Factor Oracle for Machine Improvisation

Jaime Arias

Université de Bordeaux, LaBRI, UMR 5800
Inria - Bordeaux Sud-Ouest

August 2016

Preliminaries

Preliminaries

Word

A word s is a finite sequence $s = s_1 s_2 \dots s_m$ of length $|s| = m$ on a finite alphabet Σ.

$$s = \boxed{\texttt{a} \mid \texttt{b} \mid \texttt{b} \mid \texttt{c} \mid \texttt{a} \mid \texttt{b} \mid \texttt{c} \mid \texttt{d} \mid \texttt{a} \mid \texttt{b} \mid \texttt{c}}$$

Factor

A word $x \in \Sigma^*$ is a factor of s if and only if s can be written $s = uxv$ with $u, v \in \Sigma^*$. Given integers i, j where $1 \leq i \leq j \leq m$, we denote a *factor* of s as $s[i \dots j] = s_i s_{i+1} \dots s_j$.
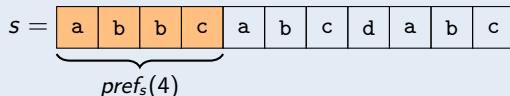
$$s = \boxed{\texttt{a} \mid \texttt{b} \mid \texttt{b} \mid \texttt{c} \mid \texttt{a} \mid \texttt{b} \mid \texttt{c} \mid \texttt{d} \mid \texttt{a} \mid \texttt{b} \mid \texttt{c}}$$
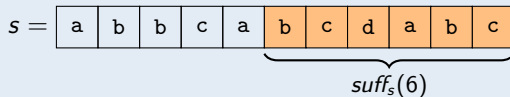
$$\underbrace{\qquad}_{s[3, 5]}$$

Preliminaries

Prefix

A factor x of s is a prefix of s if $s = xu$ with $u \in \Sigma^*$. The ith *prefix* of s, denoted $pref_s(i)$, is the prefix $s[1 \dots i]$.

$$s = \boxed{\underbrace{\begin{array}{|c|c|c|c|}\hline a & b & b & c \\\hline\end{array}}_{pref_s(4)} \begin{array}{|c|c|c|c|c|c|c|}\hline a & b & c & d & a & b & c \\\hline\end{array}}$$

Suffix
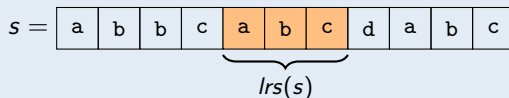
A factor x of s is a suffix of s if $s = ux$ with $u \in \Sigma^*$. The ith *suffix* of s, denoted $suff_s(i)$, is the suffix $s[i \dots m]$.

$$s = \begin{array}{|c|c|c|c|c|}\hline a & b & b & c & a \\\hline\end{array} \underbrace{\begin{array}{|c|c|c|c|c|c|}\hline b & c & d & a & b & c \\\hline\end{array}}_{suff_s(6)}$$

Preliminaries

Longest Repeated Suffix (LRS)

A factor x of s is the longest repeated suffix of s if x is a suffix of s and $|x|$ is maximal.

$$s = \boxed{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|} \hline a & b & b & c & a & b & c & d & a & b & c \\ \hline \end{array}}$$

$lrs(s)$

Factor Oracle

Factor Oracle

Overview

Forward Links

- - - -> Suffix Links
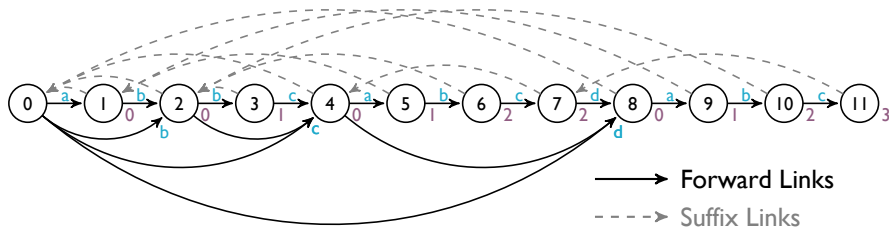
Factor Oracle

The factor oracle of a word s of length m is a *deterministic finite automaton* (Q, q_0, F, δ) where $Q = \{0, 1, \dots, m\}$ is the set of states, $q_0 = 0$ is the starting state, $F = Q$ is the set of terminal states and δ is the transition function.
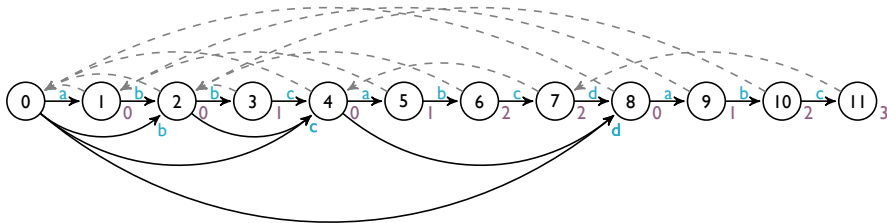
→ Forward Links
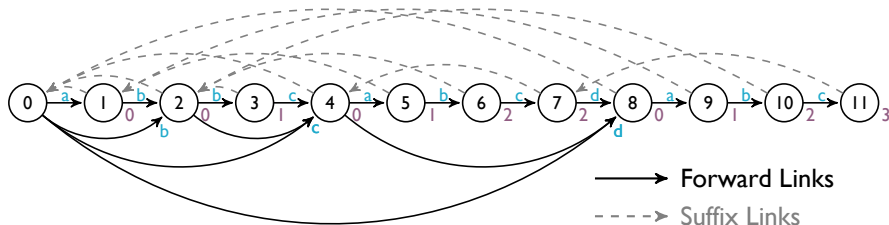
----→ Suffix Links

Suffix Link

The suffix link of a state i of the factor oracle of a word s, is equal to the state in which the *longest repeated suffix (lrs)* of $s[1 \dots i]$ is recognized.
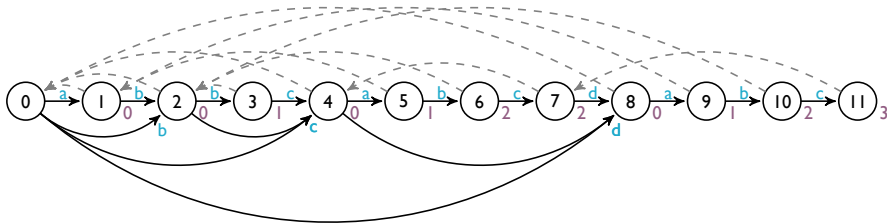
Factor Oracle

Overview

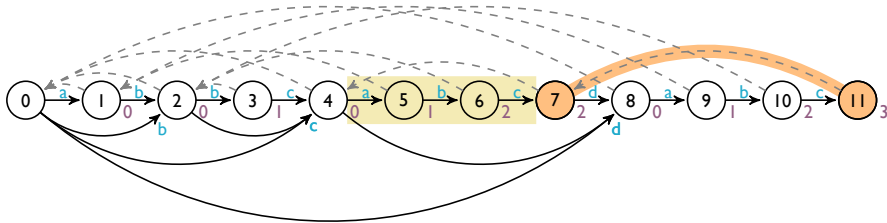## Suffix Links

- $s = $ abbcabcdabc

Factor Oracle

Overview

Suffix Links

- $s = \text{abbc}\textcolor{red}{\text{abc}}\text{d}\textcolor{red}{\text{abc}}$
- $lrs(s) = \text{abc}$

Factor Oracle

Overview

Suffix Links

- $s = \texttt{abbcabcdabc}$
- $lrs(s) = \texttt{abc}$
- $S(11) = 7$

Factor Oracle

Algorithm

Algorithm 1 Construction of a Factor Oracle

1: **function** FactorOracle($p = p_1 p_2 \ldots p_m$)
2:   Create a new oracle P with an initial state 0
3:   $S_P(0) \leftarrow -1$
4:   **for** $i \leftarrow 1, m$ **do**
5:     $Oracle(p = p_1 p_2 \ldots p_i) \leftarrow \text{AddLetter}(Oracle(p = p_1 p_2 \ldots p_{i-1}), p_i)$
6:   **end for**
7:   **return** $Oracle(p = p_1 p_2 \ldots p_m)$
8: **end function**

Factor Oracle

Algorithm

Algorithm 1 Construction of a Factor Oracle

1: **function** FactorOracle($p = p_1 p_2 \dots p_m$)
2:     Create a new oracle P with an initial state 0
3:     $S_P(0) \leftarrow -1$
4:     **for** $i \leftarrow 1, m$ **do**
5:         $Oracle(p = p_1 p_2 \dots p_i) \leftarrow$ AddLetter($Oracle(p = p_1 p_2 \dots p_{i-1}), p_i$)
6:     **end for**
7:     **return** $Oracle(p = p_1 p_2 \dots p_m)$
8: **end function**

$$p = \boxed{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|} \hline a & b & b & c & a & b & c & d & a & b & c \\ \hline \end{array}}$$

