Pattern matching part three

Hamming distance

Benjamin Sach
Exact pattern matching

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

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

**Goal:** Find all the locations where $P$ matches in $T$

$P$ matches at location $i$ iff for all $0 \leq j < m$ we have that $P[j] = T[i + j]$

*(our strings are zero-indexed)*
Exact pattern matching

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

Goal: Find all the locations where $P$ matches in $T$

$P$ matches at location $i$ iff for all $0 \leq j < m$ we have that $P[j] = T[i + j]$

(our strings are zero-indexed)
Exact pattern matching

**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>$b$</td>
<td>$a$</td>
<td>$b$</td>
<td>$a$</td>
<td>$b$</td>
<td>$a$</td>
<td>$c$</td>
<td>$a$</td>
<td>$b$</td>
<td>$a$</td>
</tr>
<tr>
<td>$P$</td>
<td>$a$</td>
<td>$b$</td>
<td>$a$</td>
<td>$a$</td>
<td>$b$</td>
<td>$a$</td>
<td>$a$</td>
<td>$b$</td>
<td>$a$</td>
<td>$a$</td>
<td>$b$</td>
<td>$a$</td>
<td>$a$</td>
</tr>
</tbody>
</table>

**Goal:** Find all the locations where $P$ matches in $T$

$P$ matches at location $i$ iff there exists $0 \leq j < m$ such that for all $0 \leq j < m$ we have that $P[j] = T[i + j]$

(our strings are zero-indexed)
Exact pattern matching

**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>b</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>$P$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>a</td>
<td>b</td>
<td>a</td>
</tr>
</tbody>
</table>

**Goal:** Find all the locations where $P$ matches in $T$

$P$ matches at location $i$ iff

for all $0 \leq j < m$ we have that $P[j] = T[i + j]$

*(our strings are zero-indexed)*
Exact pattern matching

**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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
</tr>
</tbody>
</table>

**Goal:** Find all the locations where $P$ matches in $T$

$P$ matches at location $i$ iff for all $0 \leq j < m$ we have that $P[j] = T[i + j]$

*(our strings are zero-indexed)*
Exact pattern matching

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

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

**Goal:** Find all the locations where $P$ matches in $T$

$P$ matches at location $i$ iff for all $0 \leq j < m$ we have that $P[j] = T[i + j]$

(our strings are zero-indexed)
Exact pattern matching

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

**Goal:** Find all the locations where $P$ matches in $T$

$P$ matches at location $i$ iff for all $0 \leq j < m$ we have that $P[j] = T[i + j]$

(our strings are zero-indexed)

- A naive algorithm takes $O(nm)$ time
Exact pattern matching

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

```
T: a b c b a b a c a b a
P: a b a
```

**Goal:** Find all the locations where $P$ matches in $T$

$P$ matches at location $i$ iff for all $0 \leq j < m$ we have that $P[j] = T[i + j]$

(our strings are zero-indexed)

- A naive algorithm takes $O(nm)$ time
- Many $O(n)$ time algorithms are known (for example the KMP algorithm)
Pattern matching with mismatches

**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 \\
\end{array}

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

$P$

$T$

$n$

$P$

$m$

**Goal:** For every *alignment* $i$, output

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

*The Hamming distance is the number of mismatches...*

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

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

Goal: For every alignment $i$, output

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

The Hamming distance is the number of mismatches…

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

Ham(4) = 1
Pattern matching with mismatches

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

```
T: a b c d a b a a d a c a a
P: a b d a
```

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

*The Hamming distance is the number of mismatches...*

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

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

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

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

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

[Diagram of text string $T$ and pattern string $P$]

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

*The Hamming distance is the number of mismatches...*

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

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

\[
\begin{array}{c}
\text{T} \\
\text{a b c d a b a a d d c a a} \\
\text{P} \\
\text{a b d a} \\
\end{array}
\]

this is alignment 7

\[
\begin{array}{c}
\text{n} \\
0 1 2 3 4 5 6 7 8 9 10 11 12 \\
\end{array}
\]

\[
\begin{array}{c}
\text{m} \\
\end{array}
\]

\[\text{Ham}(7) = 3\]

**Goal:** For every alignment $i$, output

\[\text{Ham}(i), \text{ the Hamming distance between } P \text{ and } T[i \ldots i + m - 1]\]

*The Hamming distance is the number of mismatches...*

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

**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>
<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>
</tr>
</tbody>
</table>

this is **alignment 8**

$\text{Ham}(8) = 3$

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

The Hamming distance is the number of mismatches...

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

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

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

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

A naive algorithm for this problem takes $O(nm)$ time
Pattern matching with mismatches

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

![Diagram of text and pattern strings]

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

*The Hamming distance is the number of mismatches...*

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

A naive algorithm for this problem takes $O(nm)$ time

...but we can do better
Imagine that the alphabet contains only a small number of different symbols, which we will consider individually...
It's a small alphabet after all

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

Replace all $d$ symbols with 1 and everything else with 0
It's a small alphabet after all

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

\[
\begin{array}{c}
\text{Replace all } d \text{ symbols with 1 and everything else with 0}
\end{array}
\]
It's a small alphabet after all

Imagine that the alphabet contains only a small number of different symbols, which we will 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></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>
</tbody>
</table>

\[
T \quad \begin{array}{cccccccc}
1 & a & c & b & 1 & a & 1 & 1 & b & 1 & c & 1 & 1 \\
\end{array}
\]

\[
P \quad \begin{array}{ccc}
1 & a & b & 1 \\
\end{array}
\]

Replace all \( d \) symbols with 1 and everything else with 0
It’s a small alphabet after all

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

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

Replace all \textit{d} symbols with 1 and everything else with 0
It’s a small alphabet after all

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

Replace all \( d \) symbols with 1 and everything else with 0.
It's a small alphabet after all

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

Replace all $d$ symbols with 1 and everything else with 0.
Imagine that the alphabet contains only a small number of different symbols, which we will consider individually...

<table>
<thead>
<tr>
<th>0 1 2 3 4 5 6 7 8 9 10 11 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_d$</td>
</tr>
<tr>
<td>1 0 0 0 1 0 1 1 0 1 0 1 1</td>
</tr>
<tr>
<td>$P_d$</td>
</tr>
<tr>
<td>1 0 0 1</td>
</tr>
</tbody>
</table>

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

We denote these new strings $T_d$ and $P_d$ and consider...
Imagine that the alphabet contains only a small number of different symbols, which we will consider individually...

