Decompressing Lempel-Ziv Compressed Text

Philip Bille, DTU
Mikko Berggren Ettienne, DTU
Travis Gagie, Dalhousie
Inge Li Gørtz, DTU
Nicola Prezza, LUISS

DCC 2020
Decompressing LZ77

background
motivation
setting
past solutions

step 1
$S^T$ definition
$S^T$ extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching

audio
Right now, when *everything* is being done online, *compression* is important — but since there’s even more downloading than uploading, *decompression* is even more important!

LZ77 is one of the most elegant, powerful and — thanks to its inclusion in several standards and tools such as gzip — popular compression schemes.

There have been several recent papers on small-space LZ77 compression but almost nothing is known about small-space LZ77 decompression.
Suppose we’re given the z-phrase parse $Z$ of a string $S[1..n]$ over $\{0, \ldots, \sigma - 1\}$.

In this talk we consider the version of LZ77 in which a phrase can be

- a single character $S[i] \neq S[i']$ for all $i' < i$, encoded as $(S[i], 0)$;
- a substring $S[i..j] = S[i'..j']$ for some $j' < i$, encoded as $(i, j - i + 1)$.

Our results can be generalized to other versions as well.
For example, if

\[ S = 010011000110100110001101001100 \]

then \( z = 8 \) and

\[ Z = (0, 0), (1, 0), (1, 1), (1, 2), (2, 3), (4, 4), (2, 8), (10, 10) . \]
For example, if

$$S = 010011000110100110001101001100$$

then \( z = 8 \) and

$$Z = (0, 0), (1, 0), (1, 1), (1, 2), (2, 3), (4, 4), (2, 8), (10, 10).$$
For example, if

\[ S = 010011000110100110001101001100 \]

then \( z = 8 \) and

\[ Z = (0, 0), (1, 0), (1, 1), (1, 2), (2, 3), (4, 4), (2, 8), (10, 10). \]
For example, if

\[ S = 010011000110100110001101001100 \]

then \( z = 8 \) and

\[ Z = (0, 0), (1, 0), (1, 1), (1, 2), (2, 3), (4, 4), (2, 8), (10, 10). \]
For example, if

\[ S = 010011000110100110001101001100 \]

then \( z = 8 \) and

\[ Z = (0, 0), (1, 0), (1, 1), (1, 2), (2, 3), (4, 4), (2, 8), (10, 10) . \]
For example, if

\[ S = 010011000110100110001101001100 \]

then \( z = 8 \) and

\[ Z = (0, 0), (1, 0), (1, 1), (1, 2), (2, 3), (4, 4), (2, 8), (10, 10). \]
For example, if

\[ S = 010011000110100110001101001100 \]

then \( z = 8 \) and

\[ Z = (0, 0), (1, 0), (1, 1), (1, 2), (2, 3), (4, 4), (2, 8), (10, 10). \]
For example, if

\[ S = 010011000110100110001101001100 \]

then \( z = 8 \) and

\[ Z = (0, 0), (1, 0), (1, 1), (1, 2), (2, 3), (4, 4), (2, 8), (10, 10) \.]
For example, if

\[ S = 010011000110100110001101001100 \]

then \( z = 8 \) and

\[ Z = (0, 0), (1, 0), (1, 1), (1, 2), (2, 3), (4, 4), (2, 8), (10, 10). \]
past solutions I

The standard solution is to decompress $S$ from left to right, phrase by phrase, which takes $O\left(z + n \cdot \frac{\log \sigma}{\log n}\right)$ time and space.

Another solution is to turn $Z$ into a context-free grammar for $S$ and generate $S$ from that, which takes $O\left(z \log \frac{n}{z} + n \cdot \frac{\log \sigma}{\log n}\right)$ time and $O\left(z \log \frac{n}{z}\right)$ space.
Puglisi and Rossi’s DCC 2019 solution was a simplified implementation of our arXiv poster.

It doesn’t have good worst-case time bounds but works well in practice.

(Rossi’s slides: https://tinyurl.com/ub6yyxd.)
Decompressing LZ77

background
motivation
setting
past solutions

step 1
\( S^T \) definition
\( S^T \) extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching
We first want to extract and store the subsequence $S^\tau$ of $S$ consisting of characters within distance $\tau$ of the nearest phrase boundary, where $\tau$ is a parameter.

For example, if $\tau = 2$ and

$$ S = 0 1 0 0 1 100 0110 10011000 1101001100 $$

then

$$ S^\tau = 0 1 0 0 1 100 0110 1000 1100 . $$
Decompressing LZ77

**Background**
- Motivation
- Setting
- Past solutions

**Step 1**
- $S^τ$ definition
- $S^τ$ extraction
- Mergeable dictionary

**Step 2**
- Substring division
- Substring extraction

**Applications**
- Decompression
- Pattern matching

$S^τ$ extraction
audio

```
0 1 0 0 1 1 0 0 0 1 1 0 1 0 0 1 1 0 0
```
Decompressing LZ77

background
motivation
setting
past solutions

step 1
\(S^T\) definition
\(S^T\) extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching

\(S^T\) extraction
audio
Decompressing LZ77

background
motivation
setting
past solutions

step 1
$S^T$ definition
$S^T$ extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching

$S^T$ extraction
audio
Decompressing
LZ77

background
motivation
setting
past solutions

step 1
$S^T$ definition
$S^T$ extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching

$S^T$ extraction
audio
Decompressing LZ77

background
motivation
setting
past solutions

step 1
$S^T$ definition
$S^T$ extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching

$S^T$ extraction
audio
Decompressing LZ77

background
motivation
setting
past solutions

step 1
$S^T$ definition
$S^T$ extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching

$S^T$ extraction
audio
Decompressing LZ77

$S^τ$ extraction

1. $S^τ$ definition
2. $S^τ$ extraction

Applications
- Decompression
- Pattern matching

Audio

Background
- Motivation
- Setting
- Past solutions

Step 1
- $S^τ$ definition
- $S^τ$ extraction

Mergeable dictionary

Step 2
- Substring division
- Substring extraction
Decompressing LZ77

$S^T$ extraction
audio

background
motivation
setting
past solutions

step 1
$S^T$ definition
$S^T$ extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching
Decompressing LZ77

**Motivation**
- Background
- Setting
- Past solutions

**Step 1**
- $S^T$ definition
- $S^T$ extraction
- Mergeable dictionary

**Step 2**
- Substring division
- Substring extraction

**Applications**
- Decompression
- Pattern matching

$S^T$ extraction
Audio
Decompressing LZ77

background
motivation
setting
past solutions

step 1
$S^T$ definition
$S^T$ extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching

$S^T$ extraction
audio
Decompressing LZ77

background
motivation
setting
past solutions

step 1
$S^T$ definition
$S^T$ extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching

$S^T$ extraction
audio

0 1 0 0 1 1 0 0 1 1 0 1 0 0 0 1 1 1 1 0 0
Decompressing LZ77

background
motivation
setting
past solutions

step 1
$S^T$ definition
$S^T$ extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching

$S^T$ extraction
audio
mergeable dictionary

In my graphics editor, I can just select all the dots in a box that’s $\tau$ shorter than a phrase and move them to the phrase’s source.

Iacono and Özkan’s *mergeable dictionary* has the same functionality now that Bille et al. have shown how to implement shifts.

If $\tau = O \left( \frac{\log n}{\log \sigma} \right)$ — so $\tau$ characters fit in $O(1)$ machine words — then extracting $S^\tau$ takes $O(z \log n)$ time and $O(z)$ space.
Decompressing LZ77

background
motivation
setting
past solutions

step 1
$S^T$ definition
$S^T$ extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching
We extract substrings of length greater than $z\tau$ by breaking them into pieces of length at most $z\tau$ and extracting the pieces \textit{consecutively}.

We extract substrings of length between $\tau + 1$ and $z\tau$ by breaking them into pieces of length of at most $\tau$ and extracting the pieces \textit{simultaneously}.

We extract substrings of length at most $\tau$ by treating those substrings as part of $S^\tau$ and repeating its extraction.
Decompressing LZ77

background
motivation
setting
past solutions

step 1
$S^T$ definition
$S^T$ extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching

substring extraction
audio
Decompressing LZ77

background
motivation
setting
past solutions

step 1
$S^T$ definition
$S^T$ extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching

substring extraction
audio
Decompressing LZ77

background
motivation
setting
past solutions

step 1
\(S^T\) definition
\(S^T\) extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching

substring extraction
audio
Decompressing LZ77

background
motivation
setting
past solutions

step 1
$S^T$ definition
$S^T$ extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching

substring extraction
audio
Decompressing LZ77

background
motivation
setting
past solutions

step 1
$S^T$ definition
$S^T$ extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching

substring extraction
audio
Decompressing LZ77

background
motivation
setting
past solutions

step 1
$S^T$ definition
$S^T$ extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching

substring extraction
audio
Decompressing LZ77

background
motivation
setting
past solutions

step 1
$S^r$ definition
$S^r$ extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching

substring extraction
audio
Decompressing LZ77

**background**
- motivation
- setting
- past solutions

**step 1**
- $S^T$ definition
- $S^T$ extraction
- mergeable dictionary

**step 2**
- substring division
- substring extraction

**applications**
- decompression
- pattern matching

**substring extraction audio**
Extracting a substring of length $\ell$ takes

$$O \left( \left( \frac{\ell}{z^\tau} + 1 \right) \cdot z \log n \right)$$

time and

$$O \left( z^\tau \cdot \frac{\log \sigma}{\log n} \right)$$

space.

If $\sigma = O(1)$ and $\tau = \Theta(\log n)$ then these bounds are $O(\ell + z \log n)$ and $O(z)$. 

substring extraction
applications
The obvious application of our results is decompression LZ77-compressed texts in small space.

For example, if \( \sigma = O(1) \) then we can choose \( \tau = \log n \) so we decompress in

\[
O \left( \frac{n}{z^\tau} \cdot z \log n \right) = O(n)
\]

time using \( O(z) \) space.
pattern matching

If we are looking for approximate matches to a pattern of length $m$ allowing $k$ mismatches, then we need only check characters within distance $m + k$ of the nearest phrase boundary.

The best known algorithm uses

$$O(z \log(n/z) + z \min(mk, k^4 + m) + \text{occ})$$

time and

$$O(z \log(n/z) + m + \text{occ})$$

space.

With our results, we can keep the same time and reduce the space to $O(z \log \log(n/z) + m + \text{occ})$ in general or $O(z + m + \text{occ})$ when $\sigma = O(1)$. 
Decompressing LZ77

background
motivation
setting
past solutions

step 1
$S^T$ definition
$S^T$ extraction
mergeable dictionary

step 2
substring division
substring extraction

applications
decompression
pattern matching

questions?