Factor Oracle

Algorithm

Algorithm 1 Construction of a Factor Oracle

1: **function** FactorOracle($p = p_1 p_2 \dots p_m$)
2:     Create a new oracle P with an initial state 0
3:     $S_P(0) \leftarrow -1$
4:     **for** $i \leftarrow 1, m$ **do**
5:         $Oracle(p = p_1 p_2 \dots p_i) \leftarrow \text{AddLetter}(Oracle(p = p_1 p_2 \dots p_{i-1}), p_i)$
6:     **end for**
7:     **return** $Oracle(p = p_1 p_2 \dots p_m)$
8: **end function**

$$p = \boxed{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|} \hline a & b & b & c & a & b & c & d & a & b & c \\ \hline \end{array}}$$

(0)

Factor Oracle
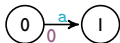
Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ ▷ $\delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     …
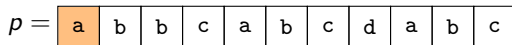7: **end function**

---

slide 1

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \dots p_m$), σ)
2:    Create state $m + 1$
3:    Create a new transition from m to $m + 1$ labeled by σ ▷ $\delta(m, \sigma) = m + 1$
4:    $k \leftarrow S_p(m)$
5:    $\pi_1 \leftarrow m$
6:    …
7: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c | $m = 0$
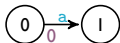
$(0)_0$

slide 2

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 ... p_m), \sigma$)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ ▷ $\delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     ...
7: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c | $m = 0$

slide 3

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     …
7: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c |
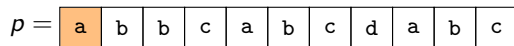
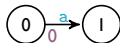$m = 0$ $k = -1$ $\pi_1 = 0$

slide 4

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     …
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ $\triangleright \delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     …
9: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c | $m = 0$ $k = -1 \pi_1 = 0$

$(0) \xrightarrow{\ a\ }_{0} (1)$
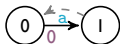
slide 5

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

```
 1: function AddLetter(Oracle(p = p_1, p_2 ... p_m), σ)
 2:     ...
 3:     if k = −1 then
 4:         S_{pσ} ← 0
 5:         lrs_{pσ} ← 0
 6:     else
 7:         S_{pσ} ← state that leads the transition from k by σ
 8:         lrs_{pσ} ← LengthCommonSuffix(π_1, S(m + 1) − 1) + 1
 9:     end if
10:     ...
11: end function
```

$p = $ | a | b | b | c | a | b | c | d | a | b | c | $m = 0 \quad k = -1 \pi_1 = 0$

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

```
 1: function AddLetter(Oracle(p = p₁, p₂ ... pₘ), σ)
 2:     …
 3:     if k = −1 then
 4:         S_{pσ} ← 0
 5:         lrs_{pσ} ← 0
 6:     else
 7:         S_{pσ} ← state that leads the transition from k by σ
 8:         lrs_{pσ} ← LengthCommonSuffix(π₁, S(m + 1) − 1) + 1
 9:     end if
10:     …
11: end function
```

$p = $ | a | b | b | c | a | b | c | d | a | b | c | $m = 0$ $k = -1 \pi_1 = 0$

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

```
 1: function AddLetter(Oracle(p = p₁, p₂ ... pₘ), σ)
 2:     ...
 3:     if k = −1 then
 4:         S_{pσ} ← 0
 5:         lrs_{pσ} ← 0
 6:     else
 7:         S_{pσ} ← state that leads the transition from k by σ
 8:         lrs_{pσ} ← LengthCommonSuffix(π₁, S(m + 1) − 1) + 1
 9:     end if
10:     ...
11: end function
```

$$p = \boxed{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|} \hline a & b & b & c & a & b & c & d & a & b & c \\ \hline \end{array}} \qquad m = 0 \quad k = -1 \pi_1 = 0$$

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ ▷ $\delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     …
7: **end function**

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 1$

slide 9

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \dots p_m$), σ)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     …
7: **end function**

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 1$

slide 10

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
2:    Create state $m + 1$
3:    Create a new transition from m to $m + 1$ labeled by σ ▷ $\delta(m, \sigma) = m + 1$
4:    $k \leftarrow S_p(m)$
5:    $\pi_1 \leftarrow m$
6:    ...
7: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c |

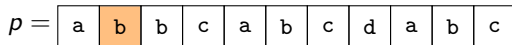$m = 1 \quad k = 0 \quad \pi_1 = 1$

slide 11

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \dots p_m$), σ)
2:     ...
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ $\triangleright \delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     ...
9: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c | $m = 1$ $k = 0$ $\pi_1 = 1$
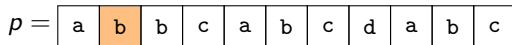

slide 12

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \ldots p_m$), σ)
2:     …
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ $\triangleright \ \delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     …
9: **end function**

$$p = \boxed{\text{a}\ \ \text{b}\ \ \text{b}\ \ \text{c}\ \ \text{a}\ \ \text{b}\ \ \text{c}\ \ \text{d}\ \ \text{a}\ \ \text{b}\ \ \text{c}}$$
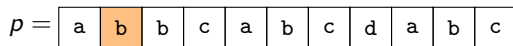
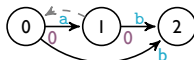$$m = 1 \quad k = 0 \quad \pi_1 = 1$$

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     …
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ $\triangleright \delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     …
9: **end function**

$$p = \boxed{\text{a} \; \text{b} \; \text{b} \; \text{c} \; \text{a} \; \text{b} \; \text{c} \; \text{d} \; \text{a} \; \text{b} \; \text{c}} \qquad m = 1 \quad k = 0 \quad \pi_1 = 0$$
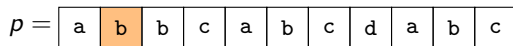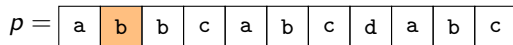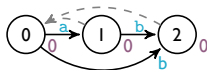
Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \ldots p_m$), σ)
2:     …
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ $\triangleright \delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     …
9: **end function**

$p = $

a	b	b	c	a	b	c	d	a	b	c

$m = 1 \quad k = -1 \, \pi_1 = 0$

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

```
 1: function AddLetter(Oracle(p = p₁, p₂ ... pₘ), σ)
 2:     …
 3:     if k = −1 then
 4:         S_{pσ} ← 0
 5:         lrs_{pσ} ← 0
 6:     else
 7:         S_{pσ} ← state that leads the transition from k by σ
 8:         lrs_{pσ} ← LengthCommonSuffix(π₁, S(m + 1) − 1) + 1
 9:     end if
10:     …
11: end function
```

$p = $ | a | b | b | c | a | b | c | d | a | b | c | $m = 1$ $k = -1$ $\pi_1 = 0$
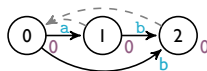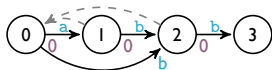
Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
2:     …
3:     **if** $k = -1$ **then**
4:         $S_{p\sigma} \leftarrow 0$
5:         $lrs_{p\sigma} \leftarrow 0$
6:     **else**
7:         $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
8:         $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
9:     **end if**
10:    …
11: **end function**

$$p = \boxed{a \mid b \mid b \mid c \mid a \mid b \mid c \mid d \mid a \mid b \mid c} \qquad m = 1 \quad k = -1 \; \pi_1 = 0$$

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

```
 1: function AddLetter(Oracle(p = p_1, p_2 ... p_m), σ)
 2:     ...
 3:     if k = −1 then
 4:         S_pσ ← 0
 5:         lrs_pσ ← 0
 6:     else
 7:         S_pσ ← state that leads the transition from k by σ
 8:         lrs_pσ ← LengthCommonSuffix(π_1, S(m + 1) − 1) + 1
 9:     end if
10:     ...
11: end function
```

$p = $

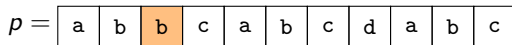| a	b	b	c	a	b	c	d	a	b	c

$m = 1 \quad k = -1 \, \pi_1 = 0$

Factor Oracle

Algorithm

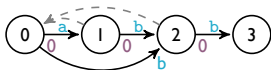**Algorithm 2** Incremental update of Factor Oracle

---

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \dots p_m$), σ)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     ...
7: **end function**

