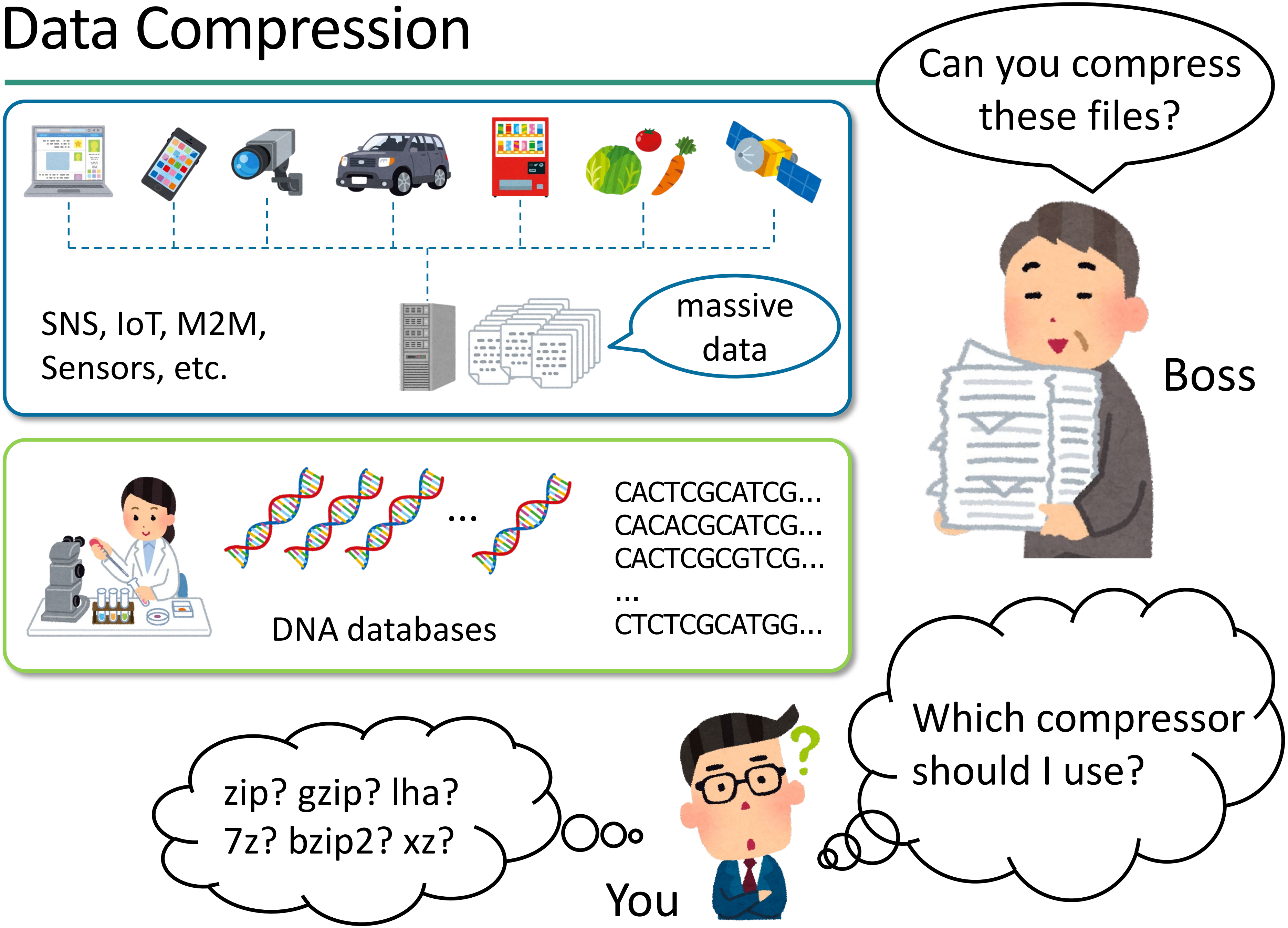
Sequences in London 2024

Sensitivity of string compressors and repetitiveness measures

Shunsuke Inenaga
(Kyushu University)

Data Compression



New Criterion for Compressors

Review 1 for Compressor C

T = ababaaba...ab

$C(T)$

(1) Excellent compression ratio!
(2) Super fast (de)compression!
(3) Searchable compression!

