Factor Oracle for Machine Improvisation

Jaime Arias

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

Septembre 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.

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

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 $s[i \dots j] = s_i s_{i+1} \dots s_j$.

$$s = \boxed{\texttt{a}}\boxed{\texttt{b}}\underbrace{\boxed{\texttt{b}}\boxed{\texttt{c}}\boxed{\texttt{a}}}_{s[3,5]}\boxed{\texttt{b}}\boxed{\texttt{c}}\boxed{\texttt{d}}\boxed{\texttt{a}}\boxed{\texttt{b}}\boxed{\texttt{c}}$$

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 $s[1 \dots i]$.

$$s = \underbrace{\boxed{\texttt{a}}\boxed{\texttt{b}}\boxed{\texttt{b}}\boxed{\texttt{c}}}_{pref_s(4)}\boxed{\texttt{a}}\boxed{\texttt{b}}\boxed{\texttt{c}}\boxed{\texttt{d}}\boxed{\texttt{a}}\boxed{\texttt{b}}\boxed{\texttt{c}}$$

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 $s[i \dots m]$.

$$s = \boxed{\texttt{a}}\boxed{\texttt{b}}\boxed{\texttt{b}}\boxed{\texttt{c}}\boxed{\texttt{a}}\underbrace{\boxed{\texttt{b}}\boxed{\texttt{c}}\boxed{\texttt{d}}\boxed{\texttt{a}}\boxed{\texttt{b}}\boxed{\texttt{c}}}_{suff_s(6)}$$

Preliminaries

Longest Repeated Suffix (LRS)

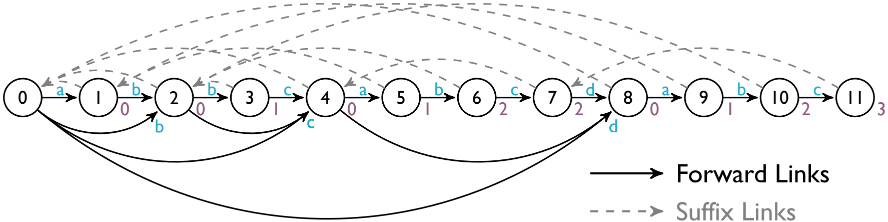
A factor x of s is the longest repeated suffix of s if x is a suffix of s and $|x|$ is maximal.

$$s = \boxed{\texttt{a}}\boxed{\texttt{b}}\boxed{\texttt{b}}\boxed{\texttt{c}}\underbrace{\boxed{\texttt{a}}\boxed{\texttt{b}}\boxed{\texttt{c}}}_{lrs(s)}\boxed{\texttt{d}}\boxed{\texttt{a}}\boxed{\texttt{b}}\boxed{\texttt{c}}$$

Factor Oracle

Factor Oracle

Overview

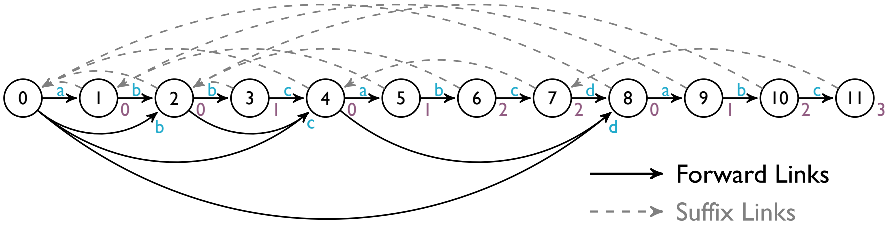


Factor Oracle

The factor oracle of a word s of length m is a *deterministic finite automaton* (Q, q_0, F, δ) where $Q = \{0, 1, \ldots, m\}$ 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.

Factor Oracle

Overview

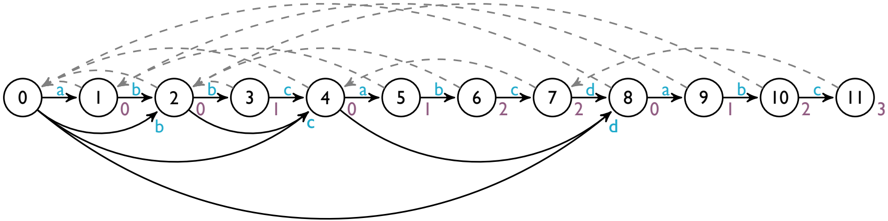


Suffix Link

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

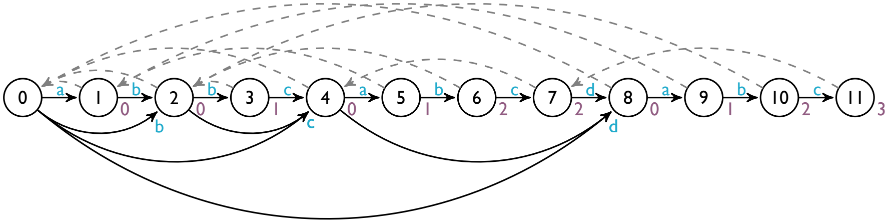


Suffix Links

- $s =$ abbcabcdabc

Factor Oracle

Overview

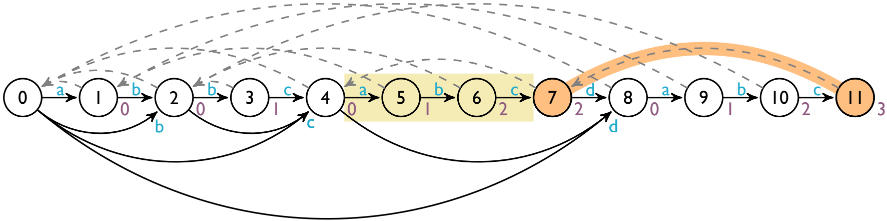


Suffix Links

- $s =$ abbc**abc**d**abc**
- $lrs(s) =$ abc

Factor Oracle

Overview



Suffix Links

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

Factor Oracle

Algorithm - Construction

Algorithm 1 Construction of a Factor Oracle

```
function FactorOracle($p = p_1 p_2 \dots p_m$)
    Create a new oracle $P$ with an initial state 0
    $S_P(0) \leftarrow -1$
    for $i \leftarrow 1, m$ do
        $Oracle(p = p_1 p_2 \dots p_i) \leftarrow$ AddLetter($Oracle(p = p_1 p_2 \dots p_{i-1}), p_i$)
    end for
    return $Oracle(p = p_1 p_2 \dots p_m)$
end function
```

Factor Oracle

Algorithm - Construction

Algorithm 1 Construction of a Factor Oracle

```
function FactorOracle($p = p_1 p_2 \ldots p_m$)
    Create a new oracle $P$ with an initial state 0
    $S_P(0) \leftarrow -1$
    for $i \leftarrow 1, m$ do
        $Oracle(p = p_1 p_2 \ldots p_i) \leftarrow$ AddLetter($Oracle(p = p_1 p_2 \ldots p_{i-1}), p_i$)
    end for
    return $Oracle(p = p_1 p_2 \ldots p_m)$
end function
```

