Computation of Variations of the LZ77 factorization and the LPF Array with Suffix Trees

Dominik Köppl

M&D Data Science Center, Tokyo Medical and Dental University, Japan

results of:

- “Non-Overlapping LZ77 Factorization and LZ78 Substring Compression Queries with Suffix Trees.” Algorithms 14(2): 44 (2021)
in this talk

variations of Lempel-Ziv 77 (LZ77) factorization:
- non-overlapping Lempel-Ziv 77 (NOV LZ)
- reversed LZ

variations of longest previous factor array (LPF):
- LPnF: longest previous non-overlapping factor array
- LPnrF: longest previous non-overlapping reversed factor array

our contribution:
- $2n$-bit representations of LPnF and LPnrF
- (near) linear-time algorithms computing the mentioned factorizations/arrays in small space
setting & example

- $T$: input text, $n - 1 := |T|$ length of $T$
- $\Sigma$: alphabet of $T$, $\sigma := |\Sigma|$ size of $\Sigma$
- $\$ < $c \forall c \in \Sigma$

<table>
<thead>
<tr>
<th>$i$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>a</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>b</td>
<td>a</td>
<td>b</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>LPF</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>LPnF</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>LPnrF</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
succinct representation

Sadakane'07: $2n$-bit representation PLCP array

- PLCP$[i]$: longest common prefix of $(T\$)[i..]$ with its lexicographically preceding suffix
- PLCP$[n] = 0$ since $(T\$)[n] = \$
- PLCP$[i] \leq n \forall i \in [1..n]$
- PLCP$[i] \geq PLCP[i - 1] - 1$
succinct representation

Sadakane’07: $2n$-bit representation PLCP array

- $PLCP[i]$: longest common prefix of $(T\$)[..i]$ with its lexicographically preceding suffix
- $PLCP[n] = 0$ since $(T\$)[n] = \$
- $PLCP[i] \leq n \ \forall \ i \in [1..n]$
- $PLCP[i] \geq PLCP[i - 1] - 1$

$T = F' F' \quad i - 1 \quad i$
succinct representation

Sadakane'07: $2n$-bit representation PLCP array

- PLCP[$i$]: longest common prefix of $(T$)$[i..]$ with its lexicographically preceding suffix
- PLCP[$n$] = 0 since $(T$)$[n] = $
- PLCP[$i$] $\leq n \ \forall \ i \in [1..n]$
- PLCP[$i$] $\geq$ PLCP[$i - 1$] - 1
- hence: $\text{PLCP}[i] - \text{PLCP}[i - 1] + 1 \geq 0$.

$\Rightarrow$ store all values of PLCP[$i$] - PLCP[$i - 1$] + 1 in unary:

$\sum_{i=1}^{n}(\text{PLCP}[i] - \text{PLCP}[i - 1] + 1) = \text{PLCP}[n] + n = n$ with PLCP[0] := 0

$\Rightarrow$ need $n$ ‘1’s and $n$ ‘0’s

$\Rightarrow 2n$ bits.
related arrays with same representation

<table>
<thead>
<tr>
<th>Ref.</th>
<th>array</th>
<th>factorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sadakane'07</td>
<td>PLPF</td>
<td>lcpcmp (Dinklage+'17)</td>
</tr>
<tr>
<td>Belazzougui and Cunial'14</td>
<td>matching statistics</td>
<td>Relative LZ (Ziv+'93)</td>
</tr>
<tr>
<td>Bannai, Inenaga, K.'17</td>
<td>LPF</td>
<td>LZ77</td>
</tr>
<tr>
<td>this talk</td>
<td>LPnF</td>
<td>NOV LZ</td>
</tr>
<tr>
<td>this talk</td>
<td>LPnrF</td>
<td>reversed LZ (Kolpakov+'09)</td>
</tr>
</tbody>
</table>

- NOV LZ: Non Overlapping Lempel-Ziv 77
- Ref.: first occurrence of $2n$-bit representation
### Previous Work

