Computing LZ78-Derivates with Suffix Trees

Dominik Köppl

Faculty of Engineering, University of Yamanashi

$T = \text{ababbbababbbabbb}$

Coding: $(a,b) (1,b) (1,2) (1,a)$
setting

text factorization
- input: text $T$ with length $n$
- output: factorization of $T$
examples of factorizations
- LZ77
- LZ78
- Lyndon factorization
goal: compute factorization in $O(n)$ time

substring compression
- index $T$ in a preprocessing step
- query: interval $[i..j] \subset [1..n]$
- output: factorization of $T[i..j]$
goal:
- query time linear to output size (output sensitive)
- index time linear in input size ($O(n)$ time)
why restricting index time?

trivial solution for substring compression:
  - compute and store the factorizations of all $\Theta(n^2)$ substrings
  - answer a query in $O(1)$ via lookup
  - however: index space is $\Omega(n^2)$ (hence time is also $\Omega(n^2)$)
work on substring factorization

<table>
<thead>
<tr>
<th>factorization</th>
<th>construction time</th>
<th>query time</th>
<th>reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>LZ77</td>
<td>$O(n \lg n)$</td>
<td>$O(z \lg n \lg \lg n)$</td>
<td>Cormode+’05</td>
</tr>
<tr>
<td>LZ77</td>
<td>$O(n \lg n)$</td>
<td>$O(z \lg n)$</td>
<td>Keller+’14</td>
</tr>
<tr>
<td>Lyndon</td>
<td>$O(n \lg n)$</td>
<td>$O(z)$</td>
<td>Babenko+’14</td>
</tr>
<tr>
<td>Lyndon</td>
<td>$O(n)$</td>
<td>$O(z)$</td>
<td>Kociumaka’16</td>
</tr>
<tr>
<td>LZ78</td>
<td>$O(n)$</td>
<td>$O(z)$</td>
<td>Köppl’21</td>
</tr>
<tr>
<td>LZD/LZMW</td>
<td>$O(n)$</td>
<td>$O(z)$</td>
<td>this talk</td>
</tr>
</tbody>
</table>

$z$: output size of respective factorization
factorizations in this talk

LZ78 derivations

- Lempel–Ziv Double (LZD) Goto’15
- Lempel–Ziv-Miller–Wegman (LZMW) Miller+’85

why?

- number of LZ78 factors is lower bounded by $\Omega(\sqrt{n})$
- in contrast, the lower bound for LZD and LZMW is $\Omega(\lg n)$
lower bound for LZ78

\[ T = a a a a a a a a a a a a \cdots \]

\[ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12 \]

coding:

since the length \(|F_x|\) of the \(x\)-th factor is \(x\), \(\sum_{x=1}^{z} |F_x| = n \iff z \in \Theta(\sqrt{n})\)
lower bound for LZ78

\[ T = \text{a a a a a a a a a a a a a a a a a a a a a a a a} \cdots \]

1 2 3 4 5 6 7 8 9 10 11 12

Coding: a

since the length \(|F_x|\) of the \(x\)-th factor is \(x\), \(\sum_{x=1}^{z} |F_x| = n \Leftrightarrow z \in \Theta(\sqrt{n})\)
lower bound for LZ78

\[ T = aaaa aaaa aaaa aaaa aaaa aaaa aaaa \cdots \]

1 2 3 4 5 6 7 8 9 10 11 12

coding: a(1, a)

since the length \( |F_x| \) of the \( x \)-th factor is \( x \),

\[ \sum_{x=1}^{z} |F_x| = n \iff z \in \Theta(\sqrt{n}) \]
lower bound for LZ78

\[ T = \textcolor{gray}{a} \underbrace{\textcolor{red}{a} \textcolor{blue}{a} \textcolor{blue}{a} \textcolor{blue}{a} \textcolor{blue}{a}}_{\text{1-4}} \textcolor{green}{a} \textcolor{red}{a} \textcolor{green}{a} \textcolor{green}{a} \textcolor{green}{a} \textcolor{green}{a} \textcolor{green}{a} \textcolor{green}{a} \textcolor{green}{a} \textcolor{green}{a} \textcolor{green}{a} \textcolor{green}{a} \textcolor{green}{a} \textcolor{green}{a} \cdots \]

\[ a(1,a)(2,a) \]

coding: \( a(1,a)(2,a) \)

since the length \( |F_x| \) of the \( x \)-th factor is \( x \), \( \sum_{x=1}^{z} |F_x| = n \iff z \in \Theta(\sqrt{n}) \)
lower bound for LZ78

