and \( j \) in \( P \), \( \text{LCP}(i, j) \) is the largest \( d \) such that

\[
T[i \ldots i + d - 1] = P[j \ldots j + d - 1]
\]

it's the furthest you can go before hitting a mismatch

**WARNING** This is the definition of LCP between two strings \( P \) and \( T \)

(previously we saw LCP on a single string)

How can we preprocess \( P \) and \( T \) for \( O(1) \) time LCP queries?
LCP - the Longest Common Prefix

For any pair of locations $i$ in $T$ and $j$ in $P$, $\text{LCP}(i, j)$ is the largest $d$ such that

\[ T[i \ldots i + d - 1] = P[j \ldots j + d - 1] \]

*it's the furthest you can go before hitting a mismatch*

**WARNING** This is the definition of LCP between two strings $P$ and $T$ 
(\textit{previously we saw LCP on a single string})

How can we preprocess $P$ and $T$ for $O(1)$ time LCP queries?
LCP - the Longest Common Prefix

For any pair of locations $i$ in $T$ and $j$ in $P$, LCP$(i, j)$ is the largest $d$ such that

$$T[i \ldots i + d - 1] = P[j \ldots j + d - 1]$$

it's the furthest you can go before hitting a mismatch

**WARNING** This is the definition of LCP between two strings $P$ and $T$ (previously we saw LCP on a single string)

How can we preprocess $P$ and $T$ for $O(1)$ time LCP queries?

Preprocess $T'$ for (single string) LCPs in $O(n + m)$ time

and $O(n + m)$ space (using the method seen previously)
LCP - the Longest Common Prefix

For any pair of locations $i$ in $T$ and $j$ in $P$, $\text{LCP}(i, j)$ is the largest $d$ such that

$$T[i \ldots i + d - 1] = P[j \ldots j + d - 1]$$

**it's the furthest you can go before hitting a mismatch**

**WARNING** This is the definition of LCP between two strings $P$ and $T$

**(previously we saw LCP on a single string)**

How can we preprocess $P$ and $T$ for $O(1)$ time LCP queries?

Preprocess $T'$ for (single string) LCPs in $O(n + m)$ time

and $O(n + m)$ space (using the method seen previously)

Queries take $O(1)$ time
LCP - the Longest Common Prefix

For any pair of locations $i$ in $T$ and $j$ in $P$, $\text{LCP}(i, j)$ is the largest $d$ such that

$$T[i \ldots i + d - 1] = P[j \ldots j + d - 1]$$

*it's the furthest you can go before hitting a mismatch*

**WARNING** This is the definition of LCP between two strings $P$ and $T$

*(previously we saw LCP on a single string)*
For any pair of locations $i$ in $T$ and $j$ in $P$, LCP($i, j$) is the largest $d$ such that

$$T[i \ldots i + d - 1] = P[j \ldots j + d - 1]$$

*it's the furthest you can go before hitting a mismatch*

**WARNING** This is the definition of LCP between two strings $P$ and $T$  
(Previously we saw LCP on a single string)

- We can process $P$ and $T$ for LCP queries in $O(n)$ time and $O(n)$ space
LCP - the Longest Common Prefix

For any pair of locations \(i\) in \(T\) and \(j\) in \(P\), \(\text{LCP}(i, j)\) is the largest \(d\) such that

\[
T[i \ldots i + d - 1] = P[j \ldots j + d - 1]
\]

*it's the furthest you can go before hitting a mismatch*

**WARNING** This is the definition of LCP between two strings \(P\) and \(T\)

*(previously we saw LCP on a single string)*

- We can process \(P\) and \(T\) for LCP queries in \(O(n)\) time and \(O(n)\) space

  *we'll saw how to do this in a previous lecture*
LCP - the Longest Common Prefix

For any pair of locations $i$ in $T$ and $j$ in $P$, LCP($i, j$) is the largest $d$ such that

$$T[i \ldots i + d - 1] = P[j \ldots j + d - 1]$$

it's the furthest you can go before hitting a mismatch

**WARNING** This is the definition of LCP between two strings $P$ and $T$

*(previously we saw LCP on a single string)*

- We can process $P$ and $T$ for LCP queries in $O(n)$ time and $O(n)$ space
  
  *we’ll saw how to do this in a previous lecture*

- Each query takes $O(1)$ time
LCP - the Longest Common Prefix

For any pair of locations $i$ in $T$ and $j$ in $P$, $\text{LCP}(i, j)$ is the largest $d$ such that

$$T[i \ldots i + d - 1] = P[j \ldots j + d - 1]$$

*it's the furthest you can go before hitting a mismatch*

**WARNING** This is the definition of LCP between two strings $P$ and $T$

*(previously we saw LCP on a single string)*

- We can process $P$ and $T$ for LCP queries in $O(n)$ time and $O(n)$ space
  
  *we’ll saw how to do this in a previous lecture*

- Each query takes $O(1)$ time

*We can use LCP queries to solve $k$-mismatch...*
$\kappa$-mismatch using LCP queries

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

$P$: $b\ c\ b\ c\ a\ a\ b$

$n$
$k$-mismatch using LCP queries

$T$: 
\[
\begin{array}{cccccccc}
 a & b & c & b & a & b & a & b \\
 c & a & b & a & b & a & a & a \\
\end{array}
\]

$P$: 
\[
\begin{array}{cccccccc}
 b & c & b & c & a & a & b \\
\end{array}
\]
Find the leftmost (at most) \( k + 1 \) mismatches between \( T[i \ldots i + m - 1] \) and \( P \) (we’ll do this for each \( i \) seperately)
$k$-mismatch using LCP queries

Find the leftmost (at most) $k + 1$ mismatches between $T[i \ldots i + m - 1]$ and $P$ (we’ll do this for each $i$ separately)
Find the leftmost (at most) $k + 1$ mismatches between $T[i \ldots i + m - 1]$ and $P$
(we’ll do this for each $i$ separately)

We can do this using (at most) $k + 1$ LCP queries
Find the leftmost (at most) $k + 1$ mismatches between $T[i \ldots i + m - 1]$ and $P$

(we’ll do this for each $i$ separately)

We can do this using (at most) $k + 1$ LCP queries

*each query finds a new mismatch*
$k$-mismatch using LCP queries

Find the leftmost (at most) $k + 1$ mismatches between $T[i \ldots i + m - 1]$ and $P$ (we’ll do this for each $i$ seperately)

We can do this using (at most) $k + 1$ LCP queries

*each query finds a new mismatch*
$k$-mismatch using LCP queries

Find the leftmost (at most) $k + 1$ mismatches between $T[i \ldots i + m - 1]$ and $P$ (we'll do this for each $i$ separately)

We can do this using (at most) $k + 1$ LCP queries

*each query finds a new mismatch*
$k$-mismatch using LCP queries

Find the leftmost (at most) $k + 1$ mismatches between $T[i \ldots i + m - 1]$ and $P$ (we'll do this for each $i$ separately)

We can do this using (at most) $k + 1$ LCP queries

*each query finds a new mismatch*
Find the leftmost (at most) $k + 1$ mismatches between $T[i \ldots i + m - 1]$ and $P$ (we’ll do this for each $i$ seperately)

We can do this using (at most) $k + 1$ LCP queries

*each query finds a new mismatch*
**k-mismatch using LCP queries**

Find the leftmost (at most) \(k + 1\) mismatches between \(T[i \ldots i + m - 1]\) and \(P\) (we’ll do this for each \(i\) seperately)

We can do this using (at most) \(k + 1\) LCP queries

*each query finds a new mismatch*
Find the leftmost (at most) $k + 1$ mismatches between $T[i \ldots i + m - 1]$ and $P$

(we’ll do this for each $i$ separately)

We can do this using (at most) $k + 1$ LCP queries

*each query finds a new mismatch*
Find the leftmost (at most) \( k + 1 \) mismatches between \( T[i \ldots i + m - 1] \) and \( P \) (we’ll do this for each \( i \) seperately)

We can do this using (at most) \( k + 1 \) LCP queries

\[ \text{each query finds a new mismatch} \]
$k$-mismatch using LCP queries

Find the leftmost (at most) $k + 1$ mismatches between $T[i \ldots i + m - 1]$ and $P$
(we’ll do this for each $i$ seperately)

We can do this using (at most) $k + 1$ LCP queries

*each query finds a new mismatch*
Find the leftmost (at most) $k + 1$ mismatches between $T[i \ldots i + m - 1]$ and $P$ (we’ll do this for each $i$ seperately)

We can do this using (at most) $k + 1$ LCP queries

*each query finds a new mismatch*
Find the leftmost (at most) $k + 1$ mismatches between $T[i \ldots i + m - 1]$ and $P$ (we’ll do this for each $i$ separately)

We can do this using (at most) $k + 1$ LCP queries

$each$ $query$ $finds$ $a$ $new$ $mismatch$
Find the leftmost (at most) $k + 1$ mismatches between $T[i \ldots i + m - 1]$ and $P$
(we’ll do this for each $i$ seperately)

We can do this using (at most) $k + 1$ LCP queries

*each query finds a new mismatch*
Find the leftmost (at most) $k + 1$ mismatches between $T[i \ldots i + m - 1]$ and $P$

(we’ll do this for each $i$ seperately)

We can do this using (at most) $k + 1$ LCP queries

*each query finds a new mismatch*
Find the leftmost (at most) $k + 1$ mismatches between $T[i \ldots i + m - 1]$ and $P$ (we'll do this for each $i$ separately).

We can do this using (at most) $k + 1$ LCP queries.

*Each query finds a new mismatch*
Find the leftmost (at most) \( k + 1 \) mismatches between \( T[i \ldots i + m - 1] \) and \( P \) (we’ll do this for each \( i \) separately)

We can do this using (at most) \( k + 1 \) LCP queries

\textit{each query finds a new mismatch}

\textit{(or we reach the end of \( P \))}
Find the leftmost (at most) $k + 1$ mismatches between $T[i \ldots i + m - 1]$ and $P$ (we’ll do this for each $i$ separately)

We can do this using (at most) $k + 1$ LCP queries

*each query finds a new mismatch*

*(or we reach the end of $P$)*

We can therefore calculate $\text{Ham}_k(i)$ in $O(k)$ time
Find the leftmost (at most) \( k + 1 \) mismatches between \( T[i \ldots i + m - 1] \) and \( P \) (we’ll do this for each \( i \) seperately)

We can do this using (at most) \( k + 1 \) LCP queries

\[
\text{each query finds a new mismatch}
\]

\[
\text{(or we reach the end of } P)\]

We can therefore calculate \( \text{Ham}_k(i) \) in \( O(k) \) time

Overall this takes \( O(nk) \) time (including the ‘preprocessing’ for LCP queries)
Find the leftmost (at most) $k + 1$ mismatches between $T[i \ldots i + m - 1]$ and $P$ (we’ll do this for each $i$ seperately)

We can do this using (at most) $k + 1$ LCP queries

*each query finds a new mismatch*

*(or we reach the end of $P$)*

We can therefore calculate $\text{Ham}_k(i)$ in $O(k)$ time

Overall this takes $O(nk)$ time (including the ‘preprocessing’ for LCP queries)

*this is pretty good but we can do better*
Definition: A symbol is frequent if it occurs at least $\sqrt{k}$ times in $P$, and infrequent otherwise.
\(k\)-mismatch using frequent/infrequent symbols

**Definition:** A symbol is *frequent* if it occurs at least \(\sqrt{k}\) times in \(P\), and *infrequent* otherwise

\[
P = [a, b, b, a, c, a, d, b, d]
\]

\(m = 9\)  \(k = 4\)  \((\sqrt{k} = 2)\)
$k$-mismatch using frequent/infrequent symbols

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{k}$ times in $P$, and *infrequent* otherwise.

\[ k = 4 \quad (\sqrt{k} = 2) \]
$k$-mismatch using frequent/infrequent symbols

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{k}$ times in $P$, and *infrequent* otherwise.

\[ k = 4 \quad (\sqrt{k} = 2) \]

$P$:

\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
a & b & b & a & c & a & d & b & d \\
\end{array}

\[ a \text{ is frequent} \]
**k**-mismatch using frequent/infrequent symbols

**Definition:** A symbol is *frequent* if it occurs at least \( \sqrt{k} \) times in \( P \), and *infrequent* otherwise.

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
\text{a} & \text{b} & \text{b} & \text{a} & \text{c} & \text{a} & \text{d} & \text{b} & \text{d} \\
\end{array}
\]

\[k = 4\quad (\sqrt{k} = 2)\]

\( a \) is frequent, \( b \) is frequent
\( k \)-mismatch using frequent/infrequent symbols

**Definition:** A symbol is *frequent* if it occurs at least \( \sqrt{k} \) times in \( P \), and *infrequent* otherwise.

\[
P = \begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
a & b & b & a & c & a & d & b & d
\end{array}
\]

\( k = 4 \) \hspace{1cm} (\sqrt{k} = 2)

\( a \) is frequent, \( b \) is frequent, \( d \) is frequent.
**$k$**-mismatch using frequent/infrequent symbols

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{k}$ times in $P$, and *infrequent* otherwise.

$k = 4$  \hspace{1cm} \left(\sqrt{k} = 2\right)

\begin{align*}
P & \quad \begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\text{a} & \text{b} & \text{b} & \text{a} & \text{c} & \text{a} & \text{d} & \text{b} & \text{d}
\end{array} \\
\end{align*}

$a$ is frequent, $b$ is frequent, $d$ is frequent
$c$ is infrequent
\( k \)-mismatch using frequent/infrequent symbols

**Definition:** A symbol is *frequent* if it occurs at least \( \sqrt{k} \) times in \( P \), and *infrequent* otherwise.

\( k = 4 \) \((\sqrt{k} = 2)\)

\( a \) is frequent, \( b \) is frequent, \( d \) is frequent
\( c \) is infrequent

*How many frequent symbols can there be?*
**$k$-mismatch using frequent/infrequent symbols**

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{k}$ times in $P$, and *infrequent* otherwise.

<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>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>b</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>d</td>
<td>b</td>
<td>d</td>
</tr>
</tbody>
</table>

$k = 4$ \hspace{1cm} ($\sqrt{k} = 2$)

$a$ is frequent, $b$ is frequent, $d$ is frequent
$c$ is infrequent

*How many frequent symbols can there be? Lots!*
$k$-mismatch using frequent/infrequent symbols

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{k}$ times in $P$, and *infrequent* otherwise.

<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>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>b</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>d</td>
<td>b</td>
<td>d</td>
</tr>
</tbody>
</table>

$k = 4$ \quad (\sqrt{k} = 2)

- $a$ is frequent
- $b$ is frequent
- $d$ is frequent
- $c$ is infrequent

How many frequent symbols can there be? **Lots!** there could be $\frac{m}{\sqrt{k}}$ frequent symbols
**k**-mismatch using frequent/infrequent symbols

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{k}$ times in $P$, and *infrequent* otherwise.

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

```
P: a b b a c a d b d
```

$k = 4$ ($\sqrt{k} = 2$)

- $a$ is frequent, $b$ is frequent, $d$ is frequent
- $c$ is infrequent

*How many frequent symbols can there be? Lots!* there could be $\frac{m}{\sqrt{k}}$ frequent symbols

**Case 1:** There are fewer than $2\sqrt{k}$ frequent symbols in $P$. 
$k$-mismatch using frequent/infrequent symbols

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{k}$ times in $P$, and *infrequent* otherwise.

<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>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>b</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td>d</td>
<td>b</td>
<td>d</td>
</tr>
</tbody>
</table>

$k = 4$  
($\sqrt{k} = 2$)

$a$ is frequent, $b$ is frequent, $d$ is frequent  
c is infrequent

*How many frequent symbols can there be?* **Lots!** there could be $\frac{m}{\sqrt{k}}$ frequent symbols

**Case 1:** There are fewer than $2\sqrt{k}$ frequent symbols in $P$.

**Algorithm summary**
$k$-mismatch using frequent/infrequent symbols

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{k}$ times in $P$, and *infrequent* otherwise.

![Sequence Example](sequence_example.png)

- $a$ is frequent, $b$ is frequent, $d$ is frequent
- $c$ is infrequent

**Algorithm summary**

**Step 0:** Classify each symbol as frequent or infrequent

**Step 1:** Count all matches involving frequent symbols (using convolutions)

**Step 2:** Count all matches involving infrequent symbols (as before)

**Case 1:** There are fewer than $2\sqrt{k}$ frequent symbols in $P$.  

There could be $\frac{m}{\sqrt{k}}$ frequent symbols.
$k$-mismatch using frequent/infrequent symbols

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{k}$ times in $P$, and *infrequent* otherwise.

![Symbol frequencies](image)

- $a$ is frequent, $b$ is frequent, $d$ is frequent
- $c$ is infrequent

*How many frequent symbols can there be?* **Lots!** There could be $\frac{m}{\sqrt{k}}$ frequent symbols.

**Case 1:** There are fewer than $2\sqrt{k}$ frequent symbols in $P$.

**Algorithm summary**

- **Step 0:** Classify each symbol as frequent or infrequent - $O(m \log m)$ time
- **Step 1:** Count all matches involving frequent symbols (using convolutions)
- **Step 2:** Count all matches involving infrequent symbols (as before)
**$k$-mismatch using frequent/infrequent symbols**

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{k}$ times in $P$, and *infrequent* otherwise.

```
0 1 2 3 4 5 6 7 8
```
```
P:  a b b a c a d b d
```

$k = 4$ ($\sqrt{k} = 2$)

*a is frequent, b is frequent, d is frequent*  
*c is infrequent*

*How many frequent symbols can there be? Lots!*  
there could be $\frac{m}{\sqrt{k}}$ frequent symbols

---

**Case 1:** There are fewer than $2\sqrt{k}$ frequent symbols in $P$.

**Algorithm summary**

**Step 0:** Classify each symbol as frequent or infrequent  
- $O(m \log m)$ time

**Step 1:** Count all matches involving frequent symbols (using convolutions)  
- $O(n\sqrt{k} \log m)$ time

**Step 2:** Count all matches involving infrequent symbols (as before)
**k**-mismatch using frequent/infrequent symbols

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{k}$ times in $P$, and *infrequent* otherwise.

$$P = \text{[a b b a c a d b d]}$$

$k = 4$  
($\sqrt{k} = 2$)

$a$ is frequent, $b$ is frequent, $d$ is frequent  
$c$ is infrequent

*How many frequent symbols can there be?*  **Lots!**  there could be $\frac{m}{\sqrt{k}}$ frequent symbols

---

**Case 1:** There are fewer than $2\sqrt{k}$ frequent symbols in $P$.

**Algorithm summary**

**Step 0:** Classify each symbol as frequent or infrequent  
- $O(m \log m)$ time

**Step 1:** Count all matches involving frequent symbols (using convolutions)  
- $O(n\sqrt{k} \log m)$ time

**Step 2:** Count all matches involving infrequent symbols (as before)  
- $O(n\sqrt{k})$ time
**$k$-mismatch using frequent/infrequent symbols**

**Definition:** A symbol is *frequent* if it occurs at least $\sqrt{k}$ times in $P$, and *infrequent* otherwise.

![Symbol Pattern](image)

$k = 4$ ($\sqrt{k} = 2$)

- $a$ is frequent, $b$ is frequent, $d$ is frequent
- $c$ is infrequent

How many frequent symbols can there be? **Lots!** there could be $\frac{m}{\sqrt{k}}$ frequent symbols

---

**Case 1:** There are fewer than $2\sqrt{k}$ frequent symbols in $P$. - $O(n\sqrt{k} \log m)$ total time

**Algorithm summary**

**Step 0:** Classify each symbol as frequent or infrequent - $O(m \log m)$ time

**Step 1:** Count all matches involving frequent symbols (using convolutions) - $O(n\sqrt{k} \log m)$ time

**Step 2:** Count all matches involving infrequent symbols (as before) - $O(n\sqrt{k})$ time
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$

\[
P = \begin{array}{cccccccccc}
  a & e & b & b & a & c & a & d & b & d & c & f & b & b \\
\end{array}
\]

$k = 4$
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurences in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$.

$$P = \text{a e b b a c a d b d c f b b}$$

$k = 4$
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$

$P = \begin{array}{cccccccccc}
    a & e & b & b & a & c & a & d & b & d & c & f & b & b \\
\end{array}$

$k = 4$
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$.

This gives us $2k$ *interesting* pattern locations, denoted $J$

\[
P = \begin{array}{cccccccccccc}
  a & e & b & b & a & c & a & d & b & d & c & f & b & b \\
\end{array}
\]

$k = 4$
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$

$$J = \{0, 2, 3, 4, 5, 7, 9, 10\}$$
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$

$$J = \{0, 2, 3, 4, 5, 7, 9, 10\}$$

$$P = \begin{array}{cccccccccc}
\text{a} & \text{e} & \text{b} & \text{b} & \text{a} & \text{c} & \text{a} & \text{d} & \text{b} & \text{d} & \text{c} & \text{f} & \text{b} & \text{b} \\
\end{array}$$

$k = 4$
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$

$J = \{0, 2, 3, 4, 5, 7, 9, 10\}$

Let $d_k(i)$ be the number of $j \in J$ such that $P[j] = T[i + j]$ i.e. the number of (single character) matches involving interesting pattern locations
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$

$J = \{0, 2, 3, 4, 5, 7, 9, 10\}$

Let $d_k(i)$ be the number of $j \in J$ such that $P[j] = T[i + j]$

i.e. the number of (single character) matches involving interesting pattern locations
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurances in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$

$J = \{0, 2, 3, 4, 5, 7, 9, 10\}$

$P = a e b b a c a d b d c f b b$

$k = 4$

$T = a c c a a b a b b a c f c d e f f b b c e a e$

$i = 4$

$d_k(i) = 3$

Let $d_k(i)$ be the number of $j \in J$ such that $P[j] = T[i + j]$

i.e. the number of (single character) matches involving interesting pattern locations
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$

\[ J = \{0, 2, 3, 4, 5, 7, 9, 10\} \]

Let $d_k(i)$ be the number of $j \in J$ such that $P[j] = T[i + j]$ i.e. the number of (single character) matches involving interesting pattern locations

**Fact** if $d_k(i) < k$ then there are more than $k$ mismatches (i.e. $\text{Ham}_k(i) = X$)
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$

$$J = \{0, 2, 3, 4, 5, 7, 9, 10\}$$

\[ \begin{array}{cccccccccc}
  P & a & e & b & b & a & c & a & d & b & d & c & f & b & b \\
  T & a & c & c & a & a & b & a & b & b & a & c & f & c & d & e & f & f & b & b & c & e & a & e \\
\end{array} \]

\[ i = 4 \quad d_k(i) = 3 \]

Let $d_k(i)$ be the number of $j \in J$ such that $P[j] = T[i + j]$

i.e. the number of (single character) matches involving interesting pattern locations

Fact if $d_k(i) < k$ then there are more than $k$ mismatches (i.e. $\text{Ham}_k(i) = X$)

because there are $2k$ interesting positions... and fewer than $k$ of them match
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$. This gives us $2k$ interesting pattern locations, denoted $J$

$$J = \{0, 2, 3, 4, 5, 7, 9, 10\}$$

$$P = \begin{array}{cccccccccc}
    a & e & b & b & a & c & a & d & b & d & c & f & b & b
\end{array}$$

$$T = \begin{array}{cccccccccccc}
    a & c & c & a & a & b & a & b & b & a & c & f & c & d & e & f & f & b & b & c & e & a & e
\end{array}$$

$$i = 4 \quad d_k(i) = 3$$

Let $d_k(i)$ be the number of $j \in J$ such that $P[j] = T[i + j]$

i.e. the number of (single character) matches involving interesting pattern locations

**Fact** if $d_k(i) < k$ then there are more than $k$ mismatches (i.e. $\text{Ham}_k(i) = X$)

because there are $2k$ interesting positions... and fewer than $k$ of them match

**Fact** There are at most $n/\sqrt{k}$ values of $i$ with $d_k(i) \geq k$
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$

$$J = \{0, 2, 3, 4, 5, 7, 9, 10\}$$

Let $d_k(i)$ be the number of $j \in J$ such that $P[j] = T[i + j]$

i.e. the number of (single character) matches involving interesting pattern locations

**Fact** if $d_k(i) < k$ then there are more than $k$ mismatches (i.e. $\text{Ham}_k(i) = X$)

because there are $2k$ interesting positions... and fewer than $k$ of them match

**Fact** There are at most $n/\sqrt{k}$ values of $i$ with $d_k(i) \geq k$

this follows from a counting argument
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$. This gives us $2k$ interesting pattern locations, denoted $J$

\[ J = \{0, 2, 3, 4, 5, 7, 9, 10\} \]

\[ P = \begin{array}{cccccccccccc} a & e & b & b & a & c & a & d & b & d & c & f & b & b \\ \end{array} \quad k = 4 \]

\[ T = \begin{array}{cccccccccccc} a & c & c & a & a & b & a & b & b & a & c & f & c & d & e & f & f & b & b & c & e & a & e \\ \end{array} \]

Let $d_k(i)$ be the number of $j \in J$ such that $P[j] = T[i + j]$

i.e. the number of (single character) matches involving interesting pattern locations

Fact There are at most $n/\sqrt{k}$ values of $i$ with $d_k(i) \geq k$
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$

Let $d_k(i)$ be the number of $j \in J$ such that $P[j] = T[i + j]$

Fact: There are at most $n/\sqrt{k}$ values of $i$ with $d_k(i) \geq k$

For any location $i'$, $T[i'] = P[j]$ for either 0 or $\sqrt{k}$ distinct $j \in J$
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$

\[ J = \{0, 2, 3, 4, 5, 7, 9, 10\} \]

Let $d_k(i)$ be the number of $j \in J$ such that $P[j] = T[i + j]$ i.e. the number of (single character) matches involving interesting pattern locations

**Fact** There are at most $n/\sqrt{k}$ values of $i$ with $d_k(i) \geq k$

For any location $i'$, $T[i'] = P[j]$ for either 0 or $\sqrt{k}$ distinct $j \in J$

This implies that $\sum_i d_k(i) \leq \sum_i \sum_{j \in J} \text{Eq}(T[i'], P[j]) \leq n\sqrt{k}$
Case 2: There are at least \(2\sqrt{k}\) frequent symbols

Pick any \(2\sqrt{k}\) frequent symbols and for each symbol pick \(\sqrt{k}\) occurences in \(P\).

This gives us \(2k\) interesting pattern locations, denoted \(J\)

\[
J = \{0, 2, 3, 4, 5, 7, 9, 10\}
\]

\[
P = \begin{array}{cccccccccc}
a & e & b & b & a & c & a & d & b & d & c & f & b & b \\
\end{array}
\]

\[
k = 4
\]

\[
T = \begin{array}{cccccccccc}
a & c & c & a & a & b & a & b & b & a & c & f & c & d & e & f & f & b & b & c & e & a & e \\
\end{array}
\]

\[
i = 4 \quad d_k(i) = 3
\]

Let \(d_k(i)\) be the number of \(j \in J\) such that \(P[j] = T[i + j]\)

\[i.e. \ the \ number \ of \ (single \ character) \ matches \ involving \ interesting \ pattern \ locations\]

Fact: There are at most \(n/\sqrt{k}\) values of \(i\) with \(d_k(i) \geq k\)

For any location \(i', T[i'] = P[j]\) for either 0 or \(\sqrt{k}\) distinct \(j \in J\)

This implies that \(\sum_i d_k(i) \leq \sum_{i'} \sum_{j \in J} \text{Eq}(T[i'], P[j]) \leq n\sqrt{k}\)
Case 2: There are at least \(2\sqrt{k}\) frequent symbols

Pick any \(2\sqrt{k}\) frequent symbols and for each symbol pick \(\sqrt{k}\) occurrences in \(P\).

This gives us \(2k\) interesting pattern locations, denoted \(J\)

\[
J = \{0, 2, 3, 4, 5, 7, 9, 10\}
\]

Let \(d_k(i)\) be the number of \(j \in J\) such that \(P[j] = T[i + j]\) i.e. the number of (single character) matches involving interesting pattern locations

**Fact** There are at most \(n/\sqrt{k}\) values of \(i\) with \(d_k(i) \geq k\)

For any location \(i'\), \(T[i'] = P[j]\) for either 0 or \(\sqrt{k}\) distinct \(j \in J\)

This implies that

\[
\sum_i d_k(i) \leq \sum_{i'} \sum_{j \in J} \text{Eq}(T[i'], P[j]) \leq n\sqrt{k}
\]
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$

$J = \{0, 2, 3, 4, 5, 7, 9, 10\}$

$P$:

```
| a | e | b | b | a | c | a | d | b | d | c | f | b | b |
```

$k = 4$

$T$:

```
| a | c | c | a | a | b | a | b | b | a | c | f | c | d | e | f | f | b | b | c | e | a | e |
```

$i = 4$

$d_k(i) = 3$

Let $d_k(i)$ be the number of $j \in J$ such that $P[j] = T[i + j]$

i.e. the number of (single character) matches involving interesting pattern locations

Fact: There are at most $n/\sqrt{k}$ values of $i$ with $d_k(i) \geq k$

Assume that more than $n/\sqrt{k}$ values of $i$ have $d_k(i) \geq k$
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$

Let $d_k(i)$ be the number of $j \in J$ such that $P[j] = T[i + j]$ i.e. the number of (single character) matches involving interesting pattern locations

Fact There are at most $n/\sqrt{k}$ values of $i$ with $d_k(i) \geq k$

Assume that more than $n/\sqrt{k}$ values of $i$ have $d_k(i) \geq k$

So $\sum_i d_k(i) \geq \frac{n}{\sqrt{k}} \cdot k$
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurences in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$

$$J = \{0, 2, 3, 4, 5, 7, 9, 10\}$$

Let $d_k(i)$ be the number of $j \in J$ such that $P[j] = T[i + j]$ i.e. the number of (single character) matches involving interesting pattern locations

Fact There are at most $n/\sqrt{k}$ values of $i$ with $d_k(i) \geq k$

Assume that more than $n/\sqrt{k}$ values of $i$ have $d_k(i) \geq k$

So $\sum_i d_k(i) \geq \frac{n}{\sqrt{k}} \cdot k > n\sqrt{k}$
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$. This gives us $2k$ interesting pattern locations, denoted $J$

$J = \{0, 2, 3, 4, 5, 7, 9, 10\}$

$P = \begin{array}{cccccccccc}
    a & e & b & b & a & c & a & d & b & d & c & f & b & b \\
\end{array}$

$T = \begin{array}{cccccccccc}
    a & c & c & a & a & b & a & b & b & a & c & f & c & d & e & f & f & b & b & c & e & a & e \\
\end{array}$

$k = 4$

$i = 4$

$d_k(i) = 3$

Let $d_k(i)$ be the number of $j \in J$ such that $P[j] = T[i + j]$

i.e. the number of (single character) matches involving interesting pattern locations

**Fact** There are at most $n/\sqrt{k}$ values of $i$ with $d_k(i) \geq k$

Assume that more than $n/\sqrt{k}$ values of $i$ have $d_k(i) \geq k$

So $\sum_i d_k(i) \geq \frac{n}{\sqrt{k}} \cdot k > n\sqrt{k}$

Contradiction!
**Case 2: There are at least** $2\sqrt{k}$ **frequent symbols**

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$.

This gives us $2k$ *interesting* pattern locations, denoted $J$.

Let $d_k(i)$ be the number of $j \in J$ such that $P[j] = T[i + j]$ 
*i.e. the number of (single character) matches involving interesting pattern locations*

**Fact** if $d_k(i) < k$ then there are more than $k$ mismatches (i.e. $\text{Ham}_k(i) = X$) 
*because there are $2k$ interesting positions... and fewer than $k$ of them match*

**Fact** There are at most $n/\sqrt{k}$ values of $i$ with $d_k(i) \geq k$  
*this follows from a counting argument*
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$

$P = \begin{array}{cccccccccccc}
  a & e & b & b & a & c & a & d & b & d & c & f & b & b \\
\end{array}$

$k = 4$

$T = \begin{array}{cccccccccccc}
  a & c & c & a & a & b & a & b & b & a & c & f & c & d & e & f & f & b & b & c & e & a & e \\
\end{array}$

$i = 4$

We can filter the text, leaving only $n/\sqrt{k}$ locations to check every other location has more than $k$ mismatches
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$

$J = \{0, 2, 3, 4, 5, 7, 9, 10\}$

We can filter the text, leaving only $n/\sqrt{k}$ locations to check every other location has more than $k$ mismatches

Check each of the remaining locations using LCP queries in $O(k)$ time
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$

$J = \{0, 2, 3, 4, 5, 7, 9, 10\}$

We can filter the text, leaving only $n/\sqrt{k}$ locations to check

every other location has more than $k$ mismatches

Check each of the remaining locations using LCP queries in $O(k)$ time

Determining which locations to check also takes $O(n\sqrt{k})$ total time
Case 2: There are at least $2\sqrt{k}$ frequent symbols

Pick any $2\sqrt{k}$ frequent symbols and for each symbol pick $\sqrt{k}$ occurrences in $P$.

This gives us $2k$ interesting pattern locations, denoted $J$

$J = \{0, 2, 3, 4, 5, 7, 9, 10\}$

We can filter the text, leaving only $n/\sqrt{k}$ locations to check

every other location has more than $k$ mismatches

Check each of the remaining locations using LCP queries in $O(k)$ time

Determining which locations to check also takes $O(n\sqrt{k})$ total time

This gives $O(n\sqrt{k})$ total time
Pattern matching with k-mismatches: putting it all together

Algorithm summary
Pattern matching with k-mismatches: putting it all together

Algorithm summary

Preprocess $P, T$ for LCP queries - $O(n)$ time
Algorithm summary

Preprocess $P, T$ for LCP queries - $O(n)$ time
Count the number of frequent symbols in $P$ - $O(m \log m)$ time
Pattern matching with k-mismatches: putting it all together

Algorithm summary

- Preprocess $P, T$ for LCP queries - $O(n)$ time
- Count the number of frequent symbols in $P$ - $O(m \log m)$ time

Case 1: $P$ has at most $2\sqrt{k}$ frequent symbols

Case 2: $P$ has more than $2\sqrt{k}$ frequent symbols
Algorithm summary

Preprocess $P, T$ for LCP queries - $O(n)$ time
Count the number of frequent symbols in $P$ - $O(m \log m)$ time

Case 1: $P$ has at most $2\sqrt{k}$ frequent symbols

Case 2: $P$ has more than $2\sqrt{k}$ frequent symbols
Pattern matching with k-mismatches: putting it all together

Algorithm summary

Preprocess $P, T$ for LCP queries - $O(n)$ time

Count the number of frequent symbols in $P$ - $O(m \log m)$ time

Case 1: $P$ has at most $2\sqrt{k}$ frequent symbols

Count matches with frequent symbols using convolution - $O(n\sqrt{k} \log m)$ time

Case 2: $P$ has more than $2\sqrt{k}$ frequent symbols
Pattern matching with k-mismatches: putting it all together

Algorithm summary

Preprocess $P, T$ for LCP queries - $O(n)$ time
Count the number of frequent symbols in $P$ - $O(m \log m)$ time

Case 1: $P$ has at most $2\sqrt{k}$ frequent symbols
  Count matches with frequent symbols using convolution - $O(n\sqrt{k} \log m)$ time
  Count matches with infrequent symbols directly - $O(n\sqrt{k})$ time

Case 2: $P$ has more than $2\sqrt{k}$ frequent symbols
Algorithm summary

Preprocess $P, T$ for LCP queries - $O(n)$ time
Count the number of frequent symbols in $P$ - $O(m \log m)$ time

Case 1: $P$ has at most $2\sqrt{k}$ frequent symbols
   Count matches with frequent symbols using convolution - $O(n\sqrt{k} \log m)$ time
   Count matches with infrequent symbols directly - $O(n\sqrt{k})$ time

Case 2: $P$ has more than $2\sqrt{k}$ frequent symbols
   Filter the text, leaving $n/\sqrt{k}$ alignments - $O(n\sqrt{k})$ time
Pattern matching with k-mismatches: putting it all together

Algorithm summary

Preprocess $P, T$ for LCP queries - $O(n)$ time

Count the number of frequent symbols in $P$ - $O(m \log m)$ time

Case 1: $P$ has at most $2\sqrt{k}$ frequent symbols

Count matches with frequent symbols using convolution - $O(n\sqrt{k} \log m)$ time

Count matches with infrequent symbols directly - $O(n\sqrt{k})$ time

Case 2: $P$ has more than $2\sqrt{k}$ frequent symbols

Filter the text, leaving $n/\sqrt{k}$ alignments - $O(n\sqrt{k})$ time

Count mismatches at these alignments using LCP queries - $O(n\sqrt{k})$ time
Pattern matching with k-mismatches: putting it all together

Algorithm summary

Preprocess $P, T$ for LCP queries - $O(n)$ time
Count the number of frequent symbols in $P$ - $O(m \log m)$ time

Case 1: $P$ has at most $2\sqrt{k}$ frequent symbols
  Count matches with frequent symbols using convolution - $O(n\sqrt{k} \log m)$ time
  Count matches with infrequent symbols directly - $O(n\sqrt{k})$ time

Case 2: $P$ has more than $2\sqrt{k}$ frequent symbols
  Filter the text, leaving $n/\sqrt{k}$ alignments - $O(n\sqrt{k})$ time
  Count mismatches at these alignments using LCP queries - $O(n\sqrt{k})$ time

Overall, we obtain a time complexity of $O(n\sqrt{k} \log m)$. 
Pattern matching with k-mismatches: putting it all together

Algorithm summary

Preprocess \( P, T \) for LCP queries - \( O(n) \) time
Count the number of frequent symbols in \( P \) - \( O(m \log m) \) time

Case 1: \( P \) has at most \( 2\sqrt{k} \) frequent symbols

- Count matches with frequent symbols using convolution - \( O(n\sqrt{k} \log m) \) time
- Count matches with infrequent symbols directly - \( O(n\sqrt{k}) \) time

Case 2: \( P \) has more than \( 2\sqrt{k} \) frequent symbols

- Filter the text, leaving \( n/\sqrt{k} \) alignments - \( O(n\sqrt{k}) \) time
- Count mismatches at these alignments using LCP queries - \( O(n\sqrt{k}) \) time

Overall, we obtain a time complexity of \( O(n\sqrt{k} \log m) \).
- this can be improved to \( O(n\sqrt{k} \log k) \)