Review 2 for Compressor C

T' = a**X**abaaba..ab

$C(T')$

But the compressed size blew up just after a small edit operation!!

The compressor C looks great because it satisfies all the known criteria (1),(2),(3)!

What!? Can it happen? My company deals with dynamic data..., so it can be problematic.

New criterion **(4) Compression Sensitivity**

Compression Sensitivity

[Definition]

Sensitivity of compressor C

T' is any string that can be obtained by performing a single character edit operation on T.

Worst-case multiplicative sensitivity

$$\max\{|C(T')|/|C(T)| : T \in \Sigma^n, \mathrm{EditDistance}(T, T') = 1\}$$

Worst-case additive sensitivity

$$\max\{|C(T')| - |C(T)| : T \in \Sigma^n, \mathrm{EditDistance}(T, T') = 1\}$$

[Intuition]

Small sensitivity $\Longleftrightarrow$ Robust compression under edits/errors

Multiplicative Compression Sensitivity

Compressor / Measure	Upper Bound	Lower Bound
LZ77 z		
LZSS s		
Bidirectional scheme b		
Smallest grammar g^*		
BISECTION g_{BISCTN}		
GCIS g_{GCIS}		
LZ78 g_{78}		$\Omega(n^{1/4})$ [Lagarde & Perifel SODA 2018]
RLBWT r		$\Omega(\log n)$ [SOFSEM 2021]
Substring complexity δ		

Very few previous work

Lower Bound for RLBWT r

r: RLE size of the rightmost column of the lexicographically sorted rotations of the input string.

Reversed Fibonacci word s_i^{rev}

	s_6^{rev} = baabaababaabb	BWT
2	aabaababaabab	b
10	aababaabaabab	b
5	aababaababaab	b
13	abaabaababaab	b
8	abaababaabaab	b
3	abaababaababa	a
11	ababaabaababa	a
6	ababaababaaba	a
1	baabaababaaba	a
9	baababaabaaba	a
4	baababaababaa	a
12	babaabaababaa	a
7	babaababaabaa	a

	$(s_6\mathrm{b})^{\mathrm{rev}}$ = bbaabaababaabb	BWT
3	aabaababaababb	b
6	aababaababbaab	b
11	aababbaabaabab	b
4	abaababaababba	a
9	abaababbaabaab	b
7	ababaababbaaba	a
12	ababbaabaababa	a
14	abbaabaababaab	b
2	baabaababaabab	b
5	baababaababbaa	a
10	baababbaabaaba	a
8	babaababbaabaa	a
13	babbaabaababaa	a
1	bbaabaababaaba	a

$i = \log_\phi n$

$r(s_i^{\mathrm{rev}}) = 2$

$r(\mathrm{b}s_i^{\mathrm{rev}}) = \Omega(\log n)$

Multi. Sensitivity of Repetitiveness Measures & Dictionary Comp.

Compressor / Measure	Edit Operation	Upper Bound	Lower Bound
Substring complexity δ	del	1.5	1.5
	ins/sub	2	2
Smallest string attractor γ	any	$O(\log n)$	2.5 [Bannai et al. ESA 2022]
Bidirectional scheme b	ins/sub	2	2
	del	2	1.5
LZ77 z	any	2	2
LZSS s	ins	2	2
	del/sub	3	3
LZ78 g_{78}	ins	$O((n / \log n)^{2/3})$	$\Omega(n^{1/4})$ [Lagarde & Perifel SODA 2018]
	del/sub		$\Omega(n^{1/4})$
LZEnd z_{End}	any	$O(\log^2 (n / \mathrm{d}))$	2
RLBWT r	any	$O(\log r \log n)$	$\Omega(\log n)$

Multiplicative Sensitivity of Grammar Compressors

Grammar compressors	Edit Operation	Upper Bound	Lower Bound
Smallest grammar g^*	any	2	-
α-balanced	any	$O(\log(n / g^*))$	-
AVL-grammar			
RePair	any	$O((n / \log n)^{2/3})$	-
Longest-Match			
Greedy			
Bisection	sub	2	2
	ins/del	$\lvert\Sigma\rvert+1$	$\lvert\Sigma\rvert$
GCIS	any	4	4
CDAWG	sub	8	2
	ins/del	13	2

Multi. Sensitivity of Repetitiveness Measures & Dictionary Comp.