\[ T = aa \underbrace{aaa \ldots}_{\text{12}} a a \ldots \]

\[ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12 \]

coding: \( a(1,a)(2,a)(3,a) \)

since the length \( |F_x| \) of the \( x \)-th factor is \( x \),
\[ \sum_{x=1}^{z} |F_x| = n \iff z \in \Theta(\sqrt{n}) \]
lower bound for LZ78

\[ T = \text{aaa} \text{aaa} \text{aaa} \text{aaa} \text{aaa} \text{aaa} \text{aaa} \text{aaa} \text{aaa} \text{aaa} \text{a} \text{a} \cdots \]

\begin{align*}
(1,a) \\
(2,a) \\
(3,a) \\
(4,a)
\end{align*}

coding: \( a(1,a)(2,a)(3,a)(4,a) \)

since the length \( |F_x| \) of the \( x \)-th factor is \( x \), \( \sum_{x=1}^{z} |F_x| = n \Leftrightarrow z \in \Theta(\sqrt{n}) \)
lower bound for LZD

\[ T = a a a a a a a a a \cdots \]

\[ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12 \]

coding:

since the length \( |F_x| \) of the \( x \)-th factor is \( 2^x \), \( \sum_{x=1}^{z} |F_x| = n \iff z \in \Theta(\lg n) \)
lower bound for LZD

\[ T = \text{a a a a a a a a a a a a a a a a a a ...} \]

1 2 3 4 5 6 7 8 9 10 11 12

\text{coding: (a,a)}

since the length \(|F_x|\) of the \(x\)-th factor is \(2^x\), \(\sum_{x=1}^{z} |F_x| = n \iff z \in \Theta(\lg n)\)
lower bound for LZD

$T = \text{aaaaaa}\text{aaaaaa}\cdots$

1 2 3 4 5 6 7 8 9 10 11 12

coding: $(a,a)(1,1)$

since the length $|F_x|$ of the $x$-th factor is $2^x$, $\sum_{x=1}^{z} |F_x| = n \Leftrightarrow z \in \Theta(\lg n)$
lower bound for LZD

\[ T = \text{aaaaaaa} \text{aaaaaaa...} \]

\[ T = \text{a a a a a a} \text{a a a a a a a a} \cdots \]

\[
\begin{align*}
\text{coding: } (a,a)(1,1)(2,2)
\end{align*}
\]

since the length \(|F_x|\) of the \(x\)-th factor is \(2^x\),
\[
\sum_{x=1}^{\infty} |F_x| = n \iff z \in \Theta(\lg n)
\]
lower bound for LZMW

\[ T = \text{coding: } a a a a a a a a a a a a a a a a \cdots \]

since the length \( |F_x| \) of the \( x \)-th factor is the \( x - 1 \)st Fibonacci number,

\[ \sum_{x=1}^{z} |F_x| = n \iff z \in \Theta(\lg n) \]
Since the length \( |F_x| \) of the \( x \)-th factor is the \( x - 1 \)st Fibonacci number, 
\[
\sum_{x=1}^{z} |F_x| = n \iff z \in \Theta(\lg n)
\]
since the length $|F_x|$ of the $x$-th factor is the $x - 1$st Fibonacci number, 
\[ \sum_{x=1}^{z} |F_x| = n \iff z \in \Theta(\lg n) \]
lower bound for LZMW

$coding: \text{ aa2}$

since the length $|F_x|$ of the $x$-th factor is the $x - 1$st Fibonacci number,

$\sum_{x=1}^{z} |F_x| = n \iff z \in \Theta(\lg n)$
lower bound for LZMW

$$T = \text{aaaaaa aaaa a...}$$

1 2 3 4 5 6 7 8 9 10 11 12

coding: \text{aa23}

since the length $|F_x|$ of the $x$-th factor is the $x - 1$st Fibonacci number,

$$\sum_{x=1}^{z} |F_x| = n \iff z \in \Theta(\lg n)$$
lower bound for LZMW

\[ T = \text{aa a aa a a a a a a a a a a a} \cdots \]

\[ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12 \]

\[ T = \text{aa a aa a a a a a a a a a a} \cdots \]

\[ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12 \]

coding: \text{aa234}

since the length \( |F_x| \) of the \( x \)-th factor is the \( x - 1 \)st Fibonacci number,

\[ \sum_{x=1}^{z} |F_x| = n \iff z \in \Theta(\lg n) \]
definition of LZD

each factor represented as a pair

- element is either a character or the index of a former factor
- greedily maximize the length by the first element first

let dst$_x$ denote the starting position of $F_x$ in $T$.

