Pattern Matching on Compressed Texts II

Shunsuke Inenaga
Kyushu University, Japan
Agenda

- Fully Compressed Pattern Matching
- Straight Line Program
- Compressed String Comparison
- Period of Compressed String
- Pattern Discovery from Compressed String (Palindrome and Square)
- FCPM for 2D SLP
- Open Problems
Fully Compressed Pattern Matching [1/3]

compressed pattern: \&(aG

compressed text:

geoiy083qa0gj(#*gpfomo)#(JGWRE$(U)%ARY)(JPED(A%RJG)ER%U)JGODAAQWT$JGWRE)$REWJFDOPJJKSeoiy083qa0gj(#*gpfomo)#(JGWRE$(U)%ARY)(JPED(A%RJG)ER%U)JGODAAQWT$JGWRE)$geoiy083qa0gj(#*gpfomo)#(JGWRE$(U)%ARY)(JPED(A%RJG)ER%U)JGODAAQWT$JGWRE)$geoiy083qa0gj(#*gpfomo)#(JGWRE$(U)%ARY)(JPED(A%RJG)ER%U)JGODAAQWT$JGWRE)$
Fully Compressed Pattern Matching [2/3]

- **uncompressed** text
  - uncompressed pattern

- **compressed** text
  - uncompressed pattern

- **compressed** text
  - compressed pattern

- Classical pattern matching algorithm

- **compressed pattern** matching algorithm

- Fully compressed pattern matching algorithm
Possible Application of FCPM

compressed text

I’m here.

compressed pattern
**Fully Compressed Pattern Matching [3/3]**

**FCPM Problem**

**Input**: \( T = \text{compress}(T) \) and \( P = \text{compress}(P) \).

**Output**: Set \( \text{Occ}(T, P) \) of substring occurrences of pattern \( P \) in text \( T \).

- \( \text{Occ}(T, P) = \{ |u| + 1 : T = uPw, \ u, w \in \Sigma^* \} \)
Straight Line Program [1/2]

**SLP** $T$: sequence of assignments

$$X_1 = expr_1 ; X_2 = expr_2; \ldots ; X_n = expr_n;$$

$X_k$: variable,

$\begin{align*}
expr_k : \quad & a \quad (a \in \Sigma) \\
& X_iX_j \quad (i, j < k).
\end{align*}$

SLP $T$ for string $T$ is a CFG in Chomsky normal form s.t. $L(T) = \{T\}$. 
Straight Line Program [2/2]

\[ X_1 = a \]
\[ X_2 = b \]
\[ X_3 = X_1X_2 \]
\[ X_4 = X_3X_1 \]
\[ X_5 = X_3X_4 \]
\[ X_6 = X_5X_5 \]
\[ X_7 = X_4X_6 \]
\[ X_8 = X_7X_5 \]

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

\[ N = O(2^n) \]
Straight Line Program [2/2]

\[ T = \overbrace{a \ b \ a \ a \ b \ a \ b \ a \ a \ b \ a \ b \ a \ a \ b \ a \ a \ b}^{N} \]

\[ N = O(2^n) \]
From LZ77 to SLP

For any string $T$ given in LZ77-compressed form of size $k$, an SLP generating $T$ of size $O(k^2)$ can be constructed in $O(k^2)$ time.

[Rytter ’00, ’03, ’04]
FCPM for SLP

FCPM Problem for SLP

**Input**: SLP $T$ for text $T$ and SLP $P$ for pattern $P$.

**Output**: Compact representation of set $Occ(T, P)$ of substring occurrences of $P$ in $T$.

- We want to solve the problem **efficiently** (i.e., polynomial time & space in $n$ and $m$).
  - $n =$ the size of SLP $T$, $m =$ the size of SLP $P$

- $|T| = O(2^n) \implies T$ (also $P$) cannot be decompressed
- $|Occ(T,P)| = O(2^n) \implies$ compact representation
Key Definition

\[ \text{Occ}^A(X, Y) = \{ i \in \text{Occ}(X, Y) \mid |X_l| - |Y| \leq i \leq |X_l| \} \]

set of occurrences of \( Y \) that cover or touch the boundary of \( X_l \) and \( X_r \).

\( X \): variable of \( T \)
\( Y \): variable of \( P \)
Key Lemma

\[ \text{Obs}^\Delta(X, Y) \text{ forms a single arithmetic progression.} \]

[Miyaazaki et al. ’97]
Key Observation

\[
\text{Occ}(X, Y) = \text{Occ}(X_l, Y) \cup \text{Occ}^A(X, Y) \cup \text{Occ}(X_r, Y) \oplus |X_l|
\]

[Myazaki et al. ’97]

Computing \(\text{Occ}(X, Y)\) is reduced to computing \(\text{Occ}^A(X, Y)\).
Compact representation of $\text{Occ}(T, P)$ which answers a membership query to $\text{Occ}(T, P)$ in $O(n)$ time.

**DP for $\text{Occ}^\wedge(X_i, Y_j)$**

<table>
<thead>
<tr>
<th>$X_n$</th>
<th>$\text{Occ}^\wedge(X_n, Y_1)$</th>
<th>$\text{Occ}^\wedge(X_n, Y_j)$</th>
<th>$\text{Occ}^\wedge(X_n, Y_m)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_i$</td>
<td>$\text{Occ}^\wedge(X_i, Y_1)$</td>
<td>$\text{Occ}^\wedge(X_i, Y_j)$</td>
<td>$\text{Occ}^\wedge(X_i, Y_m)$</td>
</tr>
<tr>
<td>$X_1$</td>
<td>$\text{Occ}^\wedge(X_1, Y_1)$</td>
<td>$\text{Occ}^\wedge(X_1, Y_j)$</td>
<td>$\text{Occ}^\wedge(X_1, Y_m)$</td>
</tr>
<tr>
<td></td>
<td>$Y_1$</td>
<td>$Y_j$</td>
<td>$Y_m$</td>
</tr>
</tbody>
</table>
## Known Results

<table>
<thead>
<tr>
<th></th>
<th>Time</th>
<th>Space</th>
<th>Compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Miyazaki et al. ’97</td>
<td>$O(m^2n^2)$</td>
<td>$O(mn)$</td>
<td>SLP</td>
</tr>
<tr>
<td>Lifshits ’07</td>
<td>$O(mn^2)$</td>
<td>$O(mn)$</td>
<td>SLP</td>
</tr>
<tr>
<td>Hirao et al. ’00</td>
<td>$O(mn)$</td>
<td>$O(mn)$</td>
<td>Balanced SLP</td>
</tr>
</tbody>
</table>

**Balanced SLP**

![Balanced SLP Diagram]
Fully Compressed Subsequence Pattern Matching [1/2]

FC Subsequence PM Problem

**Input**: SLP $T$ for text $T$ and SLP $P$ for pattern $P$.

**Output**: Find whether $P$ is a subsequence of $T$.

- $P$ is said to be a subsequence of $T$, if $P$ can be obtained by removing zero or more characters from $T$. 

Fully Compressed Subsequence Pattern Matching [2/2]

The Fully Compressed Subsequence Pattern Matching Problem on SLP compressed strings is NP-hard.

[Lifshits & Lohrey ’06]
Compressed String Comparison [1/2]

**CSC Problem**

**Input** : SLPs $T$ and $S$ for strings $T$ and $S$, resp.

**Output** : $\text{Dis(similarity)}$ of $T$ and $S$. 
## Compressed String Comparison [2/2]

<table>
<thead>
<tr>
<th>Measure</th>
<th>Time</th>
<th>Space</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equality</td>
<td>$O(mn^2)$</td>
<td>$O(mn)$</td>
<td>Lifshits ’07</td>
</tr>
<tr>
<td>Hamming Distance</td>
<td>#P-complete</td>
<td>PSPACE</td>
<td>Lifshits ’07</td>
</tr>
<tr>
<td>Longest Common Substring</td>
<td>$O((m+n)^4\log(m+n))$</td>
<td>$O((m+n)^3)$</td>
<td>Matsubara et al. ’08</td>
</tr>
<tr>
<td>Longest Common Subsequence</td>
<td>NP-hard</td>
<td>PSPACE</td>
<td>Lifshits &amp; Lohrey ’06</td>
</tr>
</tbody>
</table>
Property of common substrings [1/3]

- For each common substring $Z$ of string $S$ and $T$, there always exists a variable $X_i = X_lX_r$ and $Y_j = Y_LY_R$ such that:
  - $Z$ is a common substring of $X_i$ and $Y_j$
  - $Z$ contains an overlap between $X_l$ and $Y_R$
For each common substring $Z$ of string $S$ and $T$, there always exists a string $w$ such that:
- $w$ is a substring of $Z$
- $w$ is an overlap of variables of $S$ and $T$
Property of common substrings [1/3]

- For each common substring $Z$ of string $S$ and $T$, there always exists a string $w$ such that:
  - $Z$ can be calculated by expanding $w$
Computing Overlaps

Lemma [Karpinski et al. ’97]
For any variables $X_i$ and $X_j$ of SLP $T$, $OL(X_i, X_j)$ can be represented by $O(n)$ arithmetic progressions.

Theorem [Karpinski et al. ’97]
For any SLP $T$, $OL(X_i, X_j)$ can be computed in total of $O(n^4 \log n)$ time and $O(n^3)$ space for each $i, j$. 

Compressed Period Problem

**Input**: SLP $T$ for string $T$.

**Output**: Compact representation of set $\text{Period}(T)$ of periods of $T$.

- $\text{Period}(T) = \{ |T| - |u| : T = uv = wu, \nu, w \in \Sigma^+ \}$
An $O(n)$-size representation of $\text{Period}(T)$ can be computed in $O(n^4)$ time with $O(n^3)$ space.

[Lifshits ’06, ’07]
Compressed Palindrome Discovery Problem

**Input**: SLP $T$ for string $T$.

**Output**: Compact representation of set $Pal(T)$ of maximal palindromes of $T$.

- $Pal(T) = \{(p,q) : T[p:q] \text{ is the maximal palindrome centered at } \lfloor (p+q)/2 \rfloor \}$

- ex. $T = baabbaa$
An $O(n^2)$-size representation of $\textit{Pal}(T)$ can be computed in $O(n^4)$ time with $O(n^2)$ space.

[Matsubara et al. ’08]
Composition System

CS \( T \): sequence of assignments

\[ X_1 = expr_1 ; X_2 = expr_2; \ldots ; X_n = expr_n; \]

\( X_k \): variable,

\[ expr_k : \begin{cases} a & ( a \in \Sigma ), \\ X_i X_j & ( i, j < k ), \\ \left[p\right] X_i X_j \left[q\right] & ( i, j < k ). \end{cases} \]

- \([p]X = X[1:p]\]
- \(X[q] = X[|X|-q+1:|X|]\)
From LZ77 to CS

For any string \( T \) given in LZ77-compressed form of size \( k \), a CS generating \( T \) of size \( O(k \log k) \) can be constructed in polynomial time.

[Gasieniec et al. ’96]
Compressed Square Discovery [1/2]

Compressed Square Problem

**Input**: CS $T$ for string $T$.

**Output**: Check the square freeness of $T$ (whether $T$ contains a square or not).

- A square is any non-empty string of the form $xx$. 
We can test square freeness of $T$ in polynomial time in the size of given composition system $T$.

[Gasieniec et al. ’96, Rytter’00]
2D SLP

2D SLP $T$: sequence of assignments

$$X_1 = expr_1; \ X_2 = expr_2; \ldots; \ X_n = expr_n;$$

$X_k$ : variable,

$$expr_k : \begin{cases} 
\ a & (a \in \Sigma), \\
X_i \oplus X_j & (i, j < k, \ \text{height}(X_i) = \text{height}(X_j)), \\
X_i \boxplus X_j & (i, j < k, \ \text{width}(X_i) = \text{width}(X_j)).
\end{cases}$$

$$X_k = \begin{array}{c}
X_i \\
X_j
\end{array}$$  \hspace{1cm} horizontal concatenation $\oplus$

$$X_k = \begin{array}{c}
X_i \\
X_j
\end{array}$$  \hspace{1cm} vertical concatenation $\boxplus$
The Fully Compressed Pattern Matching Problem for 2D SLP is $\Sigma_2^P$-complete.

[Berman et al. ’97, Rytter’00]
Open Problems [1/2]

- Edit distance of two SLP-compressed strings.

- Compact representation of all maximal runs of an SLP-compressed string.
  - A run is any string $x$ whose minimal period $p$ satisfies $p \leq \frac{|x|}{2}$.
  - ex. $\overline{8} \quad (aab)^3 = aabaabaa$
Max Number of Runs in a String

N: (uncompressed) text length

0.927N
[Franek et al. ’03]

0.944565N
[Kusano et al. ’08]

1.048N
[Crochemore et al. ’08]

0.927N
[Franek et al. ’03]

1.00N

1.05N

3.44N
[Rytter ’07]

3.48N
[Puglisi et al. ’08]

5N
[Rytter ’06]

5N
[Kolpakov & Kucherov ’99]
Open Problems [2/2]

- Fully Compressed Tree Pattern Matching for grammar based XML compression.
  - TGCA (Tree Grammar Compression Algorithm) [Onuma et al. ’06]
References [1/5]


References [2/5]


References [3/5]


References [4/5]


References [5/5]