Compressor / Measure	Edit Operation	Upper Bound	Lowe Bound
Substring complexity δ	del	1.5	1.5
	ins/sub	2	2
Smallest string attractor γ	any	$O(\log n)$	2
Bidirectional scheme b	ins/sub		[Bannai et al. A 2022]
	del		
LZ77 z	any		2
LZSS s	ins		2
	del/sub	3	3
LZ78 g_{78}	ins	$O((n / \log n)^{2/3})$	$\Omega(n^{1/4})$ [Lagarde & Perifel SODA 2018]
	del/sub		$\Omega(n^{1/4})$
LZEnd z_{End}	any	$O(\log^2 (n / \mathrm{d}))$	2
RLBWT r	any	$O(\log r \log n)$	$\Omega(\log n)$

I will talk about this

LZSS (zip / gzip / lha / 7zip / png ...)

LZSS [Storer-Szymanski, 1982]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$T =$	a	b	a	a	b	a	b	a	b	a	b	a	b	a	b

Scans T from left to right and parses T

- at each fresh letter, or
- at the ending position of the longest previously occurring substring.

LZSS (zip / gzip / lha / 7zip / png ...)

LZSS [Storer-Szymanski, 1982]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$T =$	a |	b	a	a	b	a	b	a	b	a	b	a	b	a	b

Fresh letter

Scans T from left to right and parses T

- at each fresh letter, or
- at the ending position of the longest previously occurring substring.

LZSS (zip / gzip / lha / 7zip / png ...)

LZSS [Storer-Szymanski, 1982]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$T =$	a |	b |	a	a	b	a	b	a	b	a	b	a	b	a	b

Fresh letter

Scans T from left to right and parses T

- at each fresh letter, or
- at the ending position of the longest previously occurring substring.

LZSS (zip / gzip / lha / 7zip / png ...)

LZSS [Storer-Szymanski, 1982]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$T =$	a	b	a	a	b	a	b	a	b	a	b	a	b	a	b

The ending position of the longest previously occurring substring.

Scans T from left to right and parses T

- at each fresh character, or
- at the ending position of the longest previously occurring substring.

LZSS (zip / gzip / lha / 7zip / png ...)

LZSS [Storer-Szymanski, 1982]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$T =$	a	b	a	a	b	a	b	a	b	a	b	a	b	a	b

The ending position of the longest previously occurring substring.

LZSS (zip / gzip / lha / 7zip / png ...)

LZSS [Storer-Szymanski, 1982]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$T =$	a	b	a	a	b	a	b	a	b	a	b	a	b	a	b

The ending position of the longest previously occurring substring.

LZSS (zip / gzip / lha / 7zip / png ...)

LZSS [Storer-Szymanski, 1982]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$T =$	a	b	a	a	b	a	b	a	b	a	b	a	b	a	b

The ending position of the longest previously occurring substring.

LZSS (zip / gzip / lha / 7zip / png ...)

LZSS [Storer-Szymanski, 1982]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$T =$	a|	b|	a|	a	b	a|	b	a|	b	a	b	a|	b	a	b|

The ending position of the longest previously occurring substring.

LZSS (zip / gzip / lha / 7zip / png ...)

LZSS [Storer-Szymanski, 1982]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$T =$	a |	b |	a |	a	b	a |	b	a |	b	a	b	a |	b	a	b |

(5, 7)

LZSS (zip / gzip / lha / 7zip / png ...)