formal definition

A factorization $F_1 \cdots F_z$ of $T$ is LZD if

- $F_x = G_1 \cdot G_2$ with
- $G_1, G_2 \in \{F_1, \ldots, F_{x-1}\} \cup \Sigma$ such that
- $G_1$ and $G_2$ are respectively the longest possible prefixes of $T[dst_x..]$ and of $T[dst_x + |G_1|..]$. 
example for LZD

\[ T = \text{ababbbabbababbb} \]

coding:
example for LZD

\[ T = \textcolor{gray}{a} \textcolor{black}{b} \textcolor{gray}{a} \textcolor{black}{b} \textcolor{gray}{b} \textcolor{black}{a} \textcolor{gray}{b} \textcolor{black}{a} \textcolor{gray}{b} \textcolor{black}{b} \textcolor{gray}{a} \textcolor{black}{b} \textcolor{gray}{b} \]  

\text{coding: } (a,b)
example for LZD

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

\[ (1, b) \]

\[ \text{coding: } (a, b)(1, b) \]
example for LZD

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

\[ \text{coding: } (a,b)(1,b)(1,2) \]
example for LZD

\[ T = \begin{array}{cccccccccccc}
  & a & b & a & b & b & b & b & a & b & a & b & b & b \\
 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12
\end{array} \]

coding: \((a,b)(1,b)(1,2)(1,a)\)
definition of LZMW

- has like LZD two references
- however references need to be successive
- thus needs to store only one reference to a former factor index

formal definition

A factorization $F_1 \cdots F_z$ of $T$ is LZMW if $F_x$ is the longest prefix of $T[dst_x..]$ with $F_x \in \{F_{y-1}F_y : y \in [2..dst_x - 1]\} \cup \Sigma$, for every $x \in [1..z]$. 
example for LZMW

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

1 2 3 4 5 6 7 8 9 10 11 12

coding:
example for LZMW

\[ T = \text{ababaabababaabb} \]

\text{coding: a}
example for LZMW

\[ T = \text{ababbbababababb} \]

Coding: \text{ab}
example for LZMW

\[ T = \text{ababababbabababb} \]

Coding: \text{ab2}
example for LZMW

\[ T = \text{abababbabbabbabb} \]

\[ T = \text{abababbabbabb} \]

coding: \text{ab23}
example for LZMW

\[ T = \text{abababbbababab} \]

\[ \text{coding: ab234} \]
example for LZMW

\[ T = \text{ababababbabababb} \]

coding: \text{ab234b}
**LZD and LZW computation**

<table>
<thead>
<tr>
<th>Time</th>
<th>Space</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n \log \sigma)$</td>
<td>$O(n)$</td>
<td>Goto+’15</td>
</tr>
<tr>
<td>$\Omega(n^{5/4})$</td>
<td>$O(z)$</td>
<td>Goto+’15, Badkobeh+’17 where</td>
</tr>
<tr>
<td>$O(n + z \log^2 n)$ expected</td>
<td>$O(z)$</td>
<td>Badkobeh+’17</td>
</tr>
<tr>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>this talk</td>
</tr>
</tbody>
</table>

- Goto+’15 only computes LZD
- $\sigma = n^{O(1)}$ means that integer alphabets are supported
our contributions

- for the whole text, we can compute LZD and LZMW in $O(n)$ time and space
- compute the substring compression of LZD and LZMW with
  - $O(n)$ index time for preprocessing
  - $O(z)$ query time
- setting
  - $n$: length of the input
  - integer alphabet
  - word RAM
tools

for computation, we leverage the following toolbox

- **suffix tree ST Weiner’73**
  - linear-time construction of ST Farach-Colton’00

- **weighted ancestor query data structure Gawrychowski’14**
  - find an ancestor with string depth $d$ of any ST node and any $d$ in $O(1)$ time
  - constructable in linear time Belazzougui’21

- **lowest marked ancestor data structure Cole+’05**
  - can mark any ST node in $O(1)$ time
  - can find the lowest marked ancestor of any ST node in $O(1)$ time

sum of needed space and time amounts to $O(n)$ each

how used for LZD computation?
suffix tree of $T$ = abababbababb
suffix tree of $T\$ = abababbababbabb

$T = abababbababbabb$
suffix tree of $T = \text{ababba}\text{babb}$

ST root represents empty factor

$T = \text{ababba}\text{babb}$
suffix tree of $T$ = abababababbabb

- ST root represents empty factor
- compute pair $F_1 = (e_L, e_R)$ of first factor
- suffix number of $\lambda_1$ is dst$_1 = 1$
- lowest marked ancestor of $\lambda_1$ is ST root, so $e_L = T[1] = a$

$T = abababababbabb$
suffix tree of \( T \) = abababababbabb

- ST root represents empty factor
- compute pair \( F_1 = (e_L, e_R) \) of first factor
- suffix number of \( \lambda_1 \) is \( dst_1 = 1 \)
- lowest marked ancestor of \( \lambda_1 \) is ST root, so \( e_L = T[1] = a \)
- \( \lambda_2 \) is leaf with suffix number 2

\[ T = abababababbabb \]
suffix tree of $T = ab|ab|ab|ab|ab|ab|ab|ab|ab|ab|ab|ab|ab|ab|ab|ab|ab|ab$

- ST root represents empty factor
- compute pair $F_1 = (e_L, e_R)$ of first factor
- suffix number of $\lambda_1$ is $dst_1 = 1$
- lowest marked ancestor of $\lambda_1$ is ST root, so $e_L = T[1] = a$
- $\lambda_2$ is leaf with suffix number 2
- lowest marked ancestor of $\lambda_2$ is ST root, so $e_R = T[2] = b$
- mark ancestor of $\lambda_1$ with string depth 2 with 1

$T = ab|ab|ab|ab|ab|ab|ab|ab|ab|ab|ab|ab|ab|ab|ab|ab|ab|ab$

$\lambda_1 = 5|6$

$\lambda_2 = 2|7$
suffix tree of $T$ = ababababababb

process $F_2$

- suffix number of $\lambda_1$ is $dst_2 = 3$
- lowest marked ancestor of $\lambda_1$ is 3, so $e_L = 1$ (mark of 3)

$T = \text{ab|abbabababbb}$
suffix tree of $T = ababbababbabb$

process $F_2$

- suffix number of $\lambda_1$ is $\text{dst}_2 = 3$
- lowest marked ancestor of $\lambda_1$ is 3, so $e_L = 1$ (mark of 3)
- like before, $e_R = T[2] = b$
- mark ancestor of $\lambda_1$ with string depth $|F_2| = 3$ with 2

$T = ab|abb|ababbabb$
suffix tree of $T$ = abababababab

process $F_3$

- suffix number of $\lambda_1$ is $\text{dst}_3 = 6$
- lowest marked ancestor of $\lambda_1$ is 3, so $e_L = 1$ (mark of 3)

$T = ab|abb|abababb$
suffix tree of $T$ = abab|ababb|ababb|abb

process $F_3$

- suffix number of $\lambda_1$ is dst$_3 = 6$
- lowest marked ancestor of $\lambda_1$ is 3, so $e_L = 1$ (mark of 3)
- lowest marked ancestor of $\lambda_2$ is 7, so $e_L = 2$ (mark of 2)
- however: ancestor of $\lambda_1$ with string depth $|F_3| = 5$ does not exist!
suffix tree of $T_\$$ = ababbababbabb

maintaining reference for $F_3$

- locus of $F_3$ can be witnesses by node 4
- let node 4 store length of $F_3$; mark node 4

$T = ab|abb|ababb|abb$
time complexity

for processing $F_x$

- take leaf $\lambda_1$ corresponding to the starting position $\text{dst}_x$ of $F_x$
- compute the lowest marked ancestor $v_1$ of $\lambda_1$
- given $\ell_1$ is the string length of $v_1$, take leaf $\lambda_2$ having suffix number $\text{dst}_x + \ell_1$
- compute the lowest marked ancestor $v_2$ of $\lambda_2$
- length of $F_x$ is $\ell_1 + \ell_2$, where $\ell_2$ is the string length of $v_2$
- if $v_1$ (or $v_2$) refers to an implicit node, use the stored length instead of $\ell_1$ (or $\ell_2$)

each step takes $O(1)$ time, so we have $O(z)$ total time, where $z$ is the number of processed factors
LZMW computation works similarly
- mark the locus of $F_{x-1}F_x$ instead of $F_x$
- need only one lowest marked ancestor query ($v_2$ not needed)
can compute LZD and LZMW in $O(n)$ time, in the computational model
- $n$: length of the input
- alphabet can be integer
- word RAM

for substring compression:
- $O(n)$ index time
- $O(z)$ query time, where $z$ is the number of factors to output

Thank you for listening. Any questions are welcome!