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

We denote these new strings $T_d$ and $P_d$ and consider...

$$(T_d \otimes P_d)[i] = \sum_{j=0}^{m-1} P_d[j] \times T_d[i + j]$$
Imagine that the alphabet contains only a small number of different symbols, which we will consider individually.

Replace all $d$ symbols with 1 and everything else with 0.

We denote these new strings $T_d$ and $P_d$ and consider:

$$(T_d \otimes P_d)[i] = \sum_{j=0}^{m-1} P_d[j] \times T_d[i + j]$$
It's a small alphabet after all

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

\[
(T_d \otimes P_d)[i] = \sum_{j=0}^{m-1} P_d[j] \times T_d[i + j]
\]

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

We denote these new strings $T_d$ and $P_d$ and consider...
It's a small alphabet after all

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

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

We denote these new strings \( T_d \) and \( P_d \) and consider...

\[
(T_d \otimes P_d)[i] = \sum_{j=0}^{m-1} P_d[j] \times T_d[i+j]
\]

1 iff \( P[j] = T[i+j] = d \)
It's a small alphabet after all

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

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

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

We denote these new strings \( T_d \) and \( P_d \) and consider...

\[
(T_d \otimes P_d)[i] = \sum_{j=0}^{m-1} P_d[j] \times T_d[i + j]
\]

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

\[
(T_d \otimes P_d)[4] = (1 \times 1) + (0 \times 0) + (1 \times 0) + (1 \times 1) = 2
\]
It's a small alphabet after all

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

We denote these new strings $T_d$ and $P_d$ and consider...

\[
(T_d \otimes P_d)[4] = (1 \times 1) + (0 \times 0) + (1 \times 0) + (1 \times 1) = 2
\]

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

We denote these new strings $T_d$ and $P_d$ and consider...

\[
(T_d \otimes P_d)[i] = \sum_{j=0}^{m-1} P_d[j] \times T_d[i+j]
\]

This is the exactly number of matching $d$s at the $i$-th alignment.

How can we work out $(T_d \otimes P_d)$ quickly?
Imagine that the alphabet contains only a small number of different symbols, which we will consider individually...

Replace all d symbols with 1 and everything else with 0

We denote these new strings $T_d$ and $P_d$ and consider...

\[
(T_d \otimes P_d)[i] = \sum_{j=0}^{m-1} P_d[j] \times T_d[i + j]
\]

This is the exactly number of matching d's at the i-th alignment.

How can we work out $(T_d \otimes P_d)$ quickly?
Last year on COMS21103...

Let $A$ and $B$ be $(n - 1)$ degree polynomials which can be expressed as...

$$A(x) = \sum_{i=0}^{n-1} a_i x^i \quad \text{and} \quad B(x) = \sum_{i=0}^{n-1} b_i x^i$$
Last year on COMS21103...  

Let $A$ and $B$ be $(n - 1)$ degree polynomials which can be expressed as...  

$$A(x) = \sum_{i=0}^{n-1} a_i x^i \quad \text{and} \quad B(x) = \sum_{i=0}^{n-1} b_i x^i$$  

$A[i] = a_i$  

(or be seen as arrays of length $n$)  

$B[i] = b_i$
Last year on COMS21103 . . .

Let $A$ and $B$ be $(n - 1)$ degree polynomials which can be expressed as . . .

\[
A(x) = \sum_{i=0}^{n-1} a_i x^i \quad \text{and} \quad B(x) = \sum_{i=0}^{n-1} b_i x^i
\]

The polynomial $C = A \times B$ can be expressed as . . .

\[
C(x) = \sum_{i=0}^{2n-1} c_i x^i \quad \text{where} \quad c_i = \sum_{j=0}^{i} a_j \times b_{(i-j)}
\]

A $A[i] = a_i$ 

B $B[i] = b_i$ (or be seen as arrays of length $n$)
Let \( A \) and \( B \) be \((n - 1)\) degree polynomials which can be expressed as:

\[
A(x) = \sum_{i=0}^{n-1} a_i x^i \quad \text{and} \quad B(x) = \sum_{i=0}^{n-1} b_i x^i
\]

The polynomial \( C = A \times B \) can be expressed as:

\[
C(x) = \sum_{i=0}^{2n-1} c_i x^i \quad \text{where} \quad c_i = \sum_{j=0}^{i} a_j \times b_{(i-j)}
\]
Let $A$ and $B$ be $(n - 1)$ degree polynomials which can be expressed as...

$$A(x) = \sum_{i=0}^{n-1} a_i x^i \quad \text{and} \quad B(x) = \sum_{i=0}^{n-1} b_i x^i$$

The polynomial $C = A \times B$ can be expressed as...

$$C(x) = \sum_{i=0}^{2n-1} c_i x^i \quad \text{where} \quad c_i = \sum_{j=0}^{i} a_j \times b_{i-j}$$

By the *magic* of the FFT we can compute $C$ (i.e. every $c_i$) in $O(n \log n)$ time.
Last year on COMS21103...

Let $A$ and $B$ be $(n − 1)$ degree polynomials which can be expressed as...

$$A(x) = \sum_{i=0}^{n-1} a_i x^i \quad \text{and} \quad B(x) = \sum_{i=0}^{n-1} b_i x^i$$

The polynomial $C = A \times B$ can be expressed as...

$$C(x) = \sum_{i=0}^{2n-1} c_i x^i \quad \text{where} \quad c_i = \sum_{j=0}^{i} a_j \times b_{(i−j)}$$

By the *magic* of the FFT we can compute $C$ (i.e. every $c_i$) in $O(n \log n)$ time.
Let $A$ and $B$ be $(n - 1)$ degree polynomials which can be expressed as...

$$A(x) = \sum_{i=0}^{n-1} a_i x^i \quad \text{and} \quad B(x) = \sum_{i=0}^{n-1} b_i x^i$$

The polynomial $C = A \times B$ can be expressed as...

$$C(x) = \sum_{i=0}^{2n-1} c_i x^i \quad \text{where} \quad c_i = \sum_{j=0}^{i} a_j \times b_{i-j}$$

By the *magic* of the FFT we can compute $C$ (i.e. every $c_i$) in $O(n \log n)$ time.
Let $A$ and $B$ be $(n - 1)$ degree polynomials which can be expressed as...

$$A(x) = \sum_{i=0}^{n-1} a_i x^i \quad \text{and} \quad B(x) = \sum_{i=0}^{n-1} b_i x^i$$

The polynomial $C = A \times B$ can be expressed as...

$$C(x) = \sum_{i=0}^{2n-1} c_i x^i \quad \text{where} \quad c_i = \sum_{j=0}^{i} a_j \times b_{i-j}$$

By the magic of the FFT we can compute $C$ (i.e. every $c_i$) in $O(n \log n)$ time.
Last year on COMS21103...