LZSS [Storer-Szymanski, 1982]

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

$$T = \mathrm{a} \mid \mathrm{b} \mid \mathrm{a} \mid \mathrm{a}\ \mathrm{b}\ \mathrm{a} \mid \mathrm{b}\ \mathrm{a} \mid \mathrm{b}\ \mathrm{a}\ \mathrm{b}\ \mathrm{a} \mid \mathrm{b}\ \mathrm{a}\ \mathrm{b} \mid$$

(a) (b) (1, 1) (1, 3) (2, 3) (5, 8) (5, 7)

Compression size (# of phrases) $= 7$

LZSS (zip / gzip / lha / 7zip / png ...)

LZSS [Storer-Szymanski, 1982]

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$T =$	a|	b|	a|	a	b	a|	b	a|	b	a	b	a|	b	a	b|

(a) (b) (1, 1) (1, 3) (2, 3) (5, 8) (5, 7)

Compression size (# of phrases) $= 7$

LZSS with self-references

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$T =$	a|	b|	a|	a	b	a|	b	a	b	a	b	a	b	a	b|

(a) (b) (1, 1) (1, 3) (5, 13)

Compression size (# of phrases) $= 5$

Upper Bound for Sensitivity of LZSS

s : LZSS compression size before edit
s': LZSS compression size after edit

Theorem

For any string, we have the following:

Multiplicative sensitivity for LZSS is $s'/s \leq 3.$

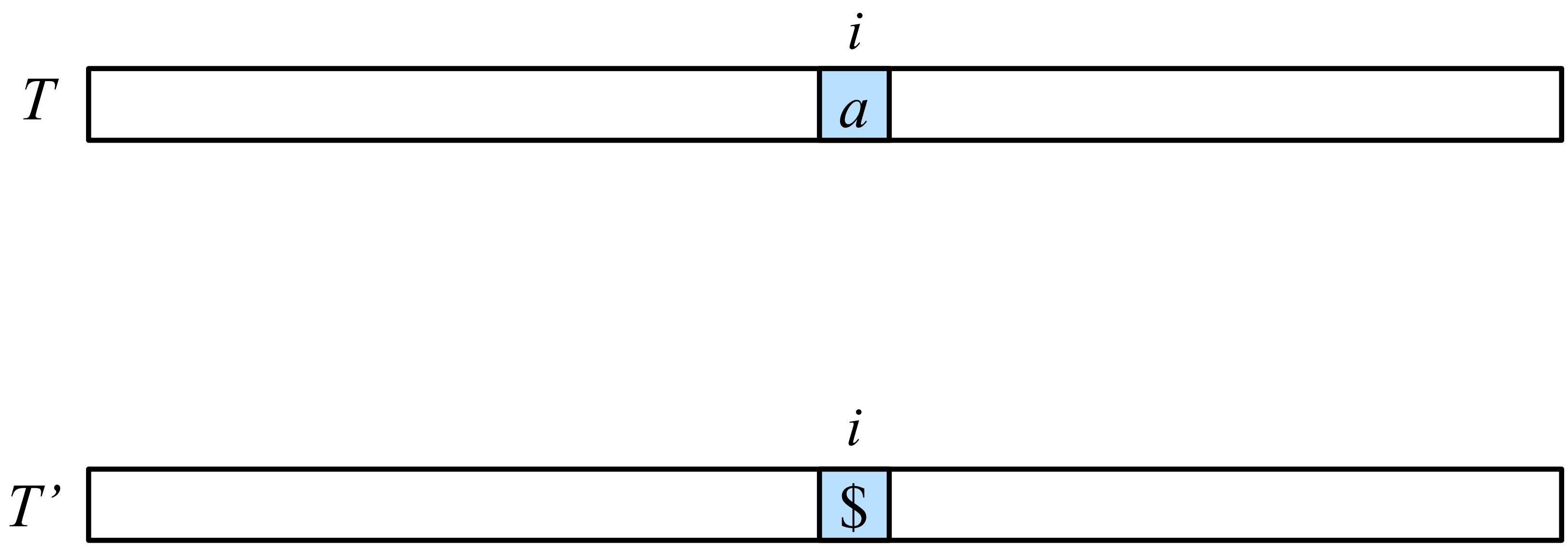
Additive sensitivity for LZSS is $s' - s \leq 2s - 2.$

These upper bounds hold for both versions of LZSS with/without self-references.

