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.

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

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

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

$$\underbrace{\mathtt{b}\ \mathtt{c}\ \mathtt{a}}_{s[3,5]}$$

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 $s[1 \ldots i]$.

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

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 $s[i \ldots m]$.

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

Preliminaries

Longest Repeated Suffix (LRS)

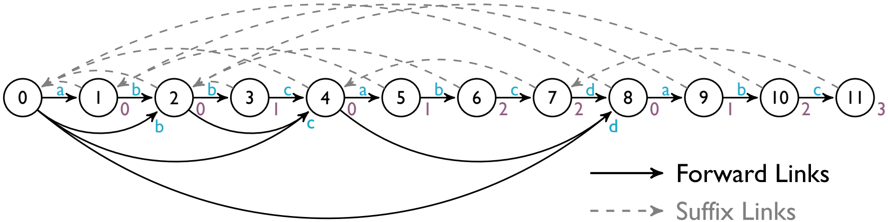
A factor x of s is the longest repeated suffix of s if x is a suffix of s and $|x|$ is maximal.

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

Factor Oracle

Factor Oracle

Overview

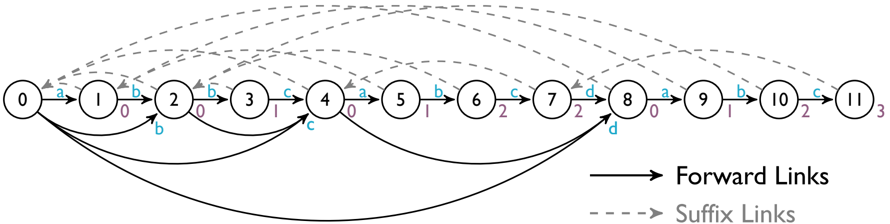


Factor Oracle

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

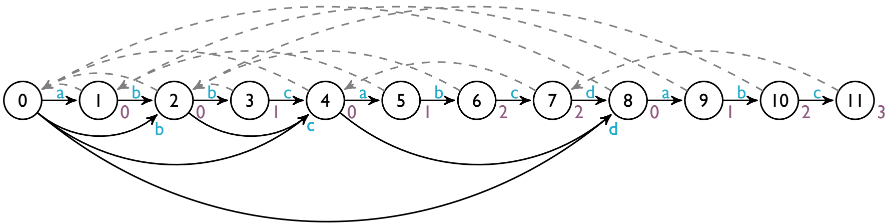


Suffix Link

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 $s[1 \ldots i]$ is recognized.

Factor Oracle

Overview

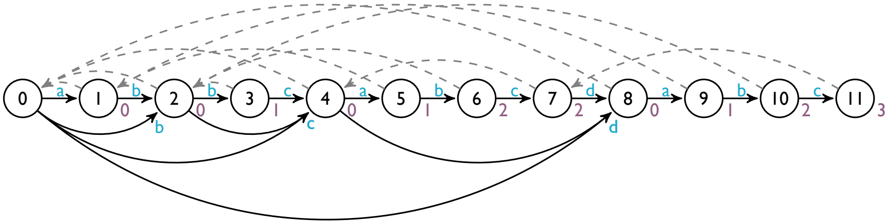


Suffix Links

- $s = \texttt{abbcabcdabc}$

Factor Oracle

Overview

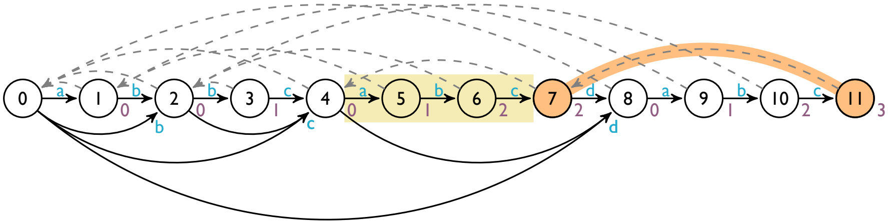


Suffix Links

- $s =$ abbcabcdabc
- $lrs(s) =$ abc

Factor Oracle

Overview



Suffix Links

- $s =$ `abbcabcdabc`
- $lrs(s) =$ `abc`
- $S(11) = 7$

Factor Oracle

Algorithm

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

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 σ $\qquad \triangleright\ \delta(m, \sigma) = m + 1$
    $k \leftarrow S_p(m)$
    $\pi_1 \leftarrow m$
    while $k > -1$ and there is no transition from k by σ **do**
        Create a new transition from k to $m + 1$ by σ $\qquad \triangleright\ \delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    if $k = -1$ **then**
        $S_{p\sigma} \leftarrow 0$
        $lrs_{p\sigma} \leftarrow 0$
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m + 1) - 1$) + 1
    end if

Factor Oracle

Algorithm

Algorithm 3 Incremental update of Factor Oracle

$k \leftarrow \text{FindBetter}(m+1, p[m+1-lrs(m+1)])$
if $k \neq 0$ **then**
$lrs_{p\sigma} \leftarrow lrs(m+1)+1$
$S_{p\sigma} \leftarrow k$
end if
$T(S_{p\sigma}) \leftarrow T(S(m+1)) \cup \{m+1\}$
return $Oracle(p = p_1 p_2 \dots p_m \sigma)$
end function

Factor Oracle

Algorithm

Algorithm 4 Find Better Algorithm

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

Factor Oracle

Algorithm

Algorithm 5 Find Better Algorithm

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

Factor Oracle

Algorithm

Algorithm 6 Length Common Suffix Algorithm

```
function LengthCommonSuffix(π1, π2)
    if S(π1) = π2 then
        return lrs(π1)
    else
        while S(π1) ≠ S(π2) do
            π2 ← S(π2)
        end while
    end if
    return min(lrs(π1), lrs(π2))
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