$p = $

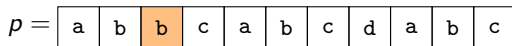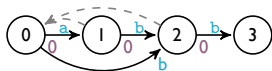| a	b	b	c	a	b	c	d	a	b	c

$m = 2$

slide 19

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     ...
7: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c |
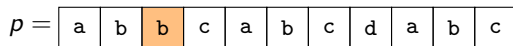
$m = 2$

slide 20

Factor Oracle
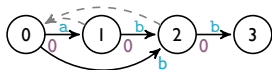
Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \dots p_m$), σ)
2: Create state $m + 1$
3: Create a new transition from m to $m + 1$ labeled by σ ▷ $\delta(m, \sigma) = m + 1$
4: $k \leftarrow S_p(m)$
5: $\pi_1 \leftarrow m$
6: ...
7: **end function**

$$p = \boxed{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|} \text{a} & \text{b} & \text{b} & \text{c} & \text{a} & \text{b} & \text{c} & \text{d} & \text{a} & \text{b} & \text{c} \end{array}}$$

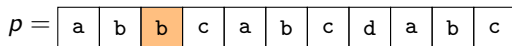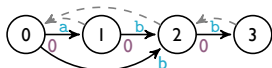$m = 2 \quad k = 0 \quad \pi_1 = 2$

slide 21

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \dots p_m$), σ)
2:     …
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ $\triangleright \delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     …
9: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c |

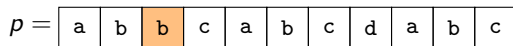$m = 2 \quad k = 0 \quad \pi_1 = 2$

slide 22

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
2:     …
3:     **if** $k = -1$ **then**
4:         $S_{p\sigma} \leftarrow 0$
5:         $lrs_{p\sigma} \leftarrow 0$
6:     **else**
7:         $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
8:         $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
9:     **end if**
10:     …
11: **end function**

$p = \boxed{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|} a & b & b & c & a & b & c & d & a & b & c \end{array}}$ $m = 2 \quad k = 0 \quad \pi_1 = 2$

slide 23

Factor Oracle

Algorithm

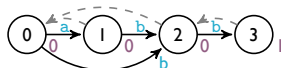**Algorithm 2** Incremental update of Factor Oracle

```
 1: function AddLetter(Oracle(p = p₁, p₂ ... pₘ), σ)
 2:     …
 3:     if k = −1 then
 4:         S_{pσ} ← 0
 5:         lrs_{pσ} ← 0
 6:     else
 7:         S_{pσ} ← state that leads the transition from k by σ
 8:         lrs_{pσ} ← LengthCommonSuffix(π₁, S(m + 1) − 1) + 1
 9:     end if
10:     …
11: end function
```

$p = $ | a | b | b | c | a | b | c | d | a | b | c | $m = 2 \quad k = 0 \quad \pi_1 = 2$
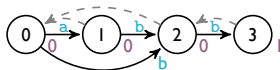
Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     …
3:     **if** $k = -1$ **then**
4:         $S_{p\sigma} \leftarrow 0$
5:         $lrs_{p\sigma} \leftarrow 0$
6:     **else**
7:         $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
8:         $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m + 1) - 1$) + 1
9:     **end if**
10:     …
11: **end function**

$p = $

a	b	b	c	a	b	c	d	a	b	c

$m = 2 \quad k = 0 \quad \pi_1 = 2$

$$lcs(2, 1) = 0$$

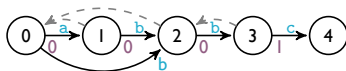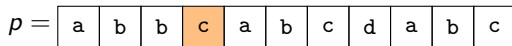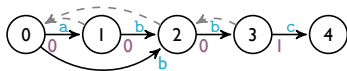slide 25

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ ▷ $\delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     …
7: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c | $m = 3$

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \ldots p_m$), σ)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     …
7: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c |

$m = 3$

slide 27

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     …
7: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c |

$m = 3 \quad k = 2 \quad \pi_1 = 3$
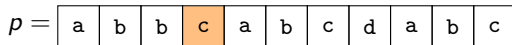
slide 28

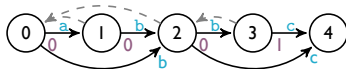# Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \ldots p_m$), σ)
2:     ...
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ ▷ $\delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     ...
9: **end function**

$$p = \boxed{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|} \hline a & b & b & c & a & b & c & d & a & b & c \\ \hline \end{array}}$$

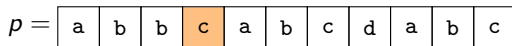$m = 3 \quad k = 2 \quad \pi_1 = 3$

slide 29
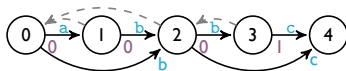
Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     …
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ ▷ $\delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     …
9: **end function**

$p = \boxed{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|} \hline a & b & b & c & a & b & c & d & a & b & c \\ \hline \end{array}}$ $m = 3 \quad k = 2 \quad \pi_1 = 3$
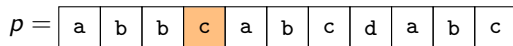

slide 30
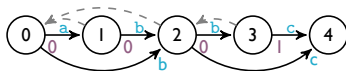
Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \dots p_m$), σ)
2:     …
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ $\triangleright \delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     …
9: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c |

$m = 3$ $k = 2$ $\pi_1 = 2$
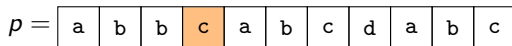
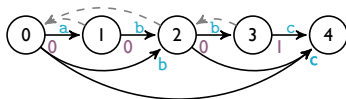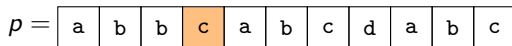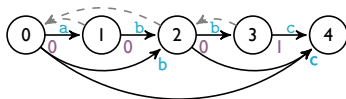slide 31

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
2:    ...
3:    **while** $k > -1$ and there is no transition from k by σ **do**
4:       Create a new transition from k to $m + 1$ by σ $\triangleright \delta(k, \sigma) = m + 1$
5:       $\pi_1 \leftarrow k$
6:       $k \leftarrow S_p(k)$
7:    **end while**
8:    ...
9: **end function**

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 3 \quad k = 0 \quad \pi_1 = 2$

slide 32

Jaime Arias - Equipe PoSET, Inria Bordeaux Sud-Ouest, LaBRI   *Factor Oracle for Machine Improvisation*

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 ... p_m$), σ)
2:     ...
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ ▷ $\delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     ...
9: **end function**

$p = \boxed{a\ |\ b\ |\ b\ |\ c\ |\ a\ |\ b\ |\ c\ |\ d\ |\ a\ |\ b\ |\ c}$ $m = 3$ $k = 0$ $\pi_1 = 2$
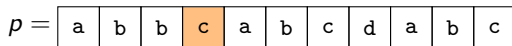

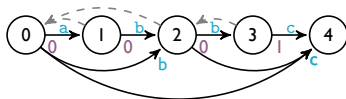slide 33

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
2: ...
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ $\triangleright \delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8: ...
9: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c |
                      $m = 3$ $k = 0$ $\pi_1 = 0$
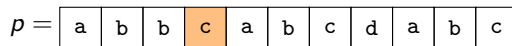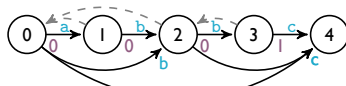

slide 34

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \dots p_m$), σ)
2:     …
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ $\triangleright \delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     …
9: **end function**

$$p = \boxed{\text{a} \mid \text{b} \mid \text{b} \mid \text{c} \mid \text{a} \mid \text{b} \mid \text{c} \mid \text{d} \mid \text{a} \mid \text{b} \mid \text{c}} \qquad m = 3 \quad k = -1 \, \pi_1 = 0$$
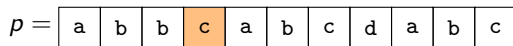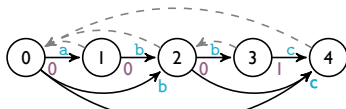

slide 35

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
2:     …
3:     **if** $k = -1$ **then**
4:         $S_{p\sigma} \leftarrow 0$
5:         $lrs_{p\sigma} \leftarrow 0$
6:     **else**
7:         $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
8:         $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
9:     **end if**
10:    …
11: **end function**

$$p = \boxed{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|} \hline a & b & b & c & a & b & c & d & a & b & c \\ \hline \end{array}}$$

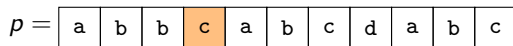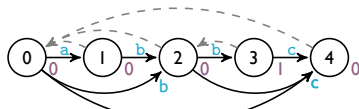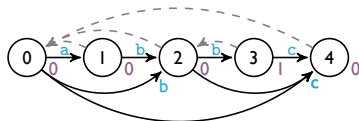$m = 3 \quad k = -1 \, \pi_1 = 0$

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     …
3:     **if** $k = -1$ **then**
4:         $S_{p\sigma} \leftarrow 0$
5:         $lrs_{p\sigma} \leftarrow 0$
6:     **else**
7:         $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
8:         $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
9:     **end if**
10:     …
11: **end function**

$p = $

a	b	b	c	a	b	c	d	a	b	c

$m = 3 \quad k = -1 \pi_1 = 0$

Jaime Arias - Equipe PoSET, Inria Bordeaux Sud-Ouest, LaBRI   *Factor Oracle for Machine Improvisation*

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

```
1: function AddLetter(Oracle(p = p₁, p₂ ... pₘ), σ)
2:     ...
3:     if k = −1 then
4:         S_pσ ← 0
5:         lrs_pσ ← 0
6:     else
7:         S_pσ ← state that leads the transition from k by σ
8:         lrs_pσ ← LengthCommonSuffix(π₁, S(m + 1) − 1) + 1
9:     end if
10:    ...
11: end function
```


$$p = \boxed{a \mid b \mid b \mid c \mid a \mid b \mid c \mid d \mid a \mid b \mid c} \qquad m = 3 \quad k = -1 \, \pi_1 = 0$$

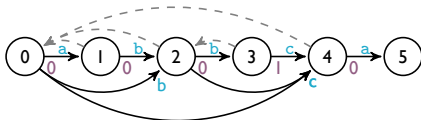# Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     ...
7: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c |

$m = 4$

slide 39
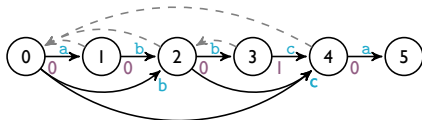
Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \dots p_m$), σ)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     ...
7: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c | $m = 4$
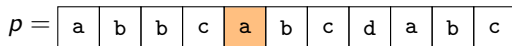

slide 40

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     ...
7: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c |

$m = 4$ $k = 0$ $\pi_1 = 4$

slide 41

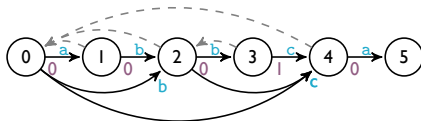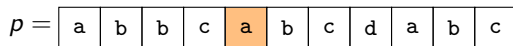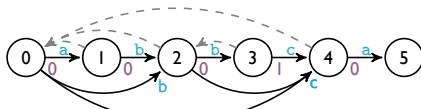# Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     …
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ $\triangleright \delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     …
9: **end function**

$p = $

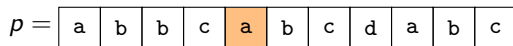| a	b	b	c	a	b	c	d	a	b	c

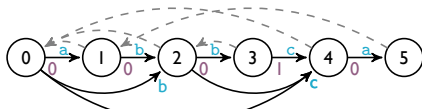$m = 4 \quad k = 0 \quad \pi_1 = 4$

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     …
3:     **if** $k = -1$ **then**
4:         $S_{p\sigma} \leftarrow 0$
5:         $lrs_{p\sigma} \leftarrow 0$
6:     **else**
7:         $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
8:         $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
9:     **end if**
10:    …
11: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c |
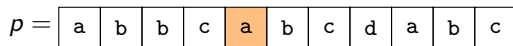
$m = 4 \quad k = 0 \quad \pi_1 = 4$
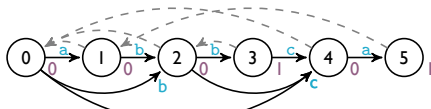
Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
2:     …
3:     **if** $k = -1$ **then**
4:         $S_{p\sigma} \leftarrow 0$
5:         $lrs_{p\sigma} \leftarrow 0$
6:     **else**
7:         $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
8:         $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
9:     **end if**
10:    …
11: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c |

$m = 4$ $k = 0$ $\pi_1 = 4$

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

```
1: function AddLetter(Oracle(p = p₁, p₂ ... pₘ), σ)
2:     …
3:     if k = −1 then
4:         Spσ ← 0
5:         lrspσ ← 0
6:     else
7:         Spσ ← state that leads the transition from k by σ
8:         lrspσ ← LengthCommonSuffix(π₁, S(m + 1) − 1) + 1
9:     end if
10:    …
11: end function
```

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     …
3:     **if** $k = -1$ **then**
4:         $S_{p\sigma} \leftarrow 0$
5:         $lrs_{p\sigma} \leftarrow 0$
6:     **else**
7:         $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
8:         $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix$(\pi_1, S(m + 1) - 1) + 1$
9:     **end if**
10:    …
11: **end function**

$$p = \boxed{a \mid b \mid b \mid c \mid a \mid b \mid c \mid d \mid a \mid b \mid c}$$

$m = 4 \quad k = 0 \quad \pi_1 = 4$

$$lcs(4, 0) = 0$$
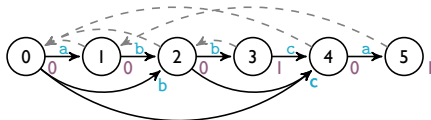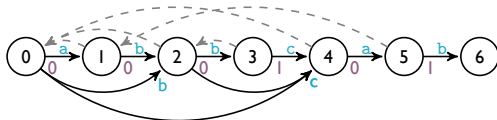
Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
2:      Create state $m + 1$
3:      Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:      $k \leftarrow S_p(m)$
5:      $\pi_1 \leftarrow m$
6:      ...
7: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c |
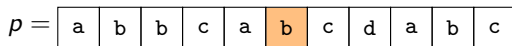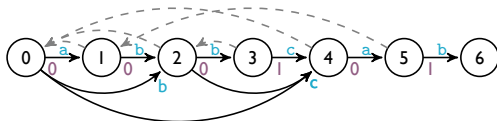
$m = 5$

slide 46

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \dots p_m$), σ)
2:   Create state $m + 1$
3:   Create a new transition from m to $m + 1$ labeled by σ ▷ $\delta(m, \sigma) = m + 1$
4:   $k \leftarrow S_p(m)$
5:   $\pi_1 \leftarrow m$
6:   …
7: **end function**

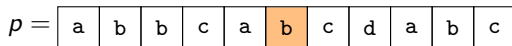$p =$ | a | b | b | c | a | b | c | d | a | b | c | $m = 5$

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \ldots p_m$), σ)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ ▷ $\delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     ...
7: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c |

$m = 5 \quad k = 1 \quad \pi_1 = 5$

slide 48

Factor Oracle
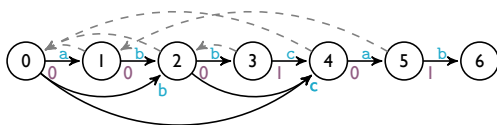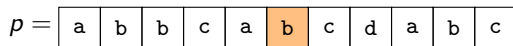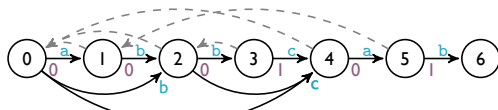
Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     …
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ $\triangleright \delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     …
9: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c |

$m = 5$ $k = 1$ $\pi_1 = 5$

slide 49

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

```
 1: function AddLetter(Oracle(p = p₁, p₂ ... pₘ), σ)
 2:     …
 3:     if k = −1 then
 4:         S_pσ ← 0
 5:         lrs_pσ ← 0
 6:     else
 7:         S_pσ ← state that leads the transition from k by σ
 8:         lrs_pσ ← LengthCommonSuffix(π₁, S(m + 1) − 1) + 1
 9:     end if
10:     …
11: end function
```

$p = \boxed{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|} a & b & b & c & a & b & c & d & a & b & c \end{array}}$
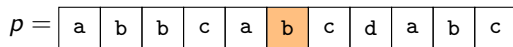
$m = 5 \quad k = 1 \quad \pi_1 = 5$

Factor Oracle
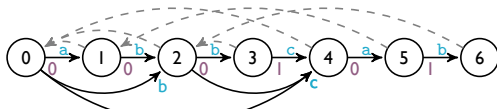
Algorithm

Algorithm 2 Incremental update of Factor Oracle

```
 1: function AddLetter(Oracle(p = p₁, p₂ ... pₘ), σ)
 2:     ...
 3:     if k = -1 then
 4:         S_pσ ← 0
 5:         lrs_pσ ← 0
 6:     else
 7:         S_pσ ← state that leads the transition from k by σ
 8:         lrs_pσ ← LengthCommonSuffix(π₁, S(m + 1) − 1) + 1
 9:     end if
10:     ...
11: end function
```


$p = $ | a | b | b | c | a | b | c | d | a | b | c |
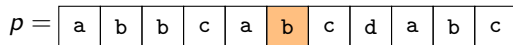
$m = 5 \quad k = 1 \quad \pi_1 = 5$
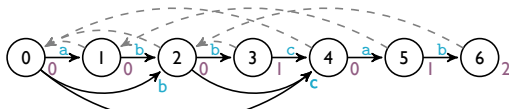
Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
2:     ...
3:     **if** $k = -1$ **then**
4:         $S_{p\sigma} \leftarrow 0$
5:         $lrs_{p\sigma} \leftarrow 0$
6:     **else**
7:         $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
8:         $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
9:     **end if**
10:    ...
11: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c |

$m = 5 \quad k = 1 \quad \pi_1 = 5$

$lcs(5, 1) = 1$

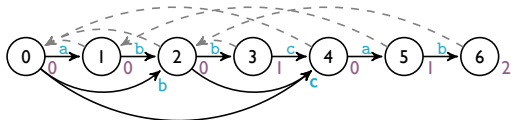slide 52

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     ...
7: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c | $m = 6$
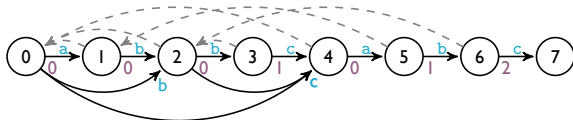
Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     …
7: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c | $m = 6$

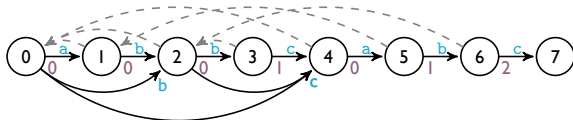slide 54

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     ...
7: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c | $m = 6$ $k = 2$ $\pi_1 = 6$

slide 55

Factor Oracle

Algorithm

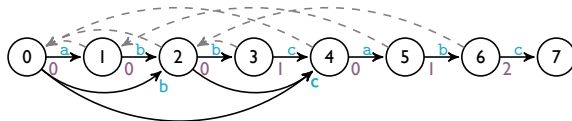**Algorithm 2** Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \ldots p_m$), σ)
2:     …
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ $\triangleright \delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     …
9: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c | $m = 6$ $k = 2$ $\pi_1 = 6$
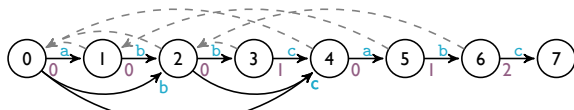

slide 56

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:   …
3:   **if** $k = -1$ **then**
4:    $S_{p\sigma} \leftarrow 0$
5:    $lrs_{p\sigma} \leftarrow 0$
6:   **else**
7:    $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
8:    $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
9:   **end if**
10:   …
11: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c |

$m = 6 \quad k = 2 \quad \pi_1 = 6$
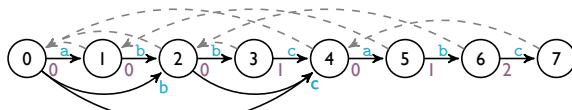
Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
2:     …
3:     **if** $k = -1$ **then**
4:         $S_{p\sigma} \leftarrow 0$
5:         $lrs_{p\sigma} \leftarrow 0$
6:     **else**
7:         $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
8:         $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
9:     **end if**
10:    …
11: **end function**

$$p = \boxed{a \mid b \mid b \mid c \mid a \mid b \mid \fbox{c} \mid d \mid a \mid b \mid c} \qquad m = 6 \quad k = 2 \quad \pi_1 = 6$$

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

```
1:  function AddLetter(Oracle(p = p₁, p₂ ... pₘ), σ)
2:      ...
3:      if k = −1 then
4:          S_{pσ} ← 0
5:          lrs_{pσ} ← 0
6:      else
7:          S_{pσ} ← state that leads the transition from k by σ
8:          lrs_{pσ} ← LengthCommonSuffix(π₁, S(m + 1) − 1) + 1
9:      end if
10:     ...
11: end function
```


$$p = \boxed{a \mid b \mid b \mid c \mid a \mid b \mid c \mid d \mid a \mid b \mid c}$$

$m = 6 \quad k = 2 \quad \pi_1 = 6$
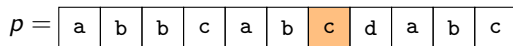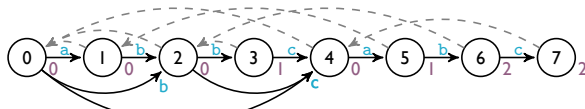
$lcs(6, 3) = 1$

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:      Create state $m + 1$
3:      Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:      $k \leftarrow S_p(m)$
5:      $\pi_1 \leftarrow m$
6:      ...
7: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c | $m = 7$

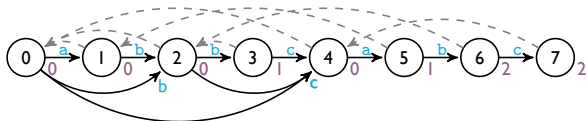slide 60

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     ...
7: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c | $m = 7$
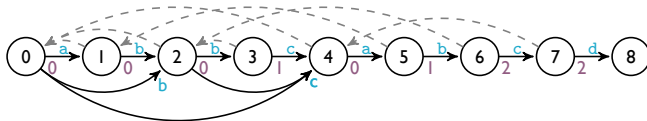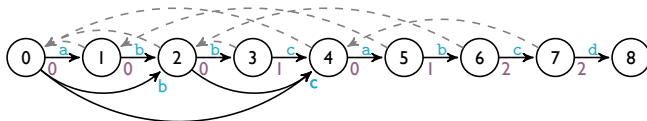
Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     …
7: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c |

$m = 7$ $k = 4$ $\pi_1 = 7$

slide 62

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

```
1: function AddLetter(Oracle(p = p₁, p₂ ... pₘ), σ)
2:     …
3:     while k > −1 and there is no transition from k by σ do
4:         Create a new transition from k to m + 1 by σ          ▷ δ(k, σ) = m + 1
5:         π₁ ← k
6:         k ← Sₚ(k)
7:     end while
8:     …
9: end function
```

$$p = \boxed{\text{a} \mid \text{b} \mid \text{b} \mid \text{c} \mid \text{a} \mid \text{b} \mid \text{c} \mid \text{d} \mid \text{a} \mid \text{b} \mid \text{c}} \qquad m = 7 \quad k = 4 \quad \pi_1 = 7$$
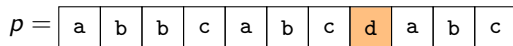

slide 63
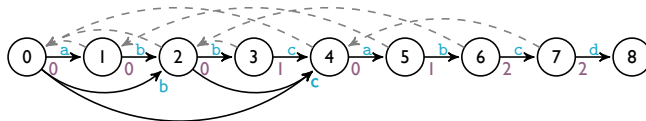
Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 ... p_m), \sigma$)
2:     ...
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ $\triangleright \delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     ...
9: **end function**

$$p = \boxed{a \;\; b \;\; b \;\; c \;\; a \;\; b \;\; c \;\; \underline{d} \;\; a \;\; b \;\; c} \qquad m = 7 \quad k = 4 \quad \pi_1 = 7$$
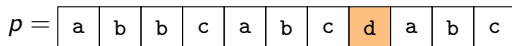

slide 64

Factor Oracle
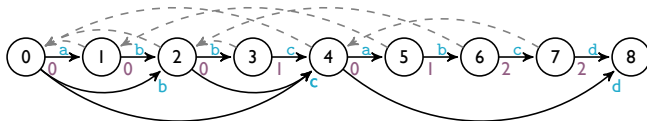
Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 ... p_m$), σ)
2:     ...
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ $\triangleright\ \delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     ...
9: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c |

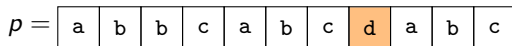$m = 7$ $k = 4$ $\pi_1 = 4$

slide 65
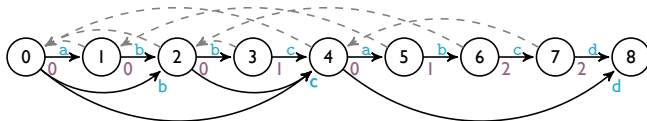
Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
2:     …
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ ▷ $\delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     …
9: **end function**

$$p = \boxed{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|} a & b & b & c & a & b & c & d & a & b & c \end{array}}$$

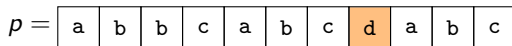$m = 7 \quad k = 0 \quad \pi_1 = 4$

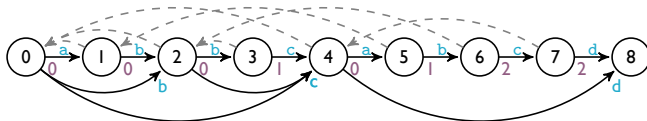slide 66

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
2:      ...
3:      **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ $\triangleright \delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:      **end while**
8:      ...
9: **end function**

$$p = \boxed{\text{a} \mid \text{b} \mid \text{b} \mid \text{c} \mid \text{a} \mid \text{b} \mid \text{c} \mid \text{d} \mid \text{a} \mid \text{b} \mid \text{c}}$$

$m = 7 \quad k = 0 \quad \pi_1 = 4$

slide 67

Factor Oracle

Algorithm

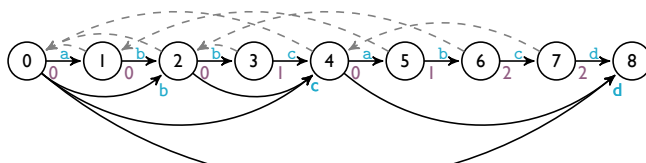**Algorithm 2** Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \dots p_m$), σ)
2:     …
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ $\triangleright \delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     …
9: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c |

$m = 7$ $k = 0$ $\pi_1 = 0$
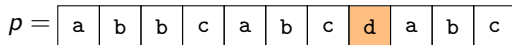
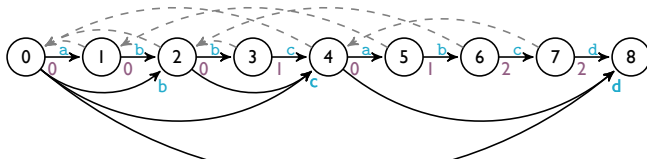slide 68

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \dots p_m$), σ)
2:     ...
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ ▷ $\delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     ...
9: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c |
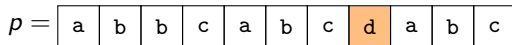
$m = 7$ $k = -1 \pi_1 = 0$
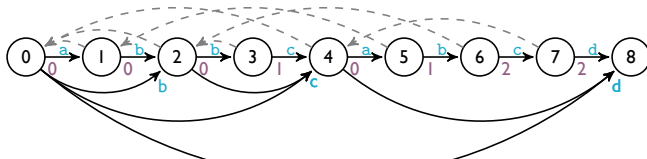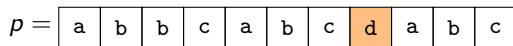

slide 69

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     …
3:     **if** $k = -1$ **then**
4:         $S_{p\sigma} \leftarrow 0$
5:         $lrs_{p\sigma} \leftarrow 0$
6:     **else**
7:         $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
8:         $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
9:     **end if**
10:    …
11: **end function**

$p = \boxed{\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|} a & b & b & c & a & b & c & d & a & b & c \end{array}}$ $\quad m = 7 \quad k = -1 \, \pi_1 = 0$

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

```
 1: function AddLetter(Oracle(p = p₁, p₂ ... pₘ), σ)
 2:     …
 3:     if k = −1 then
 4:         S_{pσ} ← 0
 5:         lrs_{pσ} ← 0
 6:     else
 7:         S_{pσ} ← state that leads the transition from k by σ
 8:         lrs_{pσ} ← LengthCommonSuffix(π₁, S(m + 1) − 1) + 1
 9:     end if
10:     …
11: end function
```


$$p = \boxed{a \mid b \mid b \mid c \mid a \mid b \mid c \mid d \mid a \mid b \mid c} \qquad m = 7 \quad k = -1 \,\pi_1 = 0$$

Factor Oracle
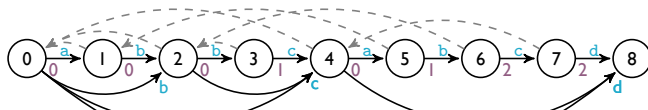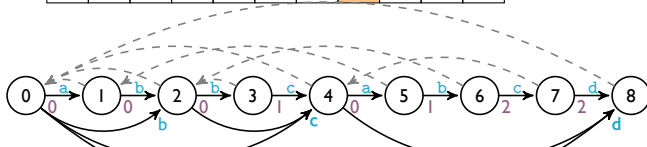
Algorithm

Algorithm 2 Incremental update of Factor Oracle

```
 1: function AddLetter(Oracle(p = p_1, p_2 ... p_m), σ)
 2:     ...
 3:     if k = -1 then
 4:         S_{pσ} ← 0
 5:         lrs_{pσ} ← 0
 6:     else
 7:         S_{pσ} ← state that leads the transition from k by σ
 8:         lrs_{pσ} ← LengthCommonSuffix(π_1, S(m + 1) - 1) + 1
 9:     end if
10:     ...
11: end function
```

$$p = \boxed{a \mid b \mid b \mid c \mid a \mid b \mid c \mid d \mid a \mid b \mid c} \qquad m = 7 \quad k = -1 \, \pi_1 = 0$$
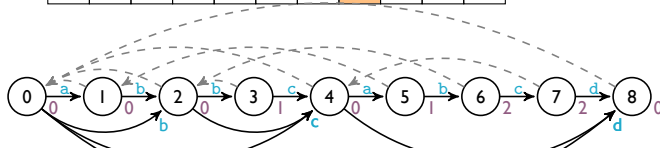
Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     …
7: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c | $m = 8$

slide 73

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ ▷ $\delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
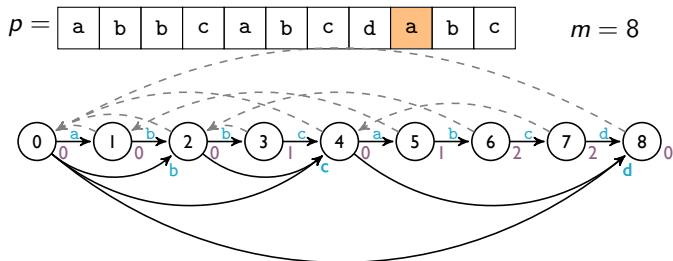6:   …
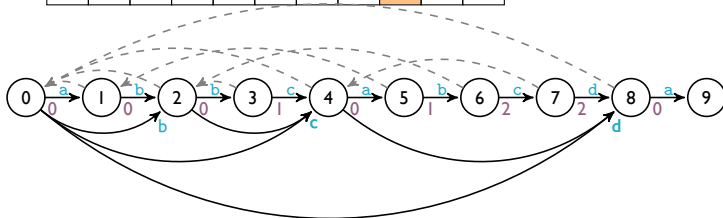7: **end function**

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
2:      Create state $m + 1$
3:      Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:      $k \leftarrow S_p(m)$
5:      $\pi_1 \leftarrow m$
6:      ...
7: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c |

$m = 8$ $k = 0$ $\pi_1 = 8$

slide 75

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \ldots p_m$), σ)
2:     …
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ ▷ $\delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     …
9: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c |

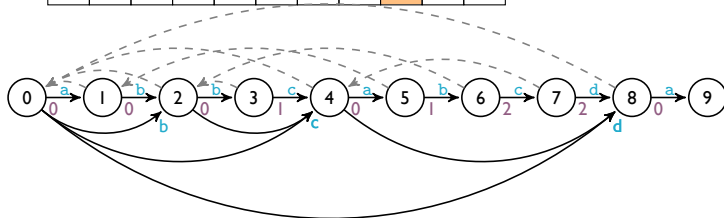$m = 8$ $k = 0$ $\pi_1 = 8$

slide 76

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     …
3:     **if** $k = -1$ **then**
4:         $S_{p\sigma} \leftarrow 0$
5:         $lrs_{p\sigma} \leftarrow 0$
6:     **else**
7:         $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
8:         $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
9:     **end if**
10:    …
11: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c |
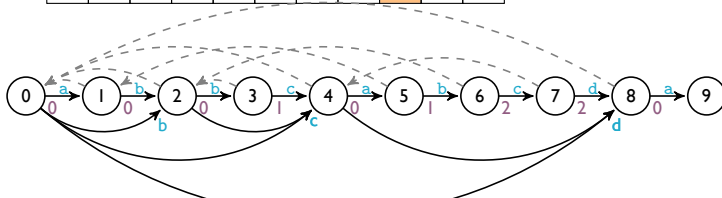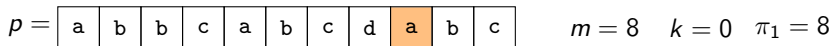
$m = 8 \quad k = 0 \quad \pi_1 = 8$

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

```
 1: function AddLetter(Oracle(p = p₁, p₂ ... pₘ), σ)
 2:     ...
 3:     if k = −1 then
 4:         S_{pσ} ← 0
 5:         lrs_{pσ} ← 0
 6:     else
 7:         S_{pσ} ← state that leads the transition from k by σ
 8:         lrs_{pσ} ← LengthCommonSuffix(π₁, S(m + 1) − 1) + 1
 9:     end if
10:     ...
11: end function
```


$p = $ | a | b | b | c | a | b | c | d | a | b | c | $m = 8$ $k = 0$ $\pi_1 = 8$
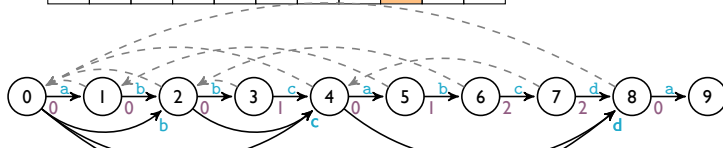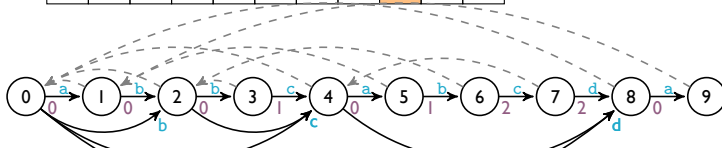
Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     …
3:     **if** $k = -1$ **then**
4:         $S_{p\sigma} \leftarrow 0$
5:         $lrs_{p\sigma} \leftarrow 0$
6:     **else**
7:         $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
8:         $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
9:     **end if**
10:    …
11: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c |

$m = 8 \quad k = 0 \quad \pi_1 = 8$

$lcs(8, 0) = 0$

slide 79

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \ldots p_m$), σ)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ ▷ $\delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     …
7: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c | $m = 9$
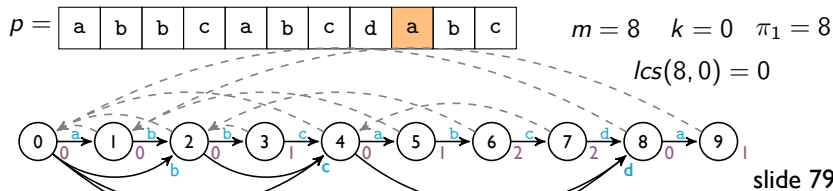
slide 80

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:      Create state $m + 1$
3:      Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:      $k \leftarrow S_p(m)$
5:      $\pi_1 \leftarrow m$
6:      …
7: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c | $m = 9$
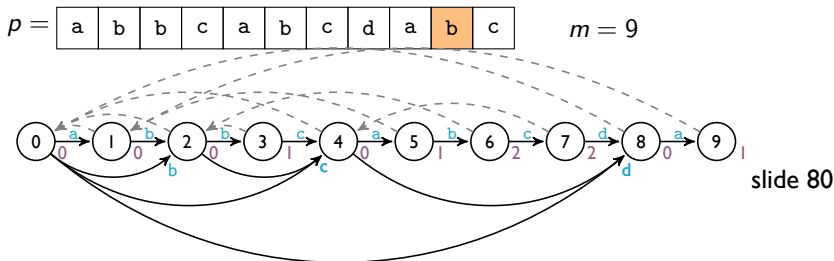
slide 81

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:      Create state $m + 1$
3:      Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:      $k \leftarrow S_p(m)$
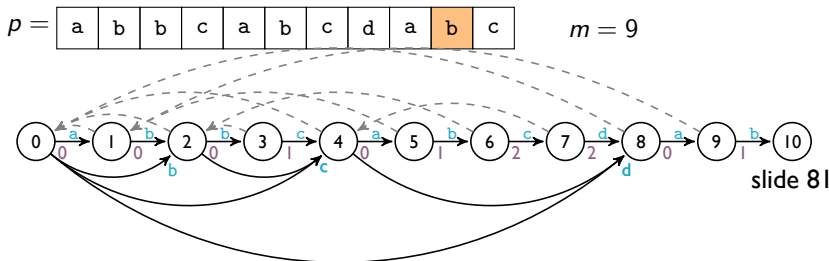5: $\pi_1 \leftarrow m$
6:      ...
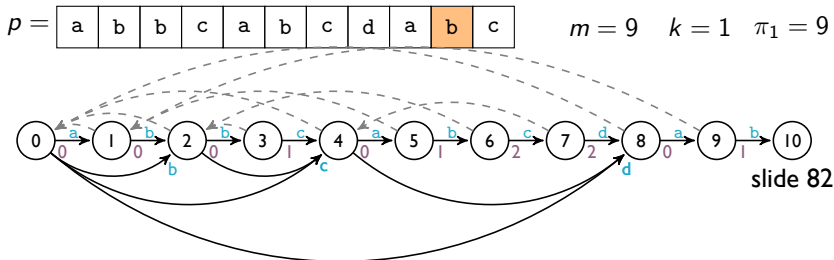7: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c |

$m = 9$ $k = 1$ $\pi_1 = 9$

slide 82

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \ldots p_m$), σ)
2:     …
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ $\triangleright \delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     …
9: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c |

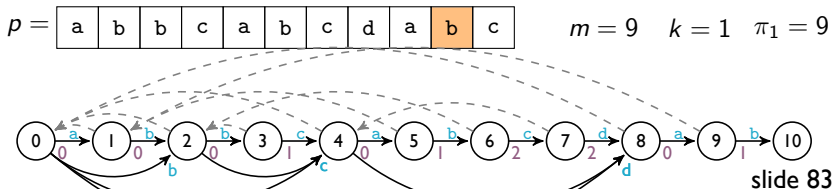$m = 9$ $k = 1$ $\pi_1 = 9$

slide 83

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 ... p_m), \sigma$)
2:     ...
3:     **if** $k = -1$ **then**
4:         $S_{p\sigma} \leftarrow 0$
5:         $lrs_{p\sigma} \leftarrow 0$
6:     **else**
7:         $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
8:         $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
9:     **end if**
10:     ...
11: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c | $m = 9$ $k = 1$ $\pi_1 = 9$

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

```
 1: function AddLetter(Oracle(p = p₁, p₂ ... pₘ), σ)
 2:     ...
 3:     if k = −1 then
 4:         S_{pσ} ← 0
 5:         lrs_{pσ} ← 0
 6:     else
 7:         S_{pσ} ← state that leads the transition from k by σ
 8:         lrs_{pσ} ← LengthCommonSuffix(π₁, S(m + 1) − 1) + 1
 9:     end if
10:     ...
11: end function
```

$p =$ | a | b | b | c | a | b | c | d | a | b | c | $m = 9$ $k = 1$ $\pi_1 = 9$
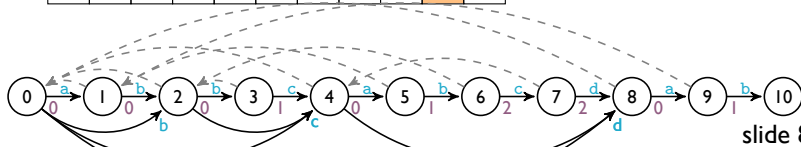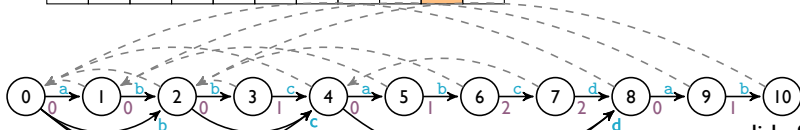
Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
2:     …
3:     **if** $k = -1$ **then**
4:         $S_{p\sigma} \leftarrow 0$
5:         $lrs_{p\sigma} \leftarrow 0$
6:     **else**
7:         $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
8:         $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
9:     **end if**
10:    …
11: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | **b** | c |
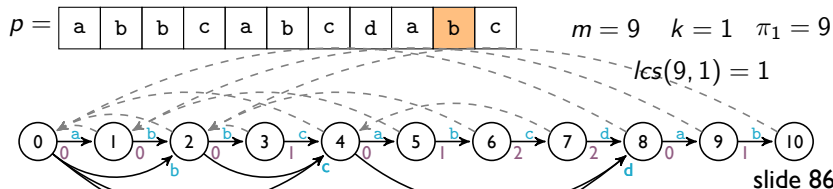
$m = 9$ $k = 1$ $\pi_1 = 9$

$lcs(9, 1) = 1$

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ ▷ $\delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     ...
7: **end function**