Upper Bound for Sensitivity of LZSS

Proof for $s' \leq 3s$ (without self-references)

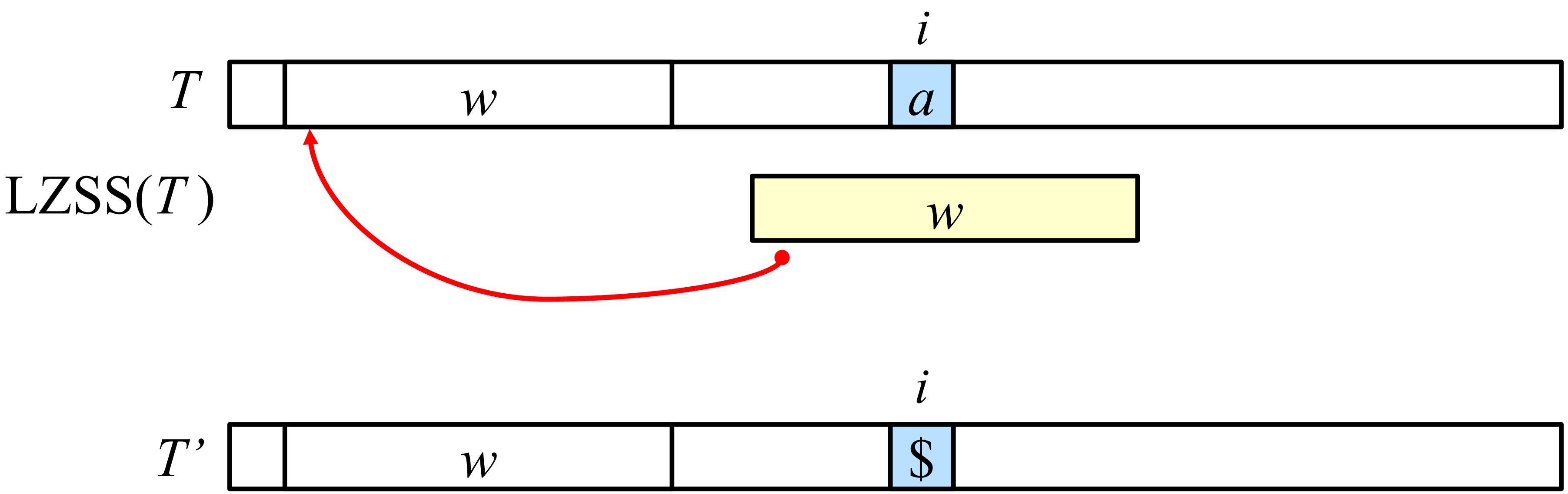
Claim 1: For the phrase w that contains the edited position i, # phrases starting in the interval for w can increase to at most 3.



Upper Bound for Sensitivity of LZSS

Proof for $s' \leq 3s$ (without self-references)