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

Factor Oracle

Algorithm - Construction

Algorithm 1 Construction of a Factor Oracle

```
function FactorOracle($p = p_1 p_2 \dots p_m$)
    Create a new oracle $P$ with an initial state 0
    $S_P(0) \leftarrow -1$
    for $i \leftarrow 1, m$ do
        $Oracle(p = p_1 p_2 \dots p_i) \leftarrow$ AddLetter($Oracle(p = p_1 p_2 \dots p_{i-1}), p_i$)
    end for
    return $Oracle(p = p_1 p_2 \dots p_m)$
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

Factor Oracle

Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$    ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

Factor Oracle

Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        $\triangleright\ \delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 0$

Factor Oracle

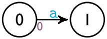
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

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

$m = 0$



Factor Oracle

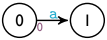
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

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

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



Factor Oracle

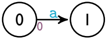
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m + 1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

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

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



Factor Oracle

Algorithm - Construction

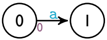
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        $S_{p\sigma} \leftarrow 0$
        $lrs_{p\sigma} \leftarrow 0$
    else
        …
    end if
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

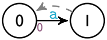
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        $S_{p\sigma} \leftarrow 0$
        $lrs_{p\sigma} \leftarrow 0$
    else
        …
    end if
    …
end function
```

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

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



Factor Oracle

Algorithm - Construction

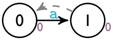
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        $S_{p\sigma} \leftarrow 0$
        $lrs_{p\sigma} \leftarrow 0$
    else
        …
    end if
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

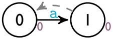
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

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

$m = 1$



Factor Oracle

Algorithm - Construction

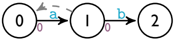
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    ...
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 1$



Factor Oracle

Algorithm - Construction

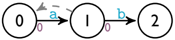
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

Algorithm - Construction

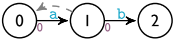
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m + 1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

Algorithm - Construction

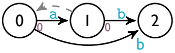
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m + 1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

Algorithm - Construction

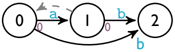
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m + 1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

Algorithm - Construction

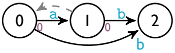
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m + 1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

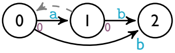
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        $S_{p\sigma} \leftarrow 0$
        $lrs_{p\sigma} \leftarrow 0$
    else
        …
    end if
    …
end function
```

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

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



Factor Oracle

Algorithm - Construction

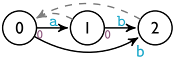
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        $S_{p\sigma} \leftarrow 0$
        $lrs_{p\sigma} \leftarrow 0$
    else
        …
    end if
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

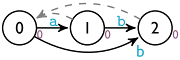
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        $S_{p\sigma} \leftarrow 0$
        $lrs_{p\sigma} \leftarrow 0$
    else
        …
    end if
    …
end function
```

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

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



Factor Oracle

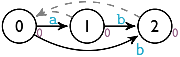
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    ...
end function
```

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

$m = 2$



Factor Oracle

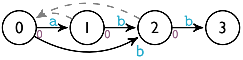
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

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

$m = 2$



Factor Oracle

Algorithm - Construction

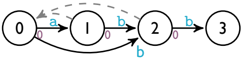
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

Algorithm - Construction

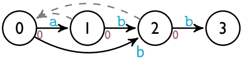
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m+1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

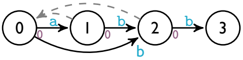
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        …
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    …
end function
```

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

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



Factor Oracle

Algorithm - Construction

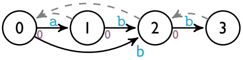
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    ...
    if $k = -1$ then
        ...
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    ...
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

Algorithm - Construction

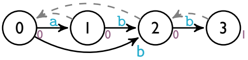
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    …
    if $k = -1$ then
        …
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    …
end function
```

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

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

$lcs(2, 1) = 0$



Factor Oracle

Algorithm - Construction

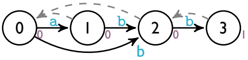
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    ...
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 3$



Factor Oracle

Algorithm - Construction

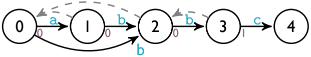
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$    ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 3$



Factor Oracle

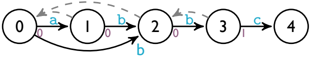
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    ...
end function
```

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

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



Factor Oracle

Algorithm - Construction

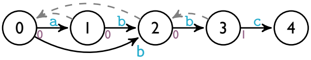
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m+1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

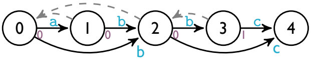
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m+1$ by $\sigma$        ▷ $\delta(k, \sigma) = m+1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

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

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



Factor Oracle

Algorithm - Construction

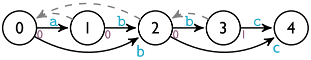
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m + 1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

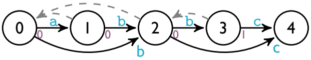
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m + 1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

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

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



Factor Oracle

Algorithm - Construction

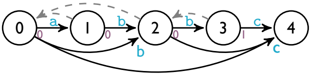
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m+1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

Algorithm - Construction

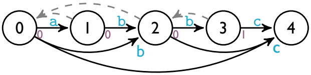
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m+1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

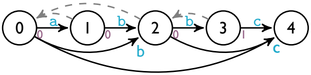
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    ...
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m + 1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    ...
end function
```

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

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



Factor Oracle

Algorithm - Construction

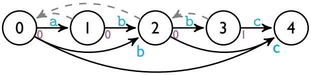
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        $S_{p\sigma} \leftarrow 0$
        $lrs_{p\sigma} \leftarrow 0$
    else
        …
    end if
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

Algorithm - Construction

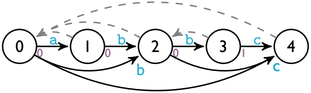
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    ...
    if $k = -1$ then
        $S_{p\sigma} \leftarrow 0$
        $lrs_{p\sigma} \leftarrow 0$
    else
        ...
    end if
    ...
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

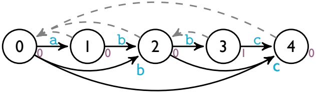
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        $S_{p\sigma} \leftarrow 0$
        $lrs_{p\sigma} \leftarrow 0$
    else
        …
    end if
    …
end function
```

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

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



Factor Oracle

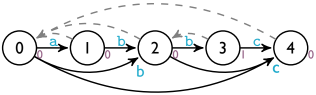
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    ...
end function
```

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

$m = 4$



Factor Oracle

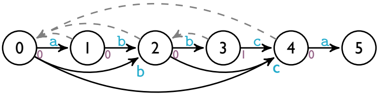
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

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

$m = 4$



Factor Oracle

Algorithm - Construction

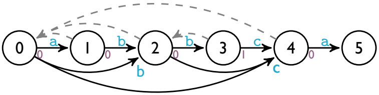
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$    ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    ...
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

Algorithm - Construction

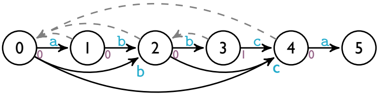
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m + 1$ by $\sigma$          ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

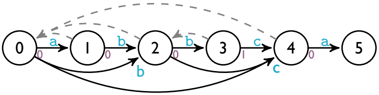
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    ...
    if $k = -1$ then
        ...
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    ...
end function
```

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

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



Factor Oracle

Algorithm - Construction

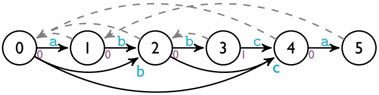
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    …
    if $k = -1$ then
        …
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

