Knuth-Morris-Pratt Algorithm

Ray Seyfarth
School of Computing
University of Southern Mississippi

January 31, 2006
Outline

Finite Automata
  Accepting a Suffix Automaton as a Table
  Suffixes

Knuth-Morris-Pratt Algorithm
A finite automaton $M = (Q, q_0, A, \Sigma, \delta)$, where

- $Q$ is a finite set of states
- $q_0 \in Q$ is the start state for $M$
- $A \subseteq Q$ is a set of accepting (or final) states
- $\Sigma$ is a finite set call the input alphabet
- $\delta : Q \times \Sigma \rightarrow Q$ is the transition function.

In the diagram above

- 0 is the start state
- 1 is the final state
- $\Sigma = \{a, b\}$
- $\delta(0, a) = 1, \delta(0, b) = 0, \delta(1, a) = 0, \delta(1, b) = 0$
Accepting abbaaaba

\[
\begin{align*}
\delta(0, a) &= 1 \\
\delta(1, b) &= 0 \\
\delta(0, b) &= 0 \\
\delta(0, a) &= 1 \\
\delta(1, a) &= 0 \\
\delta(0, a) &= 1 \\
\delta(1, b) &= 0 \\
\delta(0, a) &= 1
\end{align*}
\]

A simpler way to indicate this is using \( \phi(abbaaaba) = 1 \).

\( \phi(\epsilon) = q_0 \) and \( \phi(wa) = \delta(\phi(w), a) \)
Accepting Strings Ending with ababaca

Start with automaton for ababaca
Accepting Strings Ending with ababaca

If we start with b we should start over
Accepting Strings Ending with abbaca

a in state 1 should lead to state 1 again
Accepting Strings Ending with abbaca

b in state 2 forces a restart
Accepting Strings Ending with abbaca

a in state 3 returns to state 1
Accepting Strings Ending with abbaca

b in state 4 forces a restart
Accepting Strings Ending with abbaca

a is state 5 sends us back to state 1
Accepting Strings Ending with abbaca

b in state 5 sends us to state 4 - using abab suffix
Accepting Strings Ending with abbaca

Failure to get c in state 6 forces a restart
Accepting Strings Ending with abbaca

a in state 7 sends us to state 1
Accepting Strings Ending with abbaca

b in state 7 sends us to state 2
Accepting Strings Ending with abbaca

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
Finding a Substring

The pattern is found starting with index 4.

Search could continue past that occurrence to find multiple matches.
Use the notation $x \sqsubseteq y$ to mean $x$ is a suffix of $y$.

Given pattern $P$ with individual letters $P[1], P[2], \ldots, P[m]$

Use the notation $P_k$ to mean $P[1]P[2] \cdots P[k]$

Define the suffix function $\sigma : \Sigma^* \rightarrow \{0, 1, \ldots, m\}$

$\sigma(x) = \max\{k : P_k \sqsubseteq x\}$

$\sigma(\epsilon) = 0, \sigma(abc) = 0, \sigma(abcab) = 2$

The states of the automaton are $Q = \{0, 1, \ldots, m\}$.

$\delta(q, a) = \sigma(P_qa)$

$\delta(7, b) = 2, P_7b = ababacab$
Suffix Theorem

If $\phi$ is the final-state function of the string-matching automaton for the pattern $P$ and $T[1..n]$ is the input text for the automaton, then $\phi(T_i) = \sigma(T_i)$ for $i = 0, 1, \ldots, n$. 
Transition Function Computation

Compute-Transition-Function($P, \Sigma$)

1. $m \leftarrow \text{length}[P]$
2. for $q \leftarrow 0$ to $m$
3. for each $a \in \Sigma$
4. $k \leftarrow \min(m + 1, q + 2)$
5. repeat
6. $k \leftarrow k + 1$
7. until $P_k \sqcup P_qa$
8. $\delta(q, a) \leftarrow k$
9. end-for
10. end-for
11. return $\delta$
Outline

Finite Automata
   Accepting a Suffix
   Automaton as a Table
   Suffixes

Knuth-Morris-Pratt Algorithm
KMP Highlights

- Previous automaton creation was $O(m^3\Sigma)$
- This can be improved to $O(m\Sigma)$
- However KMP algorithm requires $O(m)$ preparation time
- The rest of KMP requires $O(n)$ time for a total of $O(n + m)$.
- KMP precomputes an array $\pi$ which is used to replace $\delta$.
- $\pi$ has $m$ entries and $\delta$ has $m\Sigma$. 
Given pattern $P$ with individual letters $P[1], P[2], \ldots, P[m]$

- $\pi : \{1, 2, \ldots, m\} \rightarrow \{0, 1, \ldots, m - 1\}$
- $\pi[q] = \max\{k : k < q \text{ and } P_k \sqsupseteq P_q\}$
- $\pi[q]$ is the longest prefix of $P$ which is a proper suffix of $P_q$
Prefix Function Computation

<table>
<thead>
<tr>
<th>i</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>
</tr>
</thead>
<tbody>
<tr>
<td>$P[i]$</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>a</td>
</tr>
<tr>
<td>$\pi[i]$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Compute-Prefix-Function($P$)

1. $m \leftarrow \text{length}[P]$
2. $\pi[1] \leftarrow 0$
3. $k \leftarrow 0$
4. for $q \leftarrow 2$ to $m$
5. while $k > 0$ and $P[k + 1] \neq P[q]$ do $k \leftarrow \pi[k]$
6. if $P[k + 1] = P[k]$ then $k \leftarrow k + 1$
7. $\pi[q] \leftarrow k$
8. end-for
9. return $\pi$