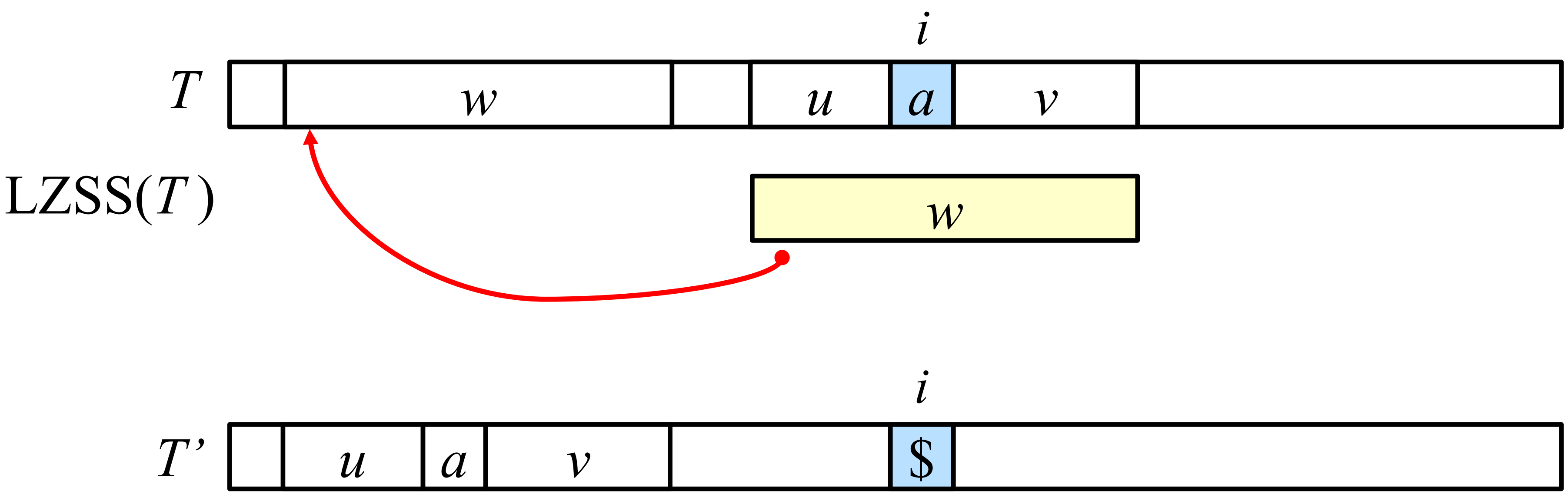
Claim 1: For the phrase w that contains the edited position i, # phrases starting in the interval for w can increase to at most 3.



Upper Bound for Sensitivity of LZSS

Proof for $s' \leq 3s$ (without self-references)

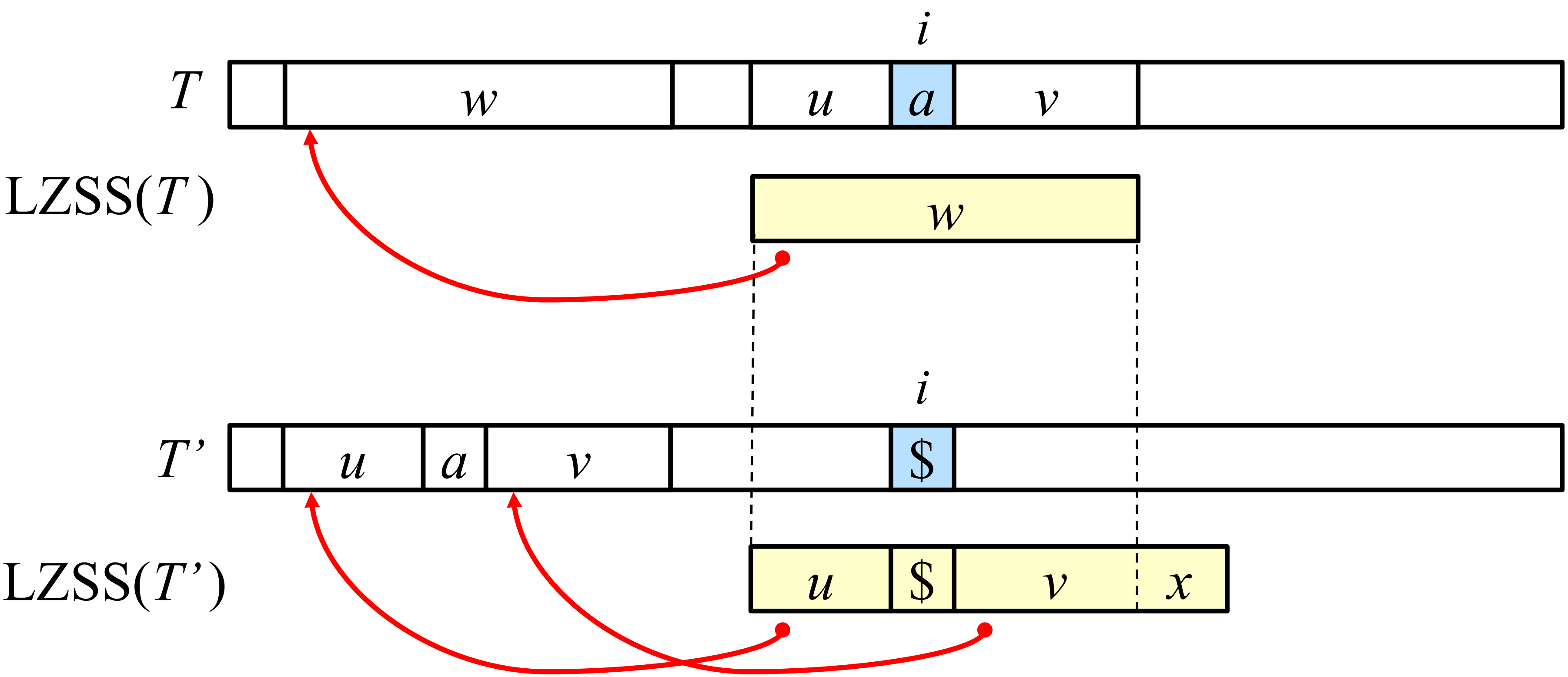
Claim 1: For the phrase w that contains the edited position i, # phrases starting in the interval for w can increase to at most 3.