Let $A$ and $B$ be $(n - 1)$ degree polynomials which can be expressed as...

$$A(x) = \sum_{i=0}^{n-1} a_i x^i \quad \text{and} \quad B(x) = \sum_{i=0}^{n-1} b_i x^i$$

The polynomial $C = A \times B$ can be expressed as...

$$C(x) = \sum_{i=0}^{2n-1} c_i x^i \quad \text{where} \quad c_i = \sum_{j=0}^{i} a_j \times b_{i-j}$$

By the *magic* of the FFT we can compute $C$ (i.e. every $c_i$) in $O(n \log n)$ time.

**Hint 1** Let $A = P_d$ and $B = T_d$
Let $A$ and $B$ be $(n - 1)$ degree polynomials which can be expressed as.

$$A(x) = \sum_{i=0}^{n-1} a_i x^i \quad \text{and} \quad B(x) = \sum_{i=0}^{n-1} b_i x^i$$

The polynomial $C = A \times B$ can be expressed as.

$$C(x) = \sum_{i=0}^{2n-1} c_i x^i \quad \text{where} \quad c_i = \sum_{j=0}^{i} P_d[j] T_d[i - j]$$

By the *magic* of the FFT we can compute $C$ (i.e. every $c_i$) in $O(n \log n)$ time.

**Hint 1** Let $A = P_d$ and $B = T_d$
Let $A$ and $B$ be $(n - 1)$ degree polynomials which can be expressed as . . .

\[ A(x) = \sum_{i=0}^{n-1} a_i x^i \quad \text{and} \quad B(x) = \sum_{i=0}^{n-1} b_i x^i \]

By the magic of the FFT we can compute $C$ (i.e. every $c_i$) in $O(n \log n)$ time.

The polynomial $C = A \times B$ can be expressed as . . .

\[ C(x) = \sum_{i=0}^{2n-1} c_i x^i \quad \text{where} \quad c_i = \sum_{j=0}^{i} P_d[j] T_d[i-j] \]

By the \textit{magic} of the FFT we can compute $C$ (i.e. every $c_i$) in $O(n \log n)$ time.

\textbf{Hint 2} Let $A = P_d$ (padded with zeros) and $B = T_d$
Last year on COMS21103...

Let $A$ and $B$ be $(n - 1)$ degree polynomials which can be expressed as...

$$A(x) = \sum_{i=0}^{n-1} a_i x^i \quad \text{and} \quad B(x) = \sum_{i=0}^{n-1} b_i x^i$$

These polynomials can be represented as arrays of length $m$:

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0 \ 0 \ 0 \ 0 \ \cdots \ 0$</td>
<td>$m$</td>
</tr>
</tbody>
</table>

$A[i] = a_i = P_d[i]$ (or be seen as arrays of length $n$) $B[i] = b_i = T_d[n - i]$

The polynomial $C = A \times B$ can be expressed as...

$$C(x) = \sum_{i=0}^{2n-1} c_i x^i \quad \text{where} \quad c_{n-i} = \sum_{j=0}^{n-i} P_d[j] T_d[i+j]$$

$C$ can be represented as an array of length $m - 1$:

<table>
<thead>
<tr>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m - 1$</td>
</tr>
</tbody>
</table>

$C[i] = c_i$

By the magic of the FFT we can compute $C$ (i.e. every $c_i$) in $O(n \log n)$ time.

**Hint 3** Let $A = P_d$ (padded with zeros) and $B = T_d$ (reversed)
Let \( A \) and \( B \) be \((n - 1)\) degree polynomials which can be expressed as

\[
A(x) = \sum_{i=0}^{n-1} a_i x^i \quad \text{and} \quad B(x) = \sum_{i=0}^{n-1} b_i x^i
\]

The polynomial \( C = A \times B \) can be expressed as

\[
C(x) = \sum_{i=0}^{2n-1} c_i x^i \quad \text{where} \quad c_{n-i} = \sum_{j=0}^{n-i} P_d[j] T_d[i+j]
\]

By the *magic* of the FFT we can compute \( C \) (i.e. every \( c_i \)) in \( O(n \log n) \) time.

**Hint 3** Let \( A = P_d \) (padded with zeros) and \( B = T_d \) (reversed) . . . now \( C \) contains \((T_d \otimes P_d)\)
Computing cross-correlations via the FFT

Let $T_\sigma$ be $T$ with all $\sigma$s replaced with 1s and everything else replaced with a 0s

$(P_\sigma$ is defined analogously)

alignment 4

$$T_\sigma$$

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

$$P_\sigma$$

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

\[
(T_\sigma \otimes P_\sigma)[i] = \sum_{j=0}^{m-1} P_\sigma[j] \times T_\sigma[i + j]
\]

is exactly number of matching $d$s at the $i$-th alignment.
Let $T_\sigma$ be $T$ with all $\sigma$s replaced with 1s and everything else replaced with a 0s $(P_\sigma$ is defined analogously)

Alignment 4

<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>n</td>
<td>T_\sigma</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>
</tr>
<tr>
<td></td>
<td>m-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>
<tr>
<td>P_\sigma</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>

$(T_\sigma \otimes P_\sigma)[i] = \sum_{j=0}^{m-1} P_\sigma[j] \times T_\sigma[i+j]$ is exactly number of matching $d$s at the $i$-th alignment.