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Time</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LPnF:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crochemore, Tischler’11</td>
<td>$O(n)$</td>
<td>$O(n \lg n)$</td>
</tr>
<tr>
<td>Crochemore+’12</td>
<td>$O(n)$</td>
<td>$O(n \lg n)$</td>
</tr>
<tr>
<td>Ohlebusch, Weber’19:</td>
<td>$O(n)$</td>
<td>$O(n \lg n)$</td>
</tr>
<tr>
<td><strong>LPnrF:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kolpakov, Kucherov’09</td>
<td>$O(n \lg \sigma)$</td>
<td>$O(n \lg n)$</td>
</tr>
<tr>
<td>Chairungsee, Crochemore’09</td>
<td>$O(n \lg \sigma)$</td>
<td>$O(n \lg n)$</td>
</tr>
<tr>
<td>Sugimoto+’16</td>
<td>$O(n \lg^2 \sigma)$</td>
<td>$O(n \lg \sigma)$</td>
</tr>
<tr>
<td>Crochemore+’12</td>
<td>$O(n)$</td>
<td>$O(n \lg n)$</td>
</tr>
</tbody>
</table>

- with 2$n$-bit representation: $O(n \lg n)$ bits of working space no longer optimal!
- can we improve working space while not sacrificing time (too much)?
### Our Results

- \( \epsilon > 0 \) selectable constant
- Basic time: \( O(\epsilon^{-1}n) \)
- \( t_{SA} = \log_{\sigma}^\epsilon n \): suffix array query

<table>
<thead>
<tr>
<th>Structure</th>
<th>Bits</th>
<th>Multiplicative Time Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOV LZ or LPnF</td>
<td>( (1 + \epsilon) n \log n + O(n) ) ( O(\epsilon^{-1} n \log \sigma) )</td>
<td>( O(1) ) ( O(t_{SA}) )</td>
</tr>
<tr>
<td>LPnrF</td>
<td>( (2 + \epsilon) n \log n + O(n) ) ( O(\epsilon^{-1} n \log \sigma) )</td>
<td>( O(1) ) ( O(t_{SA}) )</td>
</tr>
<tr>
<td>Reversed LZ</td>
<td>( (2 + \epsilon) n \log n + O(n) ) ( O(\epsilon^{-1} n \log \sigma) )</td>
<td>( O(1) ) ( O(1) )</td>
</tr>
</tbody>
</table>
NOV LZ factorization
NOV LZ factorization

\[ T = a b b a b b a b a b b b a b \]

Coding:
NOV LZ factorization

\[ T = \textbf{a} \textbf{b} \textbf{b} \textbf{b} \textbf{a} \textbf{b} \textbf{b} \textbf{a} \textbf{b} \textbf{a} \textbf{b} \]

Coding: a
NOV LZ factorization

\[ T = \underline{a} \underline{b} \underline{b} \underline{b} \underline{a} \underline{b} \underline{b} \underline{a} \underline{b} \underline{a} \underline{b} \]

Coding: \( ab \)
NOV LZ factorization

\[ T = \text{abbbababab} \]

Coding: \( \text{ab}(2,1) \)
NOV LZ factorization

\[ T = \text{ab b b a b b b a b b a b} \]

Coding: \( ab(2,1)(1,3) \)
NOV LZ factorization

\[ T = ab b b a b b b a b a b \]

Coding: \( ab(2,1)(1,3)(1,2) \)
NOV LZ factorization

\[ T = \text{ab b b a b b b a b a b} \]

Coding: \( \text{ab (2,1) (1,3) (1,2) (1,2)} \)
suffix tree of $T = \text{abbabbabab}$

$T = \text{abbabbabab}$
suffix tree of $T = abbabbabab$

- $T = abbabbabab$
- $\text{pre-order number}$
- $\text{suffix number}$
- $\text{factor starting position}$
- $\text{lowest visited nodes witnesses reference}$
- $\text{previous occurrence must not overlap otherwise trim overlap}$
suffix tree of $T = abbabbbabab$

- descend from root to leaf $\lambda$ with suffix number = factor starting position

$T = abbabbbabab$
suffix tree of $T = abbabbabab$

- descend from root to leaf $\lambda$ with suffix number = factor starting position
- as long as smallest suffix number in subtree is $< sn(\lambda)$

$sn(1) = 1$

$T = ab|bbababab$
suffix tree of $T = \text{abbababab}$

- descend from root to leaf $\lambda$ with suffix number = factor starting position
- as long as smallest suffix number in subtree is $< \text{sn}(\lambda)$

$T = a|b|bab|bababab$
suffix tree of \( T = \text{abbababab} \)

- descend from root to leaf \( \lambda \) with suffix number = factor starting position
- as long as smallest suffix number in subtree is \(< sn(\lambda)\)
- lowest visited nodes witnesses reference
- previous occurrence must not overlap

\[ T = a|b|b|\text{abbabab} \]
suffix tree of $T = \text{abbababbab}$
- descend from root to leaf $\lambda$ with suffix number = factor starting position
- as long as smallest suffix number in subtree is $< \text{sn}(\lambda)$
- lowest visited nodes witnesses reference
- previous occurrence must not overlap
- otherwise trim overlap

$T = a|b|b|\text{abb}|\text{abab}$

factor length = 3

$\text{sn}(3) = 1$

$\text{sn}(6) = 1$
sn(3) = 1
sn(6) = 1

general case:
- $\text{sn}(w)$: smallest suffix number in subtree of node $w$
- strdepth($w$): string depth of node $w$
- node $v$: lowest traversed node
- node $u$: $v$’s parent

length is
- $\min(\text{sn}(\lambda) - \text{sn}(v), \text{strdepth}(v))$
- $\min(\text{sn}(\lambda) - \text{sn}(u), \text{strdepth}(u))$

$\Rightarrow$ take the max of both

$T = a \, b \, b \, abb \, abab$
recap

can compute NOV LZ with
- $O(n)$ calls to $\text{sn}(\cdot)$
- $O(z)$ calls to $\text{strdepth}(\cdot)$

to compute $\text{LPnF}$:
- treat each leaf as a factor starting position
- use suffix links to omit the traversal from the top
  $\Rightarrow O(n)$ calls to $\text{sn}(\cdot)$ and $\text{strdepth}(\cdot)$

how much time costs a call to $\text{sn}(\cdot)$ or $\text{strdepth}(\cdot)$?
suffix trees

combination by Farach-Colton+’00:
- \( O(n) \) time
- \( O(n \lg n) \) bits working space

space-efficient \( O(n) \) time constructions:
- Fischer+’18:
  - \((1 + \epsilon) n \lg n + O(n)\) bits with \( \epsilon \in (0, 1] \)
  - \( t_{SA} = O(1/\epsilon) \) time for \( sn(\cdot) \) and \( strdepth(\cdot) \)
- Munro+’17, Belazzougui+’20:
  - \( O(n \lg \sigma) \) bits
  - \( t_{SA} = \log^\epsilon n \) with SA sampling
  - alternatively: \( strdepth(\cdot) \) in \( O(strdepth(\cdot)) \) time
reversed LZ factorization
reversed LZ

\[ T = \text{abbbabbbab} \]

Coding:
reversed LZ

\[ T = \overline{a b b a b b a b a b} \]

Coding: a
reversed LZ

$$T = \text{abbbabbbabab}$$

Coding: ab
reversed LZ

\[ T = \text{abbbabababab} \]

Coding: \( ab(1,2) \)
reversed LZ

\[ T = \text{ab bb bab bbb bb ab ab} \]

Coding: \( \text{ab}(1,2)(1,3) \)
reversed LZ

\[ T = \text{abbbabbbabab} \]

Coding: \( ab(1,2)(1,3)(3,3) \)
reversed LZ

- $T^R$: reverse of $T$
- use suffix tree of $R := T#T^R$ with $\$ < $\# < c \ \forall c \in \Sigma$
- key lemma: each factor is the string label of a suffix tree node.
key lemma

\[ R = \begin{array}{c}
\text{\hspace{1cm} } \\
\# \\
\text{\hspace{1cm} } \\
\$ \\
\end{array} \]

\[ T \quad \text{\hspace{2cm}} \quad T^R \]

17 / 21
assume factor $F$ ending before an $a$ with
key lemma

$R = \bar{a} F^R F a \# \$

- Assume factor $F$ ending before an $a$ with
- Reference preceded by $\bar{a} \in \Sigma - \{a\}$ (otherwise $F$ can be prolonged)
key lemma

\[ R = \bar{a} F^R F a \# F \bar{a} \$

\begin{align*}
\text{assume factor } F \text{ ending before an } a \text{ with} \\
\text{reference preceded by } \bar{a} \in \Sigma - \{a\} \text{ (otherwise } F \text{ can be prolonged)} \\
\text{mirrored reference in } T^R \text{ ends with } \bar{a} \\
\Rightarrow R \text{ has substrings } Fa \text{ and } F\bar{a} \\
\Rightarrow \exists \text{ node with string label } F \\
(\text{if } F^R \text{ is prefix of } T \Rightarrow F\$ \text{ is suffix of } R)
\end{align*}
cooperative 2 player game

each player takes turns at the same pace

- **Player 2**
  - selects leaves in *descending* suffix number order
  - marks all nodes on the path up to the root

- **Player 1**
  - selects leaves in *ascending* suffix number order
  - if selected leaf $\lambda$ starts with a factor $F$:
    search the lowest marked ancestor $v$ with strdepth($v$) = $|F|$

Player 1 starts.
suffix tree of $R$

leaf with factor starting position

$T = \text{abbabbabab}$
turn of Player 1: nothing marked
⇒ next factor starts at 2

$T = a|bbabbabab$
turn of Player 2: mark all nodes on the path to root

$T = a|bbabbabab$
turn of Player 2: mark all nodes on the path to root

$T = a|bbabbabab$
turn of Player 2: mark all nodes on the path to root

\[ T = a|bbabbabab \]
turn of Player 2:
mark all nodes
on the path to root

$T = a|bbabbbabab$
turn of Player 1: only root marked \( \Rightarrow \) next factor starts at 3

\[ T = a|b|bab|bab|abab \]
turn of Player 2:
mark all nodes on the path to root

$T = a|b|babbabab$
turn of Player 1:  
factor length =  
strdepth(20) = 2  

\[ T = a \| b \| bba \| bbabab \]
termination when both players meet (at symbol #)
- Player 2 never marks a node twice $\Rightarrow \mathcal{O}(n)$ node visits
- Player 1 calls $\mathcal{O}(z)$ times strdepth($w$), where $z = \#$ factors

find source positions in $\mathcal{O}(z \lg n)$ bits:
- second game, but keeping the red marked nodes
- when Player 2 reaches a red marked node from leaf $\lambda$: store there $sn(\lambda)$

note: $z = \mathcal{O}(\log_\sigma n) \Rightarrow \mathcal{O}(z \lg n) \subset \mathcal{O}(n \lg \sigma)$

total:
- $\mathcal{O}(n)$ time for reversed LZ with $\sum_{w \in W} \text{strdepth}(w) = n$, where
  - $W$ : all nodes Player 1 queried
- $\mathcal{O}(nt_{SA})$ time for LPnrF by setting $z \leftarrow n$, and using SA sampling for computing $\text{strdepth}(w)$
open problems

- $O(n \lg \sigma)$ bits + $O(n)$ time possible for computing any LP*F table?
  - Matching statistics can be computed in $O(n \lg \sigma)$ bits and $O(n)$ time because the choice for the reference is static, while all LP*F tables need references based on the already processed text.

- LCP array has notion of irreducible LCP values with a sum of $O(n \lg n)$.
  - Are there irreducible LP*F values? If so, what is their sum?

- $O(r)$ words of space, $r$: runs in the Burrows-Wheeler transform (BWT).

- matching statistics: Bannai, Gagie, I'20
- LPF: Prezza, Rosone'20
- NOV LZ by adaptation of Policriti, Prezza'18:
  - they update the RLBWT of the reversed text while computing a factor
  - instead, keep the RLBWT and update it after determining factor
  - adaptation to LPnF seems hard:
    - need to insert characters and undo these insertions in RLBWT
    - $\sum_{i=1}^{n} LPnF[i] = O(n^2)$ steps

- for LPnF or LPnrF open