Upper Bound for Sensitivity of LZSS

Proof for $s' \leq 3s$ (without self-references)

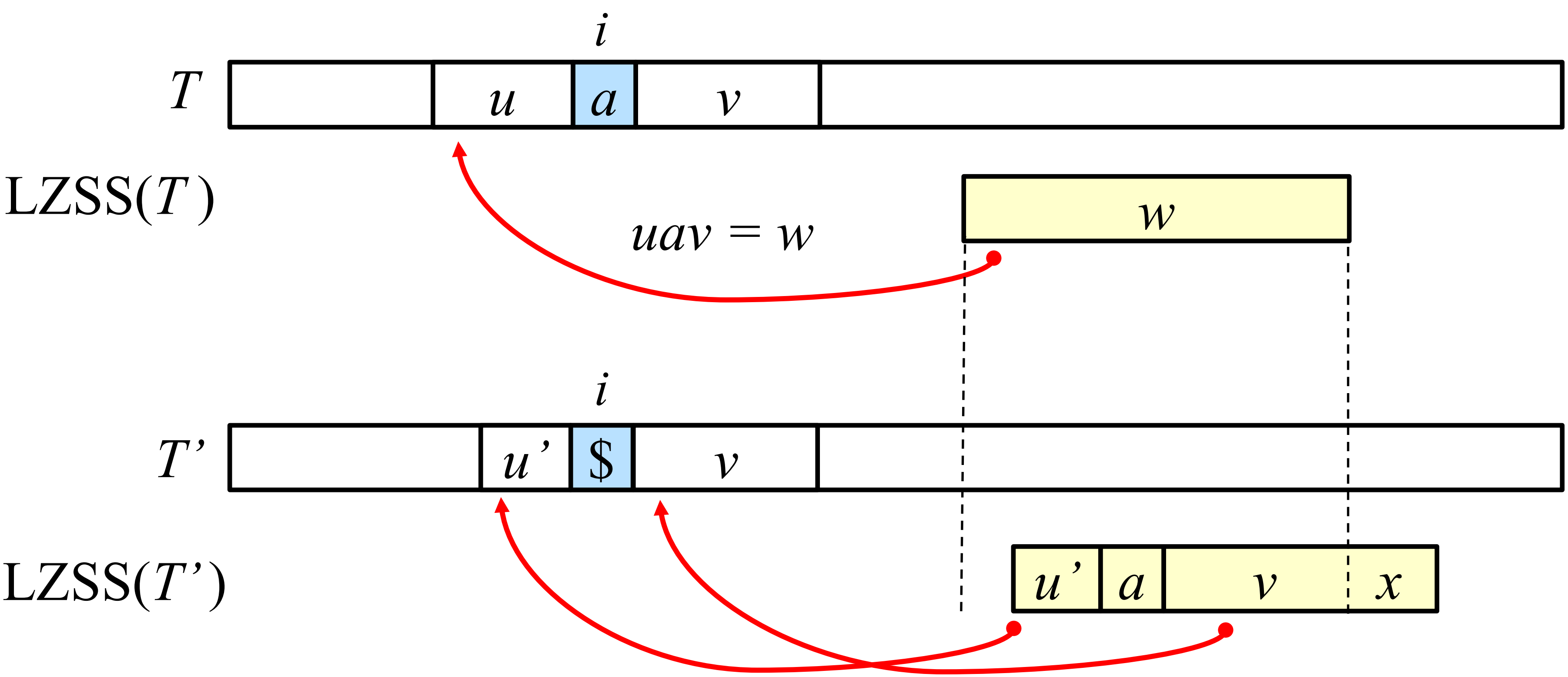
Claim 1: For the phrase w that contains the edited position i, # phrases starting in the interval for w can increase to at most 3.