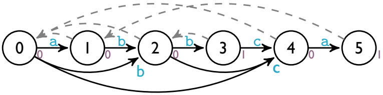
```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        …
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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

$lcs(4, 0) = 0$



Factor Oracle

Algorithm - Construction

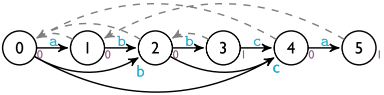
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 5$



Factor Oracle

Algorithm - Construction

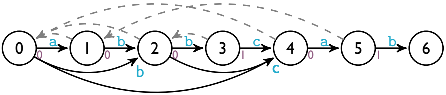
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    ...
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 5$



Factor Oracle

Algorithm - Construction

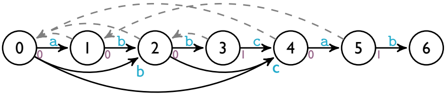
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

Algorithm - Construction

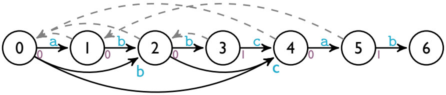
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m+1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

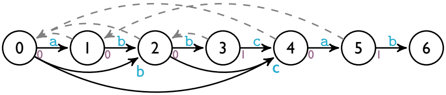
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        …
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    …
end function
```

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

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



Factor Oracle

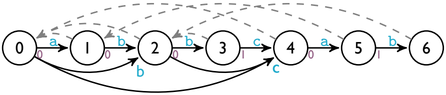
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        …
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    …
end function
```

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

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



Factor Oracle

Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

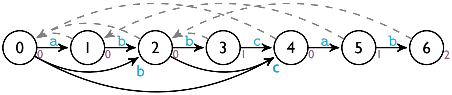
```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        …
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1)-1$) + 1
    end if
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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

$lcs(5, 1) = 1$



Factor Oracle

Algorithm - Construction

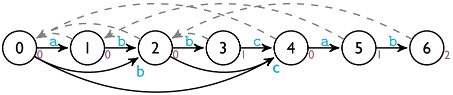
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    ...
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 6$



Factor Oracle

Algorithm - Construction

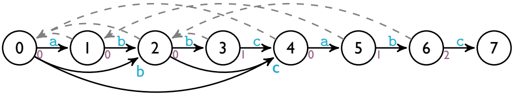
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 6$



Factor Oracle

Algorithm - Construction

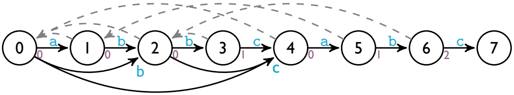
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    ...
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 6$ $\pi_1 = 6$ $k = 2$



Factor Oracle

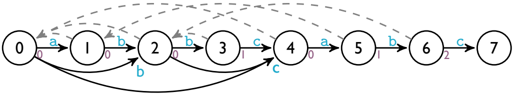
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m+1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

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

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



Factor Oracle

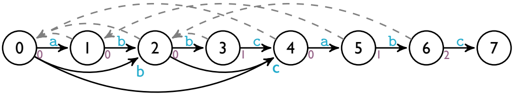
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
**function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    **if** $k = -1$ **then**
        …
    **else**
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    **end if**
    …
**end function**
```

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

$m = 6$ $\pi_1 = 6$ $k = 2$



Factor Oracle

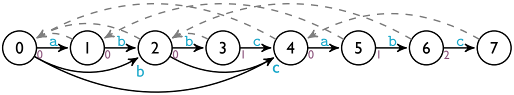
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        …
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    …
end function
```

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

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



Factor Oracle

Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

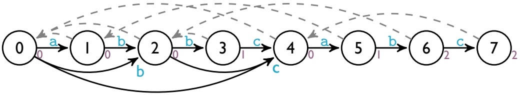
```
**function** AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    **if** $k = -1$ **then**
        …
    **else**
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    **end if**
    …
**end function**
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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

$lcs(6, 3) = 1$



Factor Oracle

Algorithm - Construction

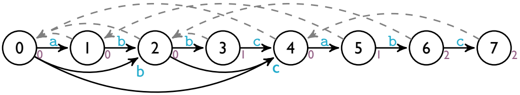
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 7$



Factor Oracle

Algorithm - Construction

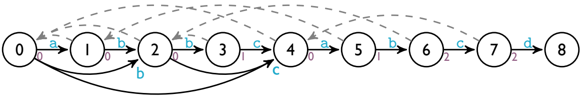
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    ...
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 7$



Factor Oracle

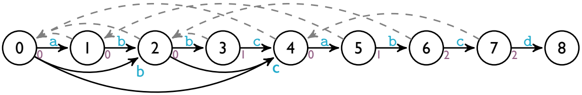
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

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

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

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



Factor Oracle

Algorithm - Construction

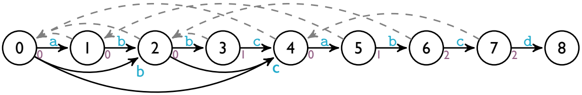
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m+1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

Algorithm - Construction

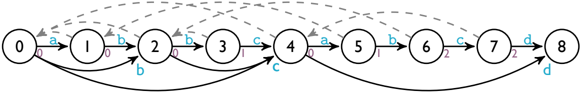
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m+1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

Algorithm - Construction

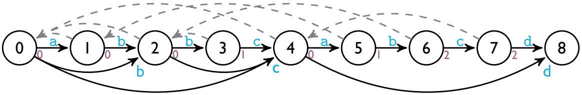
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m+1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

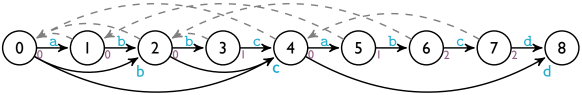
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m + 1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

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

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



Factor Oracle

Algorithm - Construction

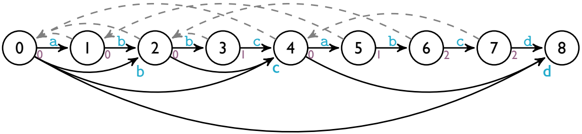
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m + 1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

Algorithm - Construction

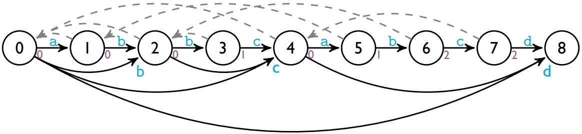
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m+1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

Algorithm - Construction

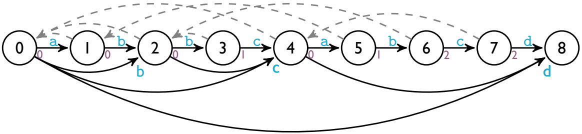
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m + 1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 7$ $\pi_1 = 0$ $k = -1$



Factor Oracle

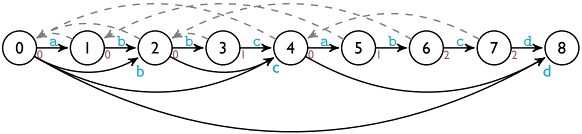
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        $S_{p\sigma} \leftarrow 0$
        $lrs_{p\sigma} \leftarrow 0$
    else
        …
    end if
    …
end function
```

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

$m = 7 \quad \pi_1 = 0 \quad k = -1$



Factor Oracle

Algorithm - Construction

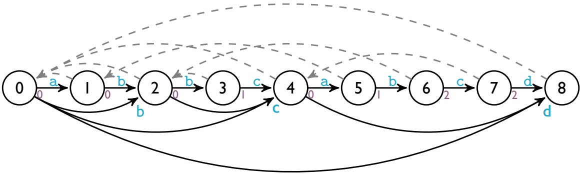
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        $S_{p\sigma} \leftarrow 0$
        $lrs_{p\sigma} \leftarrow 0$
    else
        …
    end if
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 7 \quad \pi_1 = 0 \quad k = -1$



Factor Oracle

Algorithm - Construction

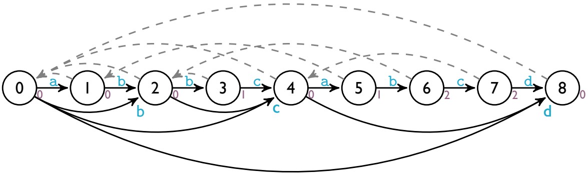
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    ...
    if $k = -1$ then
        $S_{p\sigma} \leftarrow 0$
        $lrs_{p\sigma} \leftarrow 0$
    else
        ...
    end if
    ...
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 7$ $\pi_1 = 0$ $k = -1$



Factor Oracle

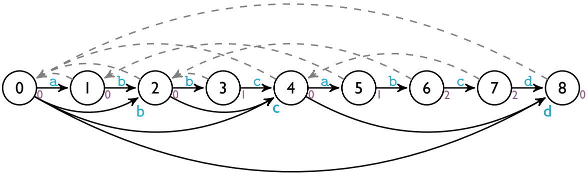
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    ...
end function
```

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

$m = 8$



Factor Oracle

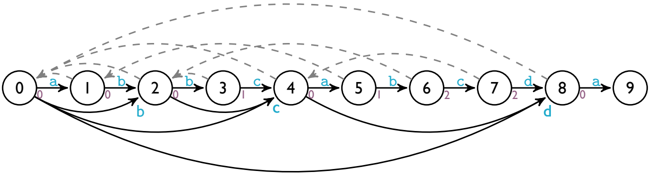
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

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

$m = 8$



Factor Oracle

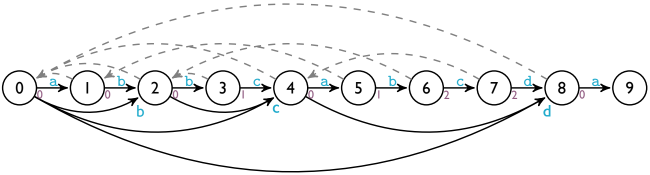
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

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

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



Factor Oracle

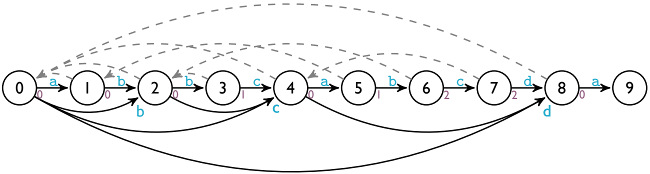
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m + 1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

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

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



Factor Oracle

Algorithm - Construction

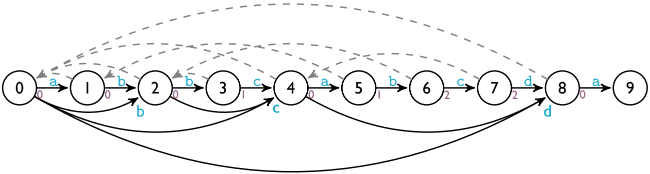
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    ...
    if $k = -1$ then
        ...
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    ...
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

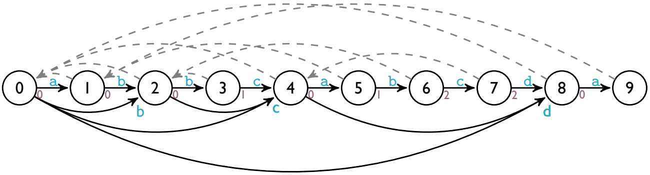
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        …
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m + 1) - 1$) + 1
    end if
    …
end function
```

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

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



Factor Oracle

Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

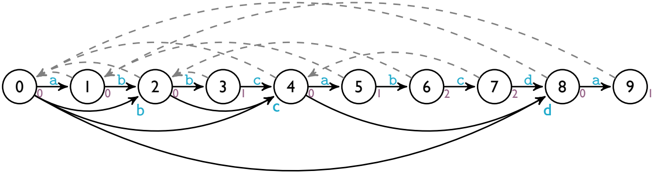
```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        …
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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

$lcs(8, 0) = 0$



Factor Oracle

Algorithm - Construction

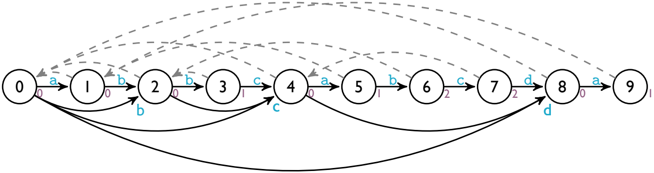
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    ...
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 9$



Factor Oracle

Algorithm - Construction

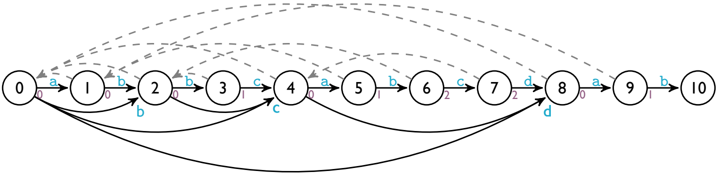
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 9$



Factor Oracle

Algorithm - Construction

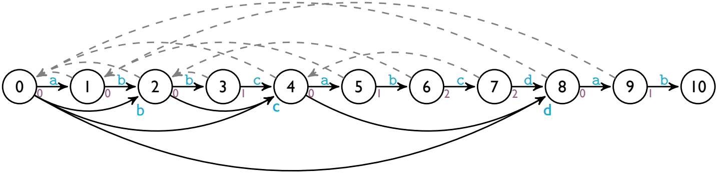
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

Algorithm - Construction

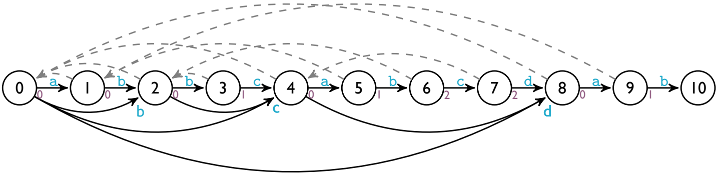
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m + 1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

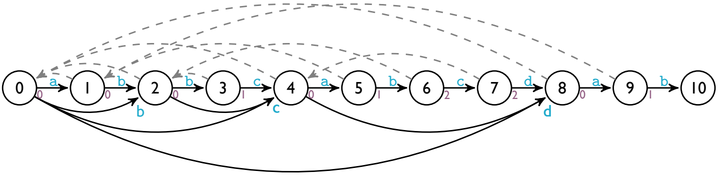
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    …
    if $k = -1$ then
        …
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    …
end function
```