We can compute $(T_\sigma \otimes P_\sigma)$ in $O(n \log n)$ time via the FFT
Computing cross-correlations via the FFT

Let $T_{\sigma}$ be $T$ with all $\sigma$s replaced with 1s and everything else replaced with a 0s

(P$_{\sigma}$ is defined analogously)

Alignment 4

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

\[
P_{\sigma} = \begin{array}{c}
1 & 0 & 0 & 1 \\
\end{array}
\]

\[
(T_{\sigma} \otimes P_{\sigma})[i] = \sum_{j=0}^{m-1} P_{\sigma}[j] \times T_{\sigma}[i + j]
\]

is exactly number of matching $d$s at the $i$-th alignment.

We can compute $(T_{\sigma} \otimes P_{\sigma})$ in $O(n \log n)$ time via the FFT

i.e after $O(n \log n)$ time we have $(T_d \otimes P_d)[i]$ for every $i$
Computing cross-correlations via the FFT

Let $T_\sigma$ be $T$ with all $\sigma$s replaced with 1s and everything else replaced with a 0s. $(P_\sigma$ is defined analogously)

We can compute $(T_\sigma \otimes P_\sigma)$ in $O(n \log m)$ time via the FFT

$$\sum_{j=0}^{m-1} P_\sigma[j] \times T_\sigma[i + j]$$

is exactly number of matching $d$s at the $i$-th alignment.

We can compute $(T_d \otimes P_d)[i]$ for every $i$
Computing cross-correlations via the FFT

Let $T_\sigma$ be $T$ with all $\sigma$s replaced with 1s and everything else replaced with a 0s

($P_\sigma$ is defined analogously)

alignment 4

```
 0 1 2 3 4 5 6 7 8 9 10 11 12
```

```
T_\sigma
1 0 0 0 1 0 1 1 0 1 0 1 1
```

```
P_\sigma
1 0 0 1
```

$(T_\sigma \otimes P_\sigma)[i] = \sum_{j=0}^{m-1} P_\sigma[j] \times T_\sigma[i + j]$ is exactly number of matching $d$s at the $i$-th alignment.

We can compute $(T_\sigma \otimes P_\sigma)$ in $O(n \log m)$ time via the FFT

i.e after $O(n \log m)$ time we have $(T_d \otimes P_d)[i]$ for every $i$

Clarification: (covered on whiteboard in lecture)
this improvement comes from splitting $T_\sigma$ into
$O(n/m)$ overlapping sections of length $2m$
and using the FFT method once per section
Computing cross-correlations via the FFT

Let $T_\sigma$ be $T$ with all $\sigma$s replaced with 1s and everything else replaced with a 0s.

$(P_\sigma$ is defined analogously)

We can compute $(T_\sigma \otimes P_\sigma)$ in $O(n \log m)$ time via the FFT.

i.e after $O(n \log m)$ time we have $(T_d \otimes P_d)[i]$ for every $i$. 

Alignment 4

\[
T_\sigma = \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\
\end{bmatrix}
\]

\[
P_\sigma = \begin{bmatrix}
1 & 0 & 0 & 1 \\
\end{bmatrix}
\]

\[
(T_\sigma \otimes P_\sigma)[i] = \sum_{j=0}^{m-1} P_\sigma[j] \times T_\sigma[i + j]
\]

is exactly number of matching $d$s at the $i$-th alignment.
Computing cross-correlations via the FFT

Let $T_\sigma$ be $T$ with all $\sigma$s replaced with 1s and everything else replaced with a 0s

$(P_\sigma$ is defined analogously)

alignment 4

\[
\begin{align*}
T_\sigma &= \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1
\end{bmatrix} \\
P_\sigma &= \begin{bmatrix}
1 & 0 & 0 & 1
\end{bmatrix}
\end{align*}
\]

$$(T_\sigma \otimes P_\sigma)[i] = \sum_{j=0}^{m-1} P_\sigma[j] \times T_\sigma[i + j]$$

is exactly number of matching $d$s at the $i$-th alignment.

We can compute $(T_\sigma \otimes P_\sigma)$ in $O(n \log m)$ time via the FFT

i.e after $O(n \log m)$ time we have $(T_d \otimes P_d)[i]$ for every $i$

$(T_\sigma \otimes P_\sigma)$ is called the cross-correlation of $T_\sigma$ and $P_\sigma$
Computing cross-correlations via the FFT

Let $T_\sigma$ be $T$ with all $\sigma$s replaced with 1s and everything else replaced with a 0s

Let $T_\sigma$ be $T$ with all $\sigma$s replaced with 1s and everything else replaced with a 0s

Let $T_\sigma$ be $T$ with all $\sigma$s replaced with 1s and everything else replaced with a 0s

$(P_\sigma$ is defined analogously)

alignment 4

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

$(T_\sigma \otimes P_\sigma)[i] = \sum_{j=0}^{m-1} P_\sigma[j] \times T_\sigma[i + j]$ is exactly number of matching $d$s at the $i$-th alignment.

We can compute $(T_\sigma \otimes P_\sigma)$ in $O(n \log m)$ time via the FFT i.e after $O(n \log m)$ time we have $(T_d \otimes P_d)[i]$ for every $i$

$(T_\sigma \otimes P_\sigma)$ is called the cross-correlation of $T_\sigma$ and $P_\sigma$ it is also very often (but technically incorrectly) called the convolution
Computing cross-correlations via the FFT

Let $T_\sigma$ be $T$ with all $\sigma$s replaced with 1s and everything else replaced with a 0s

$P_\sigma$ is defined analogously

alignment 4

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

$$\begin{array}{cccc}
1 & 0 & 0 & 1 \\
\hline
P_\sigma \\
\end{array}$$

cross-correlations are used a lot in the pattern matching literature

$$(T_\sigma \otimes P_\sigma)[i] = \sum_{j=0}^{m-1} P_\sigma[j] \times T_\sigma[i+j]$$

is exactly number of matching $d$s at the $i$-th alignment.