Upper Bound for Sensitivity of LZSS

Proof for $s' \leq 3s$ (without self-references)

Claim 2: For any phrase w to the right of the edited position i, # phrases starting in the interval for w can increase to at most 3.



Upper/Lower Bounds for Sensitivity of LZSS

Theorem (Upper Bound)

For any string, we have the following:

Multiplicative sensitivity for LZSS is $s'/s \leq 3$.

Additive sensitivity for LZSS is $s' - s \leq 2s - 2$.

Theorem (Lower Bound)

There exists a family of strings of length n such that:

Multiplicative sensitivity for LZSS is $\lim_{s \to \infty} \inf(s'/s) = 3$.

Additive sensitivity for LZSS is $s' - s = 2s - \Theta(\sqrt{s}) = \Omega(\sqrt{n})$.

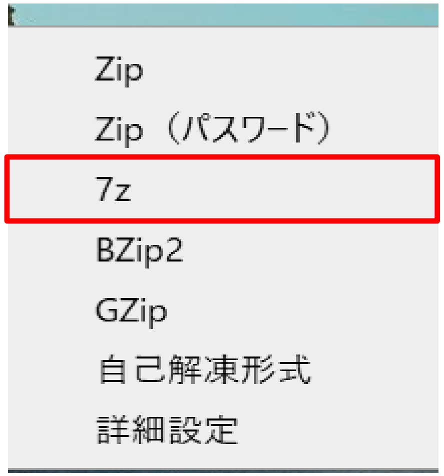
For the lower bound, we showed strings that have the aforementioned worst-case situations for most of their phrases.

Sensitivity of Compression Software

Does similar happen for
a real-world compression software?

We tested compression software

- 7zip = LZSS + RangeCoder.
- Supports **.7z** compression format.
- Compression ratio of 7zip is usually better than that of zip by 30-70%.



Sensitivity of Compression Software (7zip)

stringX： length 1,172,887 (1.12 MB)

!“#$%&‘()*+,./0123456789:;⇔?@ABCDEFGHIJKLMN!”#$%&’()*+,./01234
56789:;⇔?@ABCDEFGHIJKLM!“#$%&‘()*+,./0123456789:;<=>?@ABCDEF
GHIJKL!”#$%&’()*+,./0123456789:;<=>?@ ABCDE … ’!“#$%&!
”#$%!“#$!”#!“!!**{**OOPOPQOPQROPQRSOPQRSTOPQRSTUOPQRST …

Replace the 1083rd letter “ **{** “ with “ **~** ”

stringXp

!“#$%&‘()*+,./0123456789:;⇔?@ABCDEFGHIJKLMN!”#$%&’()*+,./01234
56789:;⇔?@ABCDEFGHIJKLM!“#$%&‘()*+,./0123456789:;<=>?@ABCDEF
GHIJKL!”#$%&’()*+,./0123456789:;<=>?@ ABCDE … ’!“#$%&!
”#$%!“#$!”#!“!!**~**OOPOPQOPQROPQRSOPQRSTOPQRSTUOPQRST …

7zip compressed size	stringX.7z	4,884 byte
	stgingXp.7z	7,189 byte

1.5 times bigger!

References

1. Sