String Matching

• T = text
• P = pattern
• n = text length
• m = pattern length.
• Z = alphabet size.
Naïve Method

Complexity?
Knuth-Morris-Pratt Algorithm

Idea of Jumping

(a)

(b)
Jump Table

<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>π[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>

KMP Algorithm

\[
\text{KMP-MATCHER}(T, P) \\
1 \quad n \leftarrow \text{length}[T] \\
2 \quad m \leftarrow \text{length}[P] \\
3 \quad \pi \leftarrow \text{Compute-Prefix-Function}(P) \\
4 \quad q \leftarrow 0 \\
5 \quad \text{for } i \leftarrow 1 \text{ to } n \\
6 \quad \quad \text{do while } q > 0 \text{ and } P[q + 1] \neq T[i] \\
7 \quad \quad \quad \text{do } q \leftarrow \pi[q] \\
8 \quad \quad \text{if } P[q + 1] = T[i] \\
9 \quad \quad \quad \text{then } q \leftarrow q + 1 \\
10 \quad \quad \text{if } q = m \\
11 \quad \quad \quad \text{then print "Pattern occurs with shift" } i - m \\
12 \quad \quad \quad q \leftarrow \pi[q]
\]
Computing Jumps

- Jumps can be computed by preprocessing the pattern, by comparing the pattern with itself.

**Complexity**

- The running time for `COMPUTE-PREFIX_FUNCTION` is:
  - $O(m)$
- The total jump cannot exceed $(m+n)$, thus complexity is $O(m+n)$. 
State Machine

FSM for Detecting String Which Ends with Even 1s

Figure 34.5 A simple two-state finite automaton with state set $Q = \{0, 1\}$, start state $q_0 = 0$, and input alphabet $\Sigma = \{a, b\}$. (a) A tabular representation of the transition function $\delta$. (b) An equivalent state-transition diagram. State 1 is the only accepting state (shown blackened). Directed edges represent transitions. For example, the edge from state 1 to state 0 labeled $b$ indicates $\delta(1, b) = 0$. This automaton accepts those strings that end in an odd number of $a$'s. More precisely, a string $x$ is accepted if and only if $x = ayz$, where $y = a$ or $y$ ends with $a$, and $z = u^*$, where $u$ is odd. For example, the sequence of states this automaton enters for input abaaa (including the start state) is $(0, 1, 0, 1, 0, 1)$, and so it accepts this input. For input abbaa, the sequence of states is $(0, 1, 0, 0, 1, 0)$, and so it rejects this input.
State Transition Diagram for String Matching

Complexity

- Fastest Compute State Transition is:
  - $O(mz)$
- Total WC running time is:
  - $O(n+mz)$
Boyer-Moore Algorithm (1976)

- Comparing from the Right to Left:
  - in the pattern, each time, there is a mismatch, see how many position the pattern can be shifted left.

- More Look ahead in the Preprocessing
  - bring into consideration the character that caused the mismatch while considering what to do next.
Example

A STRING SEARCHING EXAMPLE CONSISTING OF
STING
STING
STING
STING
STING
STING
STING

Complexity

- Boyer-Moore string search algorithm never uses more than M+N character comparisons, and uses about N/M steps of the alphabet is not small and the pattern is not long.
Rabin-Karp Method
(1980)
Computing Hash Value

14152 = (31415 - 3 \cdot 10000) \cdot 10 + 2 \pmod{13}
= (7 - 3) \cdot 10 + 2 \pmod{13}
= 8 \pmod{13}

RK Algorithm

```
RABIN-KARP-MATCHER(T, P, d, q)
1  n ← length(T)
2  m ← length(P)
3  h ← d^{m-1} \mod q
4  p ← 0
5  t₀ ← 0
6  for i ← 1 to m
7      do p ← (dp + P[i]) \mod q
8      t₀ ← (t₀ + T[i]) \mod q
9  for s ← 0 to n - m
10     do if p = tₙ
11        then if P[1..m] = T[s+1..s+m]
12           then “Pattern occurs with shift” s
13        if s < n - m
14          then tₙ₊₁ ← (d(tₙ - T[s+1]h) + T[s + m + 1]) \mod q
```

- Radix is d. The prime is q.
Complexity

- In the worst case the running time is $O((n-m+1)m)$.
  - (case $T=a^n$ and $P=a^m$)
  - Each evaluation after the first one is $O(1)$ in text.
- Average Case Complexity
  - Only one match in most cases $O(1)$
  - Thus running time is $O(n+m)$.