String matching - contd

$Y : \text{string of length } m$

$X : \text{pattern of length } n \leq m$

$Y(i) = Y_i, Y_{i+1}, \ldots, Y_{m+i-1}$ \hspace{1cm} (substring of length $n$ starting at $i$)

$X(i) = X_1, X_2, \ldots, X_i$

$A \subseteq B$ : $A$ is a suffix of $B$

$f(i) = \max_{j < i} X(j) \subseteq X(i)$

Example: $X = 10101111$, $X(1) \subseteq X(5)$

$X(3) \subseteq X(5)$

$\Rightarrow f(5) = 3$
<table>
<thead>
<tr>
<th></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>X</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>f(i)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Y $</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>a</td>
<td>$</td>
<td>$</td>
</tr>
<tr>
<td></td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>$</td>
<td>$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>$</td>
<td>$</td>
<td></td>
</tr>
</tbody>
</table>

1. Longest prefix of $X(5)$ that matches the suffix of $abaaba$ or $X(5)$. $a$

2. Longest prefix of $X(5)$ that matches a suffix of $X(5) = abaaba = X(3)$
   i.e. $f(5) = 3$
Recall that in the potential function based amortised analysis, the amortised work in step $i$

\[ A_i = W_i + \Delta(\phi) \]

\[ \uparrow \text{actual work} \quad \text{change in potential} \]

Total amortised work in potential

\[ = \text{Total work} + \phi_n - \phi_0 \]

Potential function for the string matching = extent of partial match

i.e. if we have match up to $X(i)$

Compare $X_{i+1}$ with $Y_{j+i}$

Case $X_{i+1} = Y_{j+i}$

Amortised cost

\[ = 1 < \text{actual cost } n \]

Comparison

Case $X_{i+1} \neq Y_{j+i}$

Amortised cost $\leq 0$
We keep track of the total # of comparison w.r.t. a position in Y.

The amortized cost that can be charged to any position \( Y \leq 2 \) (since mismatch only reduces potential)

\[ \Rightarrow \text{Total # comparisons} \leq 2 \cdot m \]

Cost of computing shift function

<table>
<thead>
<tr>
<th>( f(i) )</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(1) )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( f(1) + \cdots + f(i) \) = \( j+1 \) and \( x_{i+1} = x_{j+1} \),

Inductively suppose we have computed

\( f(1) + \cdots + f(i) \) then \( f(i+1) = j+1 \).
else if \( f(f(i)) = j \) and \( X_{i+1} = X_{j+1} \),

\[
\begin{align*}
&f(i+1) = j + 1 \\
&i = i + 1
\end{align*}
\]

It is similar to the string matching algorithm itself. Using a similar analysis, we can bound the running-time to \( 2n \).

So total time for Knuth-Morris-Pratt (KMP)
is \( O(n+m) \).