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

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



Factor Oracle

Algorithm - Construction

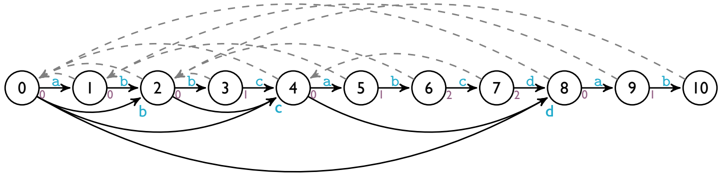
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        …
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

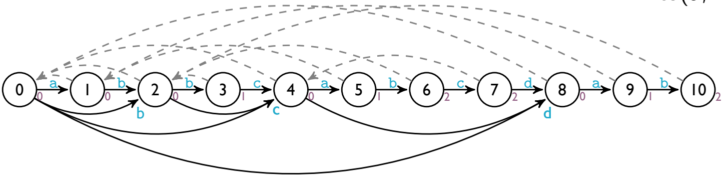
```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        …
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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

$lcs(9, 1) = 1$



Factor Oracle

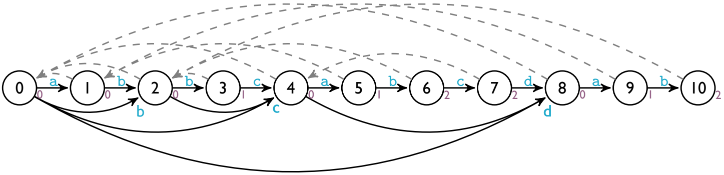
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

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

$m = 10$



Factor Oracle

Algorithm - Construction

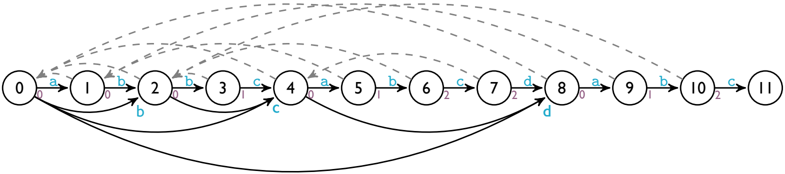
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 10$



Factor Oracle

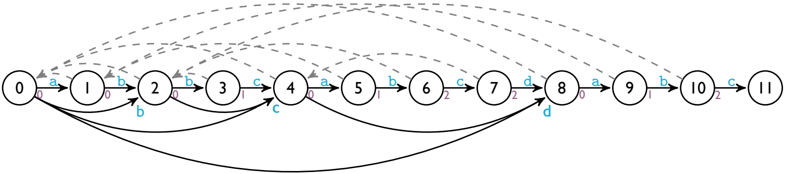
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from $m$ to $m + 1$ labeled by $\sigma$        ▷ $\delta(m, \sigma) = m + 1$
    $\pi_1 \leftarrow m$
    $k \leftarrow S_p(m)$
    ...
end function
```

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

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



Factor Oracle

Algorithm - Construction

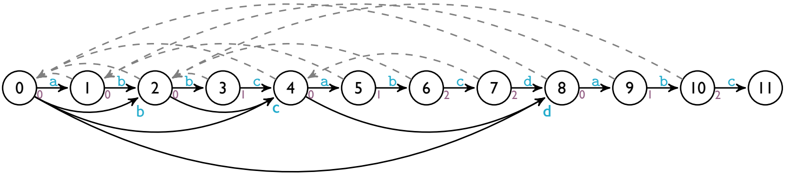
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    ...
    while $k > -1$ and there is no transition from $k$ by $\sigma$ do
        Create a new transition from $k$ to $m + 1$ by $\sigma$        ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    ...
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

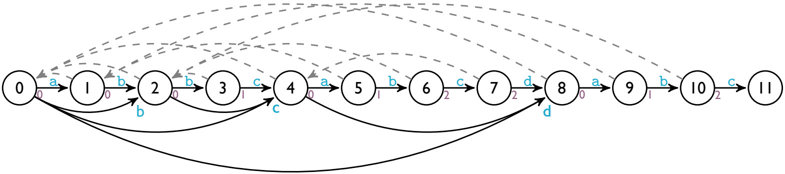
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    …
    if $k = -1$ then
        …
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    …
end function
```

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

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



Factor Oracle

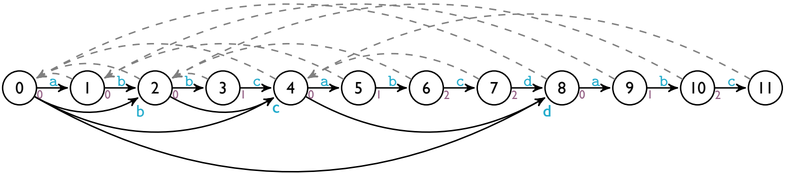
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    …
    if $k = -1$ then
        …
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    …
end function
```

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

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



Factor Oracle

Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

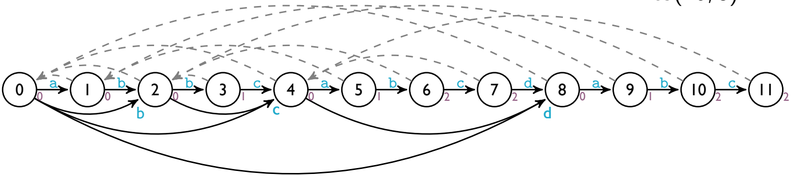
```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        …
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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

$lcs(10, 3) = 1$



Factor Oracle

Algorithm - Construction

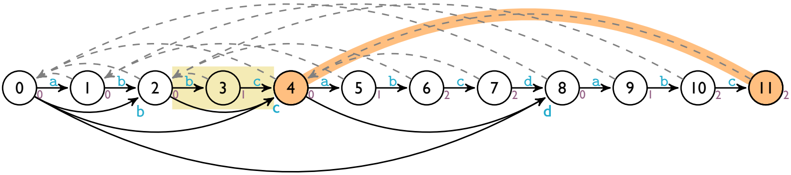
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ then
        …
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from $k$ by $\sigma$
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

Algorithm - Construction

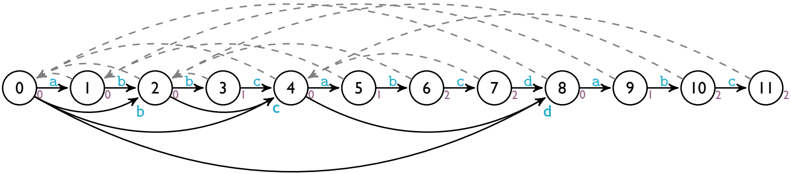
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    $k \leftarrow$ FindBetter($m + 1, p[m + 1 - lrs(m + 1)]$)
    if $k \neq 0$ then
        $lrs_{p\sigma} \leftarrow lrs(m + 1) + 1$
        $S_{p\sigma} \leftarrow k$
    end if
    $T(S_{p\sigma}) \leftarrow T(S(m + 1)) \cup \{m + 1\}$        ▷ $T(i) = \{j \mid S(j) = i \wedge i < j \leq m\}$
    return $Oracle(p = p_1 p_2 \dots p_m \sigma)$
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

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



Factor Oracle

Algorithm - Construction

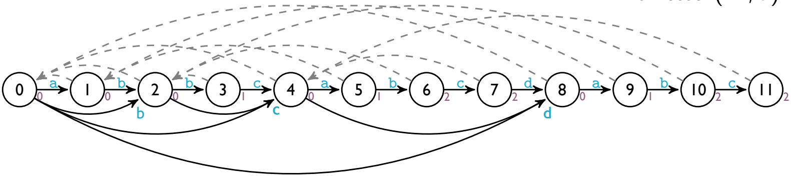
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    $k \leftarrow$ FindBetter($m + 1, p[m + 1 - lrs(m + 1)]$)
    if $k \neq 0$ then
        $lrs_{p\sigma} \leftarrow lrs(m + 1) + 1$
        $S_{p\sigma} \leftarrow k$
    end if
    $T(S_{p\sigma}) \leftarrow T(S(m + 1)) \cup \{m + 1\}$          ▷ $T(i) = \{j \mid S(j) = i \wedge i < j \leq m\}$
    return $Oracle(p = p_1 p_2 \dots p_m \sigma)$
end function
```

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

$m = 10 \quad \pi_1 = 10 \quad k = 7$

$FindBetter(11, \mathrm{a}) = 7$



Factor Oracle

Algorithm - Construction

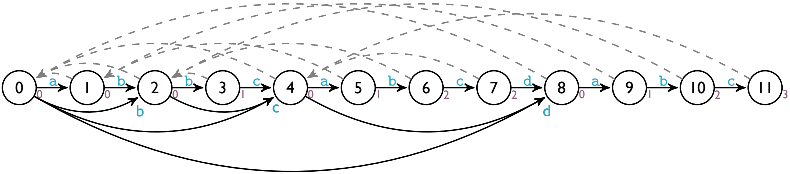
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    $k \leftarrow$ FindBetter($m+1, p[m+1-lrs(m+1)]$)
    if $k \neq 0$ then
        $lrs_{p\sigma} \leftarrow lrs(m+1)+1$
        $S_{p\sigma} \leftarrow k$
    end if
    $T(S_{p\sigma}) \leftarrow T(S(m+1)) \cup \{m+1\}$        ▷ $T(i) = \{j \mid S(j) = i \wedge i < j \leq m\}$
    return $Oracle(p = p_1 p_2 \dots p_m \sigma)$
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 10 \quad \pi_1 = 10 \quad k = 7$



Factor Oracle

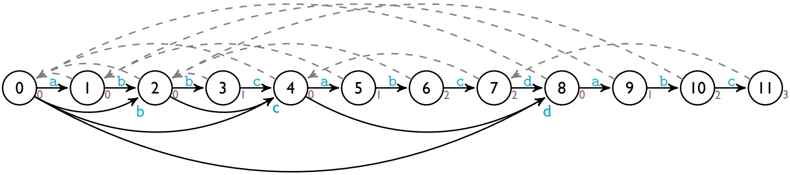
Algorithm - Construction

Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    $k \leftarrow$ FindBetter($m + 1, p[m + 1 - lrs(m + 1)]$)
    if $k \neq 0$ then
        $lrs_{p\sigma} \leftarrow lrs(m + 1) + 1$
        $S_{p\sigma} \leftarrow k$
    end if
    $T(S_{p\sigma}) \leftarrow T(S(m + 1)) \cup \{m + 1\}$        ▷ $T(i) = \{j \mid S(j) = i \wedge i < j \leq m\}$
    return $Oracle(p = p_1 p_2 \dots p_m \sigma)$
end function
```

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

$m = 10 \quad \pi_1 = 10 \quad k = 7$



Factor Oracle

Algorithm - Construction

Algorithm 3 Length Common Suffix Algorithm

```
function LengthCommonSuffix($\pi_1, \pi_2$)
    if $S(\pi_1) = \pi_2$ then
        return $lrs(\pi_1)$
    else
        while $S(\pi_1) \neq S(\pi_2)$ do
            $\pi_2 \leftarrow S(\pi_2)$
        end while
    end if
    return $min(lrs(\pi_1), lrs(\pi_2))$
end function
```

Factor Oracle

Algorithm - Construction

Algorithm 3 Length Common Suffix Algorithm

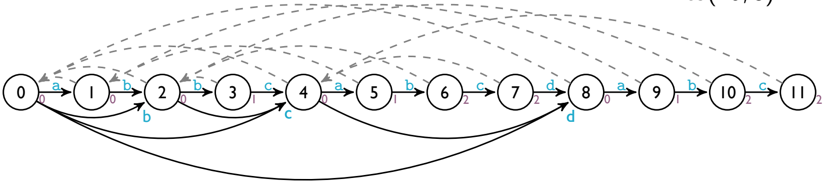
```
function LengthCommonSuffix($\pi_1, \pi_2$)
    if $S(\pi_1) = \pi_2$ then
        return $lrs(\pi_1)$
    else
        while $S(\pi_1) \neq S(\pi_2)$ do
            $\pi_2 \leftarrow S(\pi_2)$
        end while
    end if
    return $min(lrs(\pi_1), lrs(\pi_2))$
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 10 \quad \pi_1 = 10 \quad \pi_2 = 3$

$lcs(10, 3) =$



Factor Oracle

Algorithm - Construction

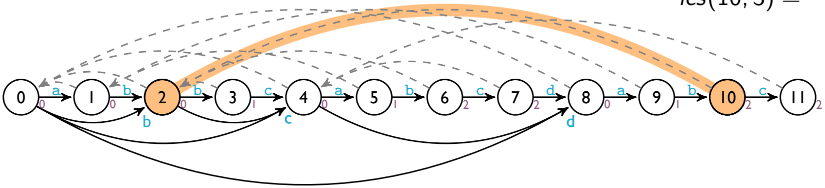
Algorithm 3 Length Common Suffix Algorithm

```
function LengthCommonSuffix($\pi_1, \pi_2$)
    if $S(\pi_1) = \pi_2$ then
        return $lrs(\pi_1)$
    else
        while $S(\pi_1) \neq S(\pi_2)$ do
            $\pi_2 \leftarrow S(\pi_2)$
        end while
    end if
    return $min(lrs(\pi_1), lrs(\pi_2))$
end function
```

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

$m = 10 \quad \pi_1 = 10 \quad \pi_2 = 3$

$lcs(10, 3) =$



Factor Oracle

Algorithm - Construction

Algorithm 3 Length Common Suffix Algorithm

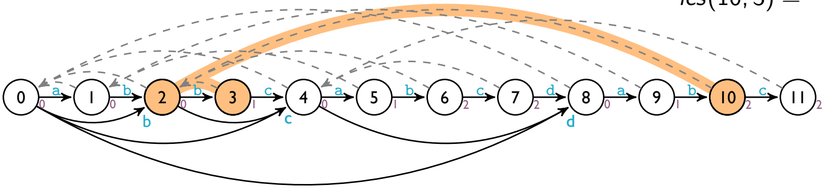
```
function LengthCommonSuffix($\pi_1, \pi_2$)
    if $S(\pi_1) = \pi_2$ then
        return $lrs(\pi_1)$
    else
        while $S(\pi_1) \neq S(\pi_2)$ do
            $\pi_2 \leftarrow S(\pi_2)$
        end while
    end if
    return $min(lrs(\pi_1), lrs(\pi_2))$
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 10 \quad \pi_1 = 10 \quad \pi_2 = 3$

$lcs(10, 3) =$



Factor Oracle

Algorithm - Construction

Algorithm 3 Length Common Suffix Algorithm

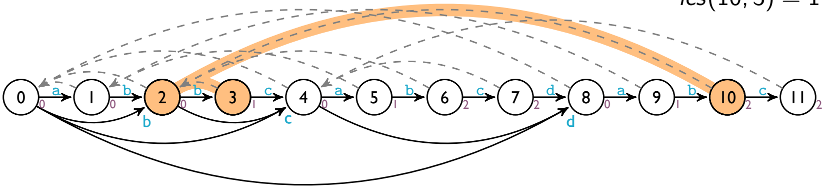
```
function LengthCommonSuffix($\pi_1, \pi_2$)
    if $S(\pi_1) = \pi_2$ then
        return $lrs(\pi_1)$
    else
        while $S(\pi_1) \neq S(\pi_2)$ do
            $\pi_2 \leftarrow S(\pi_2)$
        end while
    end if
    return $min(lrs(\pi_1), lrs(\pi_2))$
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 10 \quad \pi_1 = 10 \quad \pi_2 = 3$

$lcs(10, 3) = 1$



Factor Oracle

Algorithm - Construction

Algorithm 4 Find Better Algorithm

```
function FindBetter($i, \sigma$)
    for all the elements $j$ of $T(i)$ in increasing order do
        if $lrs(j) = lrs(i)$ and $p[j - lrs(i)] = \sigma$ then
            return j
        end if
    end for
    return 0
end function
```

Factor Oracle

Algorithm - Construction

Algorithm 4 Find Better Algorithm

```
function FindBetter($i, \sigma$)
    for all the elements $j$ of $T(S(i))$ in increasing order do
        if $lrs(j) = lrs(i)$ and $p[j - lrs(i)] = \sigma$ then
            return j
        end if
    end for
    return 0
end function
```

Factor Oracle

Algorithm - Construction

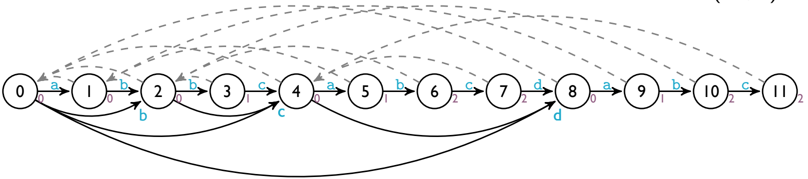
Algorithm 4 Find Better Algorithm

```
function FindBetter($i, \sigma$)
    for all the elements $j$ of $T(S(i))$ in increasing order do
        if $lrs(j) = lrs(i)$ and $p[j - lrs(i)] = \sigma$ then
            return j
        end if
    end for
    return 0
end function
```

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

$m = 10 \quad i = 11 \quad \sigma = \mathtt{a}$

$FindBetter(11, \mathtt{a}) =$



Factor Oracle

Algorithm - Construction

Algorithm 4 Find Better Algorithm

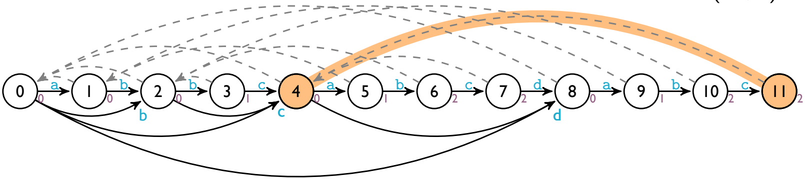
```
function FindBetter($i$, $\sigma$)
    for all the elements $j$ of $T(S(i))$ in increasing order do
        if $lrs(j) = lrs(i)$ and $p[j - lrs(i)] = \sigma$ then
            return j
        end if
    end for
    return 0
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 10 \quad i = 11 \quad \sigma = \mathtt{a}$

$FindBetter(11, \mathtt{a}) =$



Factor Oracle

Algorithm - Construction

Algorithm 4 Find Better Algorithm

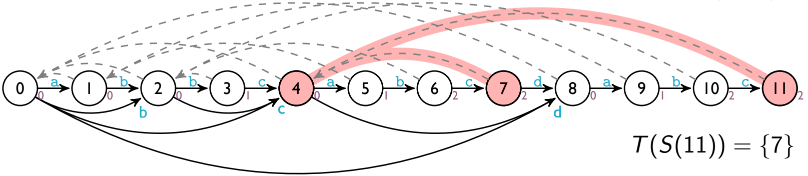
```
function FindBetter($i$, $\sigma$)
    for all the elements $j$ of $T(S(i))$ in increasing order do
        if $lrs(j) = lrs(i)$ and $p[j - lrs(i)] = \sigma$ then
            return j
        end if
    end for
    return 0
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 10 \quad i = 11 \quad \sigma = \mathtt{a}$

$FindBetter(11, \mathtt{a}) =$



$T(S(11)) = \{7\}$

Factor Oracle

Algorithm - Construction

Algorithm 4 Find Better Algorithm

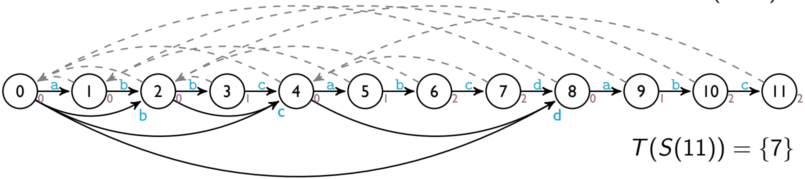
```
function FindBetter($i$, $\sigma$)
    for all the elements $j$ of $T(S(i))$ in increasing order do
        if $lrs(j) = lrs(i)$ and $p[j - lrs(i)] = \sigma$ then
            return j
        end if
    end for
    return 0
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 10 \quad i = 11 \quad \sigma = \mathtt{a}$

$FindBetter(11, \mathtt{a}) =$



$T(S(11)) = \{7\}$

$j = 7$

Factor Oracle

Algorithm - Construction

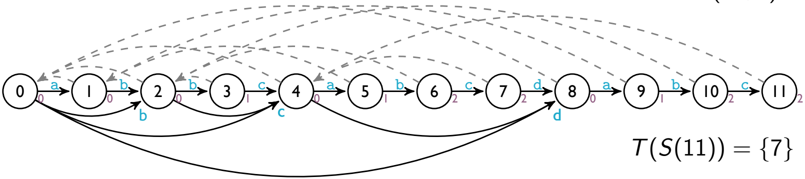
Algorithm 4 Find Better Algorithm

```
function FindBetter($i, \sigma$)
    for all the elements $j$ of $T(S(i))$ in increasing order do
        if $lrs(j) = lrs(i)$ and $p[j - lrs(i)] = \sigma$ then
            return j
        end if
    end for
    return 0
end function
```

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

$m = 10 \quad i = 11 \quad \sigma = \mathtt{a}$

$FindBetter(11, \mathtt{a}) =$



$T(S(11)) = \{7\}$

$j = 7$

Factor Oracle

Algorithm - Construction

Algorithm 4 Find Better Algorithm

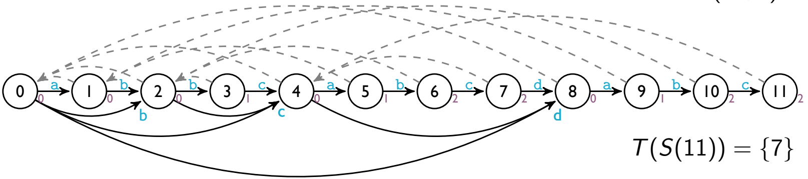
```
function FindBetter($i$, $\sigma$)
    for all the elements $j$ of $T(S(i))$ in increasing order do
        if $lrs(j) = lrs(i)$ and $p[j - lrs(i)] = \sigma$ then
            return j
        end if
    end for
    return 0
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 10 \quad i = 11 \quad \sigma = \mathtt{a}$

$FindBetter(11, \mathtt{a}) = 7$



$T(S(11)) = \{7\}$

$j = 7$

Factor Oracle

Algorithm - Improvisation

Algorithm 5 FO-Generate function

Require: Oracle $P = p_1, p_2 \dots p_m$ in active state i, a generated sequence v, and a continuation parameter $0 \leq q \leq 1$.

```
Generate uniformly distribute random number $u$
**if** $u < q$ **then**
    $i \leftarrow i + 1$        $v \leftarrow vp_i$
**else**
    Choose at random a symbol $\sigma \in \{\sigma_j \mid \delta(S(i), \sigma_j) \neq \perp\}$
    $i \leftarrow \delta(S(i), \sigma)$        $v \leftarrow v\sigma$
**end if**
**return** Sequence $v$
```

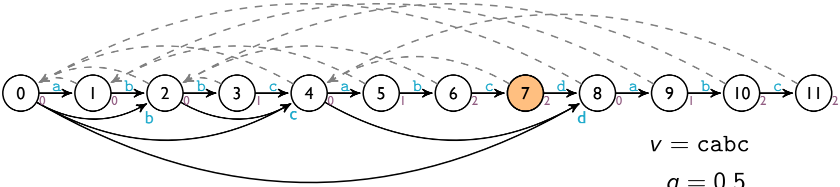
Factor Oracle

Algorithm - Improvisation

Algorithm 5 FO-Generate function

Require: Oracle $P = p_1, p_2 \ldots p_m$ in active state i, a generated sequence v, and a continuation parameter $0 \leq q \leq 1$.

```
Generate uniformly distribute random number $u$
if $u < q$ then
    $i \leftarrow i + 1$        $v \leftarrow vp_i$
else
    Choose at random a symbol $\sigma \in \{\sigma_j \mid \delta(S(i), \sigma_j) \neq \perp\}$
    $i \leftarrow \delta(S(i), \sigma)$        $v \leftarrow v\sigma$
end if
return Sequence $v$
```



$v = \text{cabc}$

$q = 0.5$

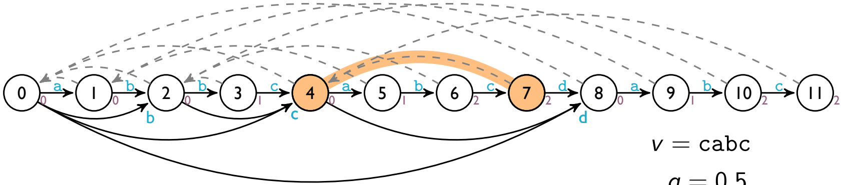
Factor Oracle

Algorithm - Improvisation

Algorithm 5 FO-Generate function

Require: Oracle $P = p_1, p_2 \ldots p_m$ in active state i, a generated sequence v, and a continuation parameter $0 \leq q \leq 1$.

```
Generate uniformly distribute random number $u$
if $u < q$ then
    $i \leftarrow i + 1$        $v \leftarrow vp_i$
else
    Choose at random a symbol $\sigma \in \{\sigma_j \mid \delta(S(i), \sigma_j) \neq \bot\}$
    $i \leftarrow \delta(S(i), \sigma)$        $v \leftarrow v\sigma$
end if
return Sequence $v$
```



$v = \text{cabc}$

$q = 0.5$

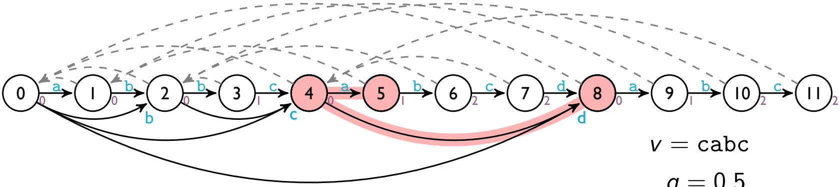
Factor Oracle

Algorithm - Improvisation

Algorithm 5 FO-Generate function

Require: Oracle $P = p_1, p_2 \dots p_m$ in active state i, a generated sequence v, and a continuation parameter $0 \leq q \leq 1$.

```
Generate uniformly distribute random number $u$
if $u < q$ then
    $i \leftarrow i + 1$        $v \leftarrow vp_i$
else
    Choose at random a symbol $\sigma \in \{\sigma_j \mid \delta(S(i), \sigma_j) \neq \bot\}$
    $i \leftarrow \delta(S(i), \sigma)$        $v \leftarrow v\sigma$
end if
return Sequence $v$
```



$v = \text{cabc}$

$q = 0.5$

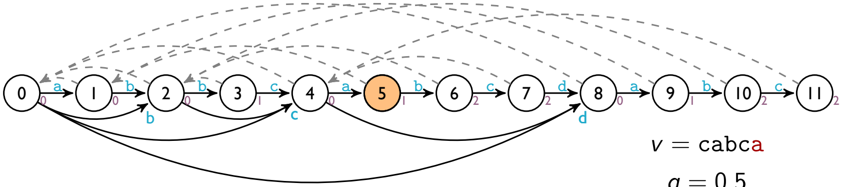
Factor Oracle

Algorithm - Improvisation

Algorithm 5 FO-Generate function

Require: Oracle $P = p_1, p_2 \dots p_m$ in active state i, a generated sequence v, and a continuation parameter $0 \leq q \leq 1$.

```
Generate uniformly distribute random number $u$
if $u < q$ then
    $i \leftarrow i + 1$        $v \leftarrow vp_i$
else
    Choose at random a symbol $\sigma \in \{\sigma_j \mid \delta(S(i), \sigma_j) \neq \bot\}$
    $i \leftarrow \delta(S(i), \sigma)$        $v \leftarrow v\sigma$
end if
return Sequence $v$
```



$v = \texttt{cabca}$

$q = 0.5$

Thank you for your attention! ☺

Factor Oracle for Machine Improvisation

Jaime Arias

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

Septembre 2016

References

- Gerard Assayag and Shlomo Dubnov.
  Using Factor Oracles for Machine Improvisation.
  Soft Computing, 8(9), sep 2004.

- Shlomo Dubnov, Gérard Assayag, and Arshia Cont.
  Audio Oracle: a New Algorithm for Fast Learning of audio Structures.
  In *International Computer Music Conference*, 2007.

- Arnaud Lefebvre, Thierry Lecroq, and Joël Alexandre.
  An Improved Algorithm for Finding Longest Repeats with a Modified Factor Oracle.
  J. Autom. Lang. Comb., 8(4):647–657, 2003.

- Greg Surges and Shlomo Dubnov.
  Feature Selection and Composition Using PyOracle.
  In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2013.