We can compute $(T_\sigma \otimes P_\sigma)$ in $O(n \log m)$ time via the FFT

i.e after $O(n \log m)$ time we have $(T_d \otimes P_d)[i]$ for every $i$

$(T_\sigma \otimes P_\sigma)$ is called the cross-correlation of $T_\sigma$ and $P_\sigma$

it is also very often (but technically incorrectly) called the convolution
Computing cross-correlations via the FFT

Let $T_{\sigma}$ be $T$ with all $\sigma$s replaced with 1s and everything else replaced with a 0s

$$(P_{\sigma} \text{ is defined analogously)}$$

alignment 4

$$T_{\sigma} = \begin{array}{cccccccccccc}
1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1
\end{array}$$

$$P_{\sigma} = \begin{array}{cccc}
1 & 0 & 0 & 1
\end{array}$$

$$(T_{\sigma} \otimes P_{\sigma})[i] = \sum_{j=0}^{m-1} P_{\sigma}[j] \times T_{\sigma}[i + j]$$

is exactly number of matching $d$s at the $i$-th alignment.

We can compute $(T_{\sigma} \otimes P_{\sigma})$ in $O(n \log n)$ time via the FFT

i.e after $O(n \log n)$ time we have $(T_d \otimes P_d)[i]$ for every $i$

$(T_{\sigma} \otimes P_{\sigma})$ is called the cross-correlation of $T_{\sigma}$ and $P_{\sigma}$

it is also very often (but technically incorrectly) called the convolution
Computing cross-correlations via the FFT

Let $T_\sigma$ be $T$ with all $\sigma$s replaced with $1$s and everything else replaced with a $0$s.

$(P_\sigma$ is defined analogously)

$$T_\sigma \ egin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$P_\sigma \ egin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}$$

is exactly number of matching $d$s at the $i$-th alignment.

We can compute $(T_\sigma \otimes P_\sigma)$ in $O(n \log n)$ time via the FFT

i.e after $O(n \log n)$ time we have $(T_d \otimes P_d)[i]$ for every $i$

$(T_\sigma \otimes P_\sigma)$ is called the cross-correlation of $T_\sigma$ and $P_\sigma$

it is also very often (but technically incorrectly) called the convolution
It’s a small alphabet after all

We have seen 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

$\quad$ (in the example $\Sigma = \{a, b, c, d\}$ so $|\Sigma| = 4$)

Algorithm Summary

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

$\quad$ $\text{Ham}(i) = m - \sum_{\sigma \in \Sigma} (T_\sigma \otimes P_\sigma)[i]$.
It's a small alphabet after all

We have seen 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.

(in the example $\Sigma = \{a, b, c, d\}$ so $|\Sigma| = 4$)

**Algorithm Summary**

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

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

matches involving $\sigma$
It's a small alphabet after all

We have seen 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

(in the example $\Sigma = \{a, b, c, d\}$ so $|\Sigma| = 4$)

Algorithm Summary

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

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

all matches
It's a small alphabet after all

We have seen 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.

(in the example $\Sigma = \{a, b, c, d\}$ so $|\Sigma| = 4$)

**Algorithm Summary**

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

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

mismatches $= m - \text{matches}$
It's a small alphabet after all

We have seen 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.

(in the example $\Sigma = \{a, b, c, d\}$ so $|\Sigma| = 4$)

**Algorithm Summary**

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

$$\text{Ham}(i) = m - \sum_{\sigma \in \Sigma} (T_\sigma \otimes P_\sigma)[i].$$
It's a small alphabet after all

We have seen 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.

(in the example $\Sigma = \{a, b, c, d\}$ so $|\Sigma| = 4$)

**Algorithm Summary**

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

$$\text{Ham}(i) = m - \sum_{\sigma \in \Sigma} (T_\sigma \otimes P_\sigma)[i].$$
It's a small alphabet after all

We have seen 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.

(in the example $\Sigma = \{a, b, c, d\}$ so $|\Sigma| = 4$)

**Algorithm Summary**

Construct $T_\sigma$ and $P_\sigma$ for each symbol $\sigma$ in $\Sigma$ $(O(n|\Sigma|)$ time)

Compute $(T_\sigma \otimes P_\sigma)$ for each symbol $\sigma$ in $\Sigma$ $(O(n|\Sigma| \log m)$ time)

For every $i$, compute,

$$\text{Ham}(i) = m - \sum_{\sigma \in \Sigma} (T_\sigma \otimes P_\sigma)[i].$$
It’s a small alphabet after all

We have seen 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

(in the example $\Sigma = \{a, b, c, d\}$ so $|\Sigma| = 4$)

**Algorithm Summary**

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

Compute $(T_\sigma \otimes P_\sigma)$ for each 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)
It's a small alphabet after all

We have seen 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

(in the example $\Sigma = \{a, b, c, d\}$ so $|\Sigma| = 4$)

Algorithm Summary

Construct $T_\sigma$ and $P_\sigma$ for each symbol $\sigma$ in $\Sigma$ 

(\(O(n|\Sigma|)\) time)

Compute $(T_\sigma \otimes P_\sigma)$ for each symbol $\sigma$ in $\Sigma$

(\(O(n|\Sigma| \log m)\) time)

For every $i$, compute,

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

(\(O(n|\Sigma|)\) time)

This takes $O(n|\Sigma| \log m)$ total time (and $O(n)$ space)
It's a small alphabet after all

We have seen 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.

(in the example $\Sigma = \{a, b, c, d\}$ so $|\Sigma| = 4$)

**Algorithm Summary**

Construct $T_\sigma$ and $P_\sigma$ for each symbol $\sigma$ in $\Sigma$ $(O(n|\Sigma|)$ time)

Compute $(T_\sigma \otimes P_\sigma)$ for each symbol $\sigma$ in $\Sigma$ $(O(n|\Sigma| \log m)$ time)

For every $i$, compute,

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

$(O(n|\Sigma|)$ time)

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

However, $|\Sigma|$ could be as big as $m$...
It’s a small alphabet after all

We have seen 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

(in the example $\Sigma = \{a, b, c, d\}$ so $|\Sigma| = 4$)

Algorithm Summary

Construct $T_\sigma$ and $P_\sigma$ for each symbol $\sigma$ in $\Sigma$  

$(O(n|\Sigma|)$ time)

Compute $(T_\sigma \otimes P_\sigma)$ for each symbol $\sigma$ in $\Sigma$  

$(O(n|\Sigma| \log m)$ time)

For every $i$, compute,

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

$(O(n|\Sigma|)$ time)

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

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

in which case, this is worse than the naive method!
Coping with a large alphabet

We will now see an algorithm which runs in $O(n\sqrt{m \log m})$ time regardless of the alphabet size.
Coping with a large alphabet

We will now see an algorithm which runs in $O(n\sqrt{m \log m})$ time regardless of the alphabet size.

**Definition:** An alphabet symbol is *frequent* if it occurs at least $\sqrt{m}$ times in $P$. 
Coping with a large alphabet

We will now see an algorithm which runs in $O(n \sqrt{m \log m})$ time regardless of the alphabet size.

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

![Diagram showing an example string P with m = 9 and frequent symbols highlighted]
Coping with a large alphabet

We will now see an algorithm which runs in $O(n\sqrt{m \log m})$ time regardless of the alphabet size.

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

<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>$m = 9$</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>
</tr>
</tbody>
</table>

($\sqrt{m} = 3$)
Coping with a large alphabet

We will now see an algorithm which runs in $O(n\sqrt{m \log m})$ time regardless of the alphabet size.

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

The sequence $P$ is

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

with $m = 9$ and $\sqrt{m} = 3$.

$a$ is frequent.
Coping with a large alphabet

We will now see an algorithm which runs in $O(n\sqrt{m\log m})$ time regardless of the alphabet size.

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

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

($\sqrt{m} = 3$)

$a$ is frequent, $b$ is frequent
Coping with a large alphabet

We will now see an algorithm which runs in \( O(n\sqrt{m \log m}) \) time regardless of the alphabet size.

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

\[
P = \text{a b b a c a d b d}
\]

\( m = 9 \quad \text{(} \sqrt{m} = 3 \text{)} \)

\( a \) is frequent, \( b \) is frequent, 
\( c \) and \( d \) are infrequent.
Coping with a large alphabet

We will now see an algorithm which runs in $O(n\sqrt{m \log m})$ time regardless of the alphabet size.

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

<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>$m = 9$</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>
</tr>
</tbody>
</table>

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

**Key idea:** Our algorithm will have two main stages:
Coping with a large alphabet

We will now see an algorithm which runs in $O(n\sqrt{m \log m})$ time regardless of the alphabet size.

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

Key idea: Our algorithm will have two main stages:

- Stage 1 will count all the matches involving frequent symbols (at each alignment of $P$ and $T$)

Here is an example:

$P$: $a$ $b$ $b$ $a$ $c$ $a$ $d$ $b$ $d$

$m = 9$\hspace{1cm}($\sqrt{m} = 3$)

$a$ is frequent, $b$ is frequent
$c$ and $d$ are infrequent
Coping with a large alphabet

We will now see an algorithm which runs in $O(n\sqrt{m \log m})$ time regardless of the alphabet size.

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

```
<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>m = 9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

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

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

**Key idea:** Our algorithm will have two main stages:

**Stage 1** will count all the matches involving *frequent* symbols (at each alignment of $P$ and $T$).

**Stage 2** will count all the matches involving *infrequent* symbols (at each alignment of $P$ and $T$).
Coping with a large alphabet

We will now see an algorithm which runs in $O(n\sqrt{m \log m})$ time regardless of the alphabet size

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

$$m = 9$$

$P$

```
<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>
```

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

**Key idea:** Our algorithm will have two main stages:

- **Stage 1** will count all the matches involving *frequent* symbols
  (at each alignment of $P$ and $T$)

- **Stage 2** will count all the matches involving *infrequent* symbols
  (at each alignment of $P$ and $T$)

The total number of matches is the sum of the matches from **Stage 1** and **Stage 2**
Coping with a large alphabet

We will now see an algorithm which runs in $O(n\sqrt{m \log m})$ time \textit{regardless of the alphabet size}.

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

$$m = 9$$

<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>

$a$ is \textit{frequent}, $b$ is \textit{frequent},

$c$ and $d$ are \textit{infrequent}.

**Key idea:** Our algorithm will have two main stages:

**Stage 1** will count all the matches involving \textit{frequent} symbols

(at each alignment of $P$ and $T$)

**Stage 2** will count all the matches involving \textit{infrequent} symbols

(at each alignment of $P$ and $T$)

The total number of matches is the sum of the matches from **Stage 1** and **Stage 2**.
The frequent/infrequent symbols trick

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

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<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>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$m = 9$

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

$a$ is *frequent*, $b$ is *frequent*
$c$ and $d$ are *infrequent*
The frequent/infrequent symbols trick

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

![Example sequence and frequency count]

Stage 1: For each alignment $i$, count the number of matches involving frequent symbols:
**The frequent/infrequent symbols trick**

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

Stage 1: For each alignment \( i \), count the number of matches involving **frequent** symbols:

Consider each frequent symbol \( \sigma \in \Sigma \) separately and compute \((T_\sigma \otimes P_\sigma)\)
The frequent/infrequent symbols trick

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

Stage 1: For each alignment $i$, count the number of matches involving frequent symbols:

Consider each frequent symbol $\sigma \in \Sigma$ separately and compute $(T_\sigma \otimes P_\sigma)$ in $O(n \log m)$ time (per symbol $\sigma$) using cross-correlations.
The frequent/infrequent symbols trick

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

Stage 1: For each alignment $i$, count the number of matches involving frequent symbols:

Consider each frequent symbol $\sigma \in \Sigma$ separately and compute $(T_\sigma \otimes P_\sigma)$ in $O(n \log m)$ time (per symbol $\sigma$) using cross-correlations

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

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

![Example alignment](image)

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

**Stage 1:** For each alignment $i$, count the number of matches involving frequent symbols:

Consider each frequent symbol $\sigma \in \Sigma$ separately and compute $(T_\sigma \otimes P_\sigma)$

in $O(n \log m)$ time (per symbol $\sigma$) 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:** An alphabet symbol is *frequent* if it occurs at least $\sqrt{m}$ times in $P$.

Stage 1: For each alignment $i$, count the number of matches involving frequent symbols:

Consider each frequent symbol $\sigma \in \Sigma$ separately and compute $(T\sigma \otimes P\sigma)$ in $O(n \log m)$ time (per symbol $\sigma$) 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:** An alphabet symbol is *frequent* if it occurs at least $\sqrt{m}$ times in $P$.

Stage 1: For each alignment $i$, count the number of matches involving frequent symbols:

Consider each frequent symbol $\sigma \in \Sigma$ separately and compute $(T_\sigma \otimes P_\sigma)$

in $O(n \log m)$ time (per symbol $\sigma$) 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:** An alphabet symbol is *frequent* if it occurs at least $\sqrt{m}$ times in $P$.

Stage 1: For each alignment $i$, count the number of matches involving frequent symbols:

Consider each frequent symbol $\sigma \in \Sigma$ separately and compute $(T_\sigma \otimes P_\sigma)$ in $O(n \log m)$ time (per symbol $\sigma$) 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!
The frequent/infrequent symbols trick

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

Stage 1: For each alignment \( i \), count the number of matches involving frequent symbols:

Consider each frequent symbol \( \sigma \in \Sigma \) separately and compute \( (T_\sigma \otimes P_\sigma) \)

in \( O(n \log m) \) time (per symbol \( \sigma \)) 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...

\( (\sqrt{m} + 1) \sqrt{m} > m \)  \text{ Contradiction!}

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

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

Stage 1: For each alignment \( i \), count the number of matches involving frequent symbols:

Consider each frequent symbol \( \sigma \in \Sigma \) separately and compute \( (T_{\sigma} \otimes P_\sigma) \)
in \( O(n \log m) \) time (per symbol \( \sigma \)) using cross-correlations.

How many frequent symbols can there be?

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

\[ \text{each occurs at least } \sqrt{m} \text{ times...} \quad (\sqrt{m} + 1)\sqrt{m} > m \quad \text{Contradiction!} \]

so there are at most \( \sqrt{m} \) frequent symbols

So Stage 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 $P$.

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>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>
<td></td>
</tr>
<tr>
<td>$m$</td>
<td>9</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>
</tbody>
</table>
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*

Stage 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

Stage 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

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

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

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

| \( T \)  | 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 |

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

Construct an array \( A \) of length \((n - m + 1)\) - which initially contains 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 $P$.

| $T$ | 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 |

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

Every symbol is either frequent or infrequent

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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*.

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

<table>
<thead>
<tr>
<th>$P$</th>
<th>a</th>
<th>b</th>
<th>b</th>
<th>a</th>
<th>c</th>
<th>a</th>
<th>d</th>
<th>b</th>
<th>d</th>
</tr>
</thead>
</table>

| $A$  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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 \( P \).

Every symbol is either frequent or infrequent

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

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

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

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

<table>
<thead>
<tr>
<th>Stage 2: Count all matches involving infrequent symbols.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct an array $A$ of length $(n - m + 1)$ - which initially contains all zeros</td>
</tr>
<tr>
<td>Make a single pass through $T$…</td>
</tr>
<tr>
<td>For each character $T[k]$, (where $0 \leq k &lt; n$)</td>
</tr>
<tr>
<td>If $T[k]$ is infrequent…</td>
</tr>
</tbody>
</table>
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.

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

Stage 2: Count all matches involving infrequent symbols.

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

---

Stage 2: Count all matches involving infrequent symbols.

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

![Diagram]

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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 $P$.

Every symbol is either frequent or infrequent

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

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

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

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

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

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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 $(k - j) < 0$)

**Example:**

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

$k - j < 0$
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

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

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

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

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

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

---

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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 arrays]

Every symbol is either frequent or infrequent

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

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

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

```
T = [X d X X 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]
```

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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 $P$.

Stage 2: Count all matches involving infrequent symbols.

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

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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 their frequencies]

- **Stage 2:** Count all matches involving infrequent symbols.

Construct an array \( A \) of length \( (n - m + 1) \) - which initially contains 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
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*

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

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

$A = [1, 0, 0, 0, 0, 0, 0, 0, 0]$
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>a</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>
<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>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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 \( P \).

\[
\begin{vmatrix}
T & \text{d} & \text{b} & \text{x} & \text{c} & \text{c} & \text{c} & \text{d} & \text{a} & \text{d} & \text{c} & \text{d} & \text{c} & \text{d} & \text{a} & \text{c} \\
P & \text{a} & \text{b} & \text{b} & \text{a} & \text{c} & \text{a} & \text{d} & \text{b} & \text{d} \\
A & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{vmatrix}
\]

Every symbol is either frequent or infrequent

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

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

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

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

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

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.

<table>
<thead>
<tr>
<th>$T$</th>
<th>a</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>c</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>1</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>
<td></td>
</tr>
</tbody>
</table>

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

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

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

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

<table>
<thead>
<tr>
<th>$T$</th>
<th>d</th>
<th>b</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>
</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>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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*

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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*.

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

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

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

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

<table>
<thead>
<tr>
<th>T</th>
<th>d</th>
<th>b</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>
</tr>
<tr>
<td>A</td>
<td>1</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>
</tr>
</tbody>
</table>

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

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

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

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

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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**

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

Stage 2: Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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></th>
<th>d</th>
<th>b</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><strong>T</strong></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><strong>P</strong></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>
</tr>
<tr>
<td><strong>A</strong></td>
<td>1</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>

Every symbol is either frequent or infrequent.

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

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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 $P$.

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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*.

$T$ =

```
A d b b 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
```

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

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

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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 \(\text{except when } (k - j) < 0\)
The infrequent/frequent symbols trick

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

\[
\begin{array}{cccccccccc}
T & d & b & c & c & c & d & a & d & c & d & a & c \\
| & a & b & b & a & c & a & d & b & d |
\end{array}
\]

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

Every symbol is either frequent or infrequent.

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

Stage 2: Count all matches involving infrequent symbols.

Construct an array \( A \) of length \((n - m + 1)\) - which initially contains 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](image)

Every symbol is either *frequent* or *infrequent*

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

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

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

![Diagram](image)

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

$A = [1, 1, 1, 0, 0, 0, 0, 0]$

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

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

<table>
<thead>
<tr>
<th>$T$</th>
<th>dna</th>
<th>abc</th>
<th>ccd</th>
<th>dad</th>
<th>cdc</th>
<th>ddc</th>
<th>aac</th>
<th>cdd</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$</td>
<td>ab</td>
<td>bba</td>
<td>aca</td>
<td>dad</td>
<td>bbd</td>
<td>ddd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A$</td>
<td>1110000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Stage 2: Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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>( 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>
</tr>
<tr>
<td>( A )</td>
<td>1</td>
<td>1</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>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>$T$</th>
<th>d</th>
<th>b</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>
</tr>
<tr>
<td>$A$</td>
<td>1</td>
<td>1</td>
<td>1</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>

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

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

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

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

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

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

Stage 2: Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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 $P$.

![Image of symbols with arrows indicating frequent and infrequent status]

Every symbol is either frequent or infrequent

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

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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*

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

Construct an array \( A \) of length \((n - m + 1)\) - which initially contains 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 \( (\text{except when } (k - j) < 0) \)
**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.

---

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

Construct an array \( A \) of length \((n - m + 1)\) - which initially contains 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 $P$.

$\hspace{1cm} k = 7 \hspace{1cm} j = 6 \hspace{1cm} k - j = 1$

Every symbol is either *frequent* or *infrequent*

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

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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>$a$</th>
<th>$d$</th>
<th>$b$</th>
<th>$b$</th>
<th>$c$</th>
<th>$c$</th>
<th>$c$</th>
<th>$c$</th>
<th>$d$</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>
<td></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>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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 of symbols]

Every symbol is either frequent or infrequent

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

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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 \(\text{except when } (k - j) < 0\)
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 b c c c d d c d d a c</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>a b b a c a d b d</td>
</tr>
<tr>
<td>A</td>
<td>1 2 1 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

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

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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*.

<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>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>
</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>
<td></td>
</tr>
</tbody>
</table>

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

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

Stage 2: Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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 arrays]

Every symbol is either *frequent* or *infrequent*

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

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

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

Stage 2: Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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*

<table>
<thead>
<tr>
<th>$T$</th>
<th>d b c c c d d c d c d a c</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$</td>
<td>a b b a c a d b d</td>
</tr>
<tr>
<td>$A$</td>
<td>1 3 1 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 3 1 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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 $P$.

Stage 2: Count all matches involving infrequent symbols.

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

Fact $A[i]$ is the number of matches at alignment $i$ involving an infrequent symbol
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*

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

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

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

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

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

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

How quick is Stage 2?

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

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

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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 $(k - j) < 0$)

$O(n)$ 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

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

Stage 2: Count all matches involving infrequent symbols.

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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 Stage 2?

$O(n)$ time

(Each list has length less than $\sqrt{m}$)

Store a list for each infrequent symbol

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

$T$  
\[d \ b \ c \ c \ c \ d \ c \ d \ c \ d \ c \]

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

<table>
<thead>
<tr>
<th>T</th>
<th>d b x c c c d x d c d c d x c</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>a b b a c a d b d</td>
</tr>
<tr>
<td>A</td>
<td>1 3 1 2 0 2 1 1</td>
</tr>
</tbody>
</table>

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

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

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

$$O(n\sqrt{m})$$ time

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 $(k - j) < 0$)

$O(n)$ time

How quick is Stage 2?
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*

---

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

Construct an array $A$ of length $(n - m + 1)$ - which initially contains 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 $P$.

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

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

Every symbol is either frequent or infrequent

$a$ is *frequent*, $b$ is *frequent*
$c$ and $d$ are *infrequent*
Pattern matching with mismatches: putting it all together

Algorithm summary
Pattern matching with mismatches: putting it all together

Algorithm summary

Stage 0: Classify each symbol as frequent or infrequent - $O(m \log m)$ time
Pattern matching with mismatches: putting it all together

Algorithm summary

**Stage 0:** Classify each symbol as frequent or infrequent - $O(m \log m)$ time

**Stage 1:** Count all matches involving frequent symbols. - $O(n \sqrt{m} \log m)$ time
Pattern matching with mismatches: putting it all together

Algorithm summary

**Stage 0:** Classify each symbol as frequent or infrequent - $O(m \log m)$ time

**Stage 1:** Count all matches involving frequent symbols. - $O(n \sqrt{m} \log m)$ time

**Stage 2:** Count all matches involving infrequent symbols. - $O(n \sqrt{m})$ time
Algorithm summary

**Stage 0:** Classify each symbol as frequent or infrequent - $O(m \log m)$ time

**Stage 1:** Count all matches involving frequent symbols. - $O(n \sqrt{m} \log m)$ time

**Stage 2:** Count all matches involving infrequent symbols. - $O(n \sqrt{m})$ time
Pattern matching with mismatches: putting it all together

Algorithm summary

**Stage 0:** Classify each symbol as frequent or infrequent - \(O(m \log m)\) time

**Stage 1:** Count all matches involving frequent symbols. - \(O(n \sqrt{m} \log m)\) time

**Stage 2:** Count all matches involving infrequent symbols. - \(O(n \sqrt{m})\) time

*at any alignment \(i\)*

*the number of mismatches is just \(m\) minus the total number of matches*
Pattern matching with mismatches: putting it all together

Algorithm summary

**Stage 0:** Classify each symbol as frequent or infrequent - $O(m \log m)$ time

**Stage 1:** Count all matches involving frequent symbols. - $O(n \sqrt{m} \log m)$ time

**Stage 2:** Count all matches involving infrequent symbols. - $O(n \sqrt{m})$ time

at any alignment $i$
the number of mismatches is just $m$ minus the total number of matches

Overall, we obtain a time complexity of $O(n \sqrt{m} \log m)$. 
Algorithm summary

**Stage 0:** Classify each symbol as frequent or infrequent - $O(m \log m)$ time

**Stage 1:** Count all matches involving frequent symbols. - $O(n \sqrt{m} \log m)$ time

**Stage 2:** Count all matches involving infrequent symbols. - $O(n \sqrt{m})$ time

At any alignment $i$
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 **Stage 1** takes longer than **Stage 2**...
Pattern matching with mismatches: putting it all together

Algorithm summary

Stage 0: Classify each symbol as frequent or infrequent - $O(m \log m)$ time

Stage 1: Count all matches involving frequent symbols. - $O(n \sqrt{m \log m})$ time

Stage 2: Count all matches involving infrequent symbols. - $O(n \sqrt{m \log m})$ time

At any alignment $i$
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.)
Conclusion

**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>
<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>
</tr>
</tbody>
</table>

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

A naive algorithm for this problem takes $O(nm)$ time

We have seen two alternative algorithms:

- One algorithm takes $O(n|\Sigma| \log m)$ time (where $|\Sigma|$ is the alphabet size)
- The other algorithm takes $O(n\sqrt{m \log m})$ time (regardless of the alphabet size)