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ $\triangleright \delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     ...
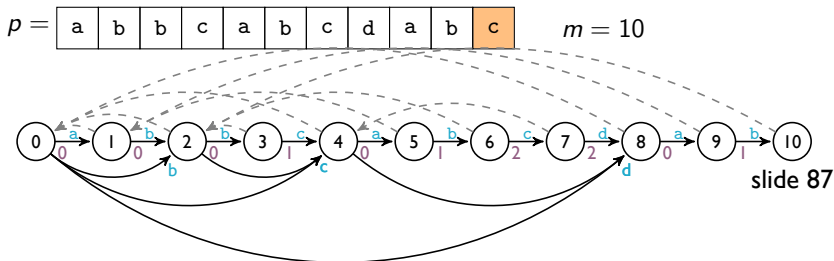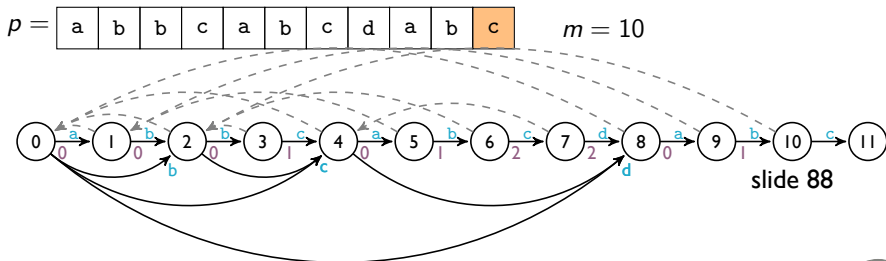7: **end function**

---

slide 88

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 \ldots p_m$), σ)
2:     Create state $m + 1$
3:     Create a new transition from m to $m + 1$ labeled by σ ▷ $\delta(m, \sigma) = m + 1$
4:     $k \leftarrow S_p(m)$
5:     $\pi_1 \leftarrow m$
6:     …
7: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c | $m = 10$ $k = 2$ $\pi_1 = 10$
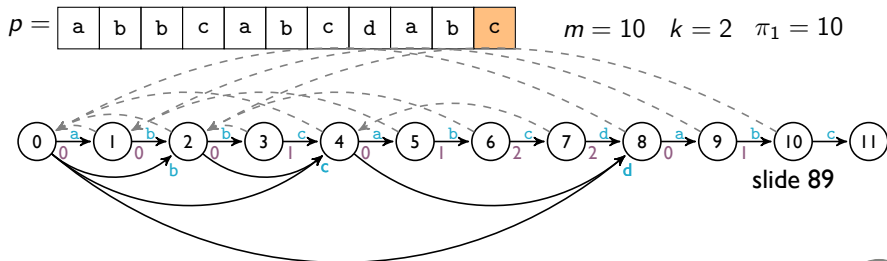
slide 89

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
2:     …
3:     **while** $k > -1$ and there is no transition from k by σ **do**
4:         Create a new transition from k to $m + 1$ by σ $\triangleright \delta(k, \sigma) = m + 1$
5:         $\pi_1 \leftarrow k$
6:         $k \leftarrow S_p(k)$
7:     **end while**
8:     …
9: **end function**

$p =$ | a | b | b | c | a | b | c | d | a | b | c |
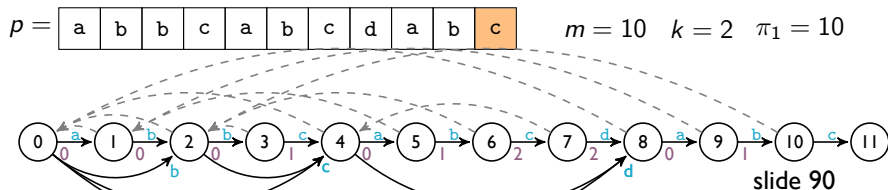
$m = 10$ $k = 2$ $\pi_1 = 10$

slide 90

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

```
 1: function AddLetter(Oracle(p = p₁, p₂ ... pₘ), σ)
 2:     ...
 3:     if k = −1 then
 4:         S_pσ ← 0
 5:         lrs_pσ ← 0
 6:     else
 7:         S_pσ ← state that leads the transition from k by σ
 8:         lrs_pσ ← LengthCommonSuffix(π₁, S(m + 1) − 1) + 1
 9:     end if
10:     ...
11: end function
```

$p =$ | a | b | b | c | a | b | c | d | a | b | **c** | $m = 10$ $k = 2$ $\pi_1 = 10$
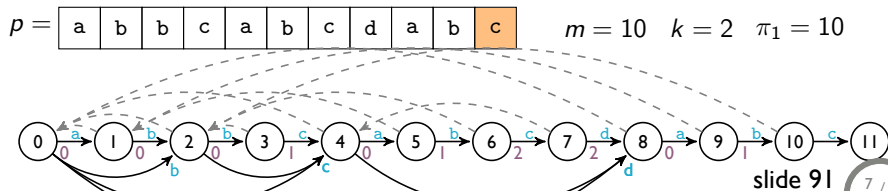
Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter($Oracle(p = p_1, p_2 ... p_m), \sigma$)
2:     ...
3:     **if** $k = -1$ **then**
4:         $S_{p\sigma} \leftarrow 0$
5:         $lrs_{p\sigma} \leftarrow 0$
6:     **else**
7:         $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
8:         $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m + 1) - 1$) + 1
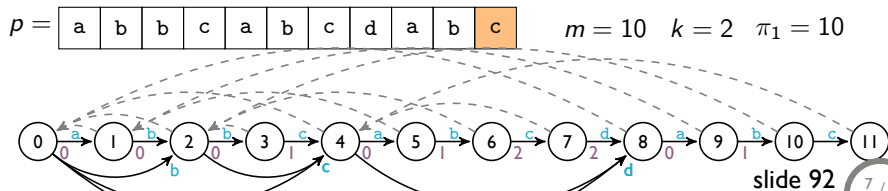9:   **end if**
10:    ...
11: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c | $m = 10$ $k = 2$ $\pi_1 = 10$

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

1: **function** AddLetter(*Oracle*($p = p_1, p_2 ... p_m$), σ)
2:     …
3:     **if** $k = -1$ **then**
4:         $S_{p\sigma} \leftarrow 0$
5:         $lrs_{p\sigma} \leftarrow 0$
6:     **else**
7:         $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
8:         $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
9:     **end if**
10:    …
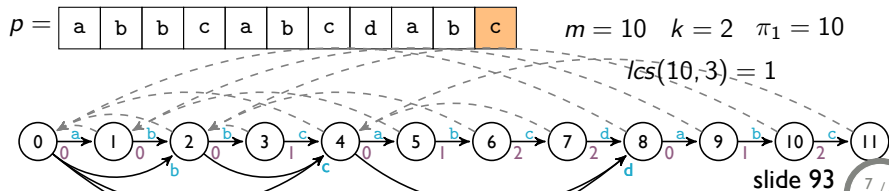11: **end function**

$p = $ | a | b | b | c | a | b | c | d | a | b | c |

$m = 10 \quad k = 2 \quad \pi_1 = 10$

$lcs(10, 3) = 1$

slide 93

Factor Oracle

Algorithm

Algorithm 3 Find Better Algorithm

1: **function** FindBetter(*i*, *a*)
2:     **for** all the elements *j* of $T(i)$ in increasing order **do**
3:         **if** $lrs(j) = lrs(i)$ and $p[j - lrs(i)] = a$ **then**
4:             **return** j
5:         **end if**
6:     **end for**
7:     **return** 0
8: **end function**

Factor Oracle

Algorithm

Algorithm 3 Find Better Algorithm

1: **function** FindBetter(i, a)
2:     **for** all the elements j of $T(S(i))$ in increasing order **do**
3:         **if** $lrs(j) = lrs(i)$ and $p[j - lrs(i)] = a$ **then**
4:             **return** j
5:         **end if**
6:     **end for**
7:     **return** 0
8: **end function**

Factor Oracle

Algorithm

Algorithm 4 Length Common Suffix Algorithm

1: **function** LengthCommonSuffix(π_1, π_2)
2:     **if** $S(\pi_1) = \pi_2$ **then**
3:         **return** $lrs(\pi_1)$
4:     **else**
5:         **while** $S(\pi_1) \neq S(\pi_2)$ **do**
6:             $\pi_2 \leftarrow S(\pi_2)$
7:         **end while**
8:     **end if**
9:     **return** $min(lrs(\pi_1), lrs(\pi_2))$
10: **end function**

Thank you for your attention! ☺

Factor Oracle for Machine Improvisation

Jaime Arias

Université de Bordeaux, LaBRI, UMR 5800
Inria - Bordeaux Sud-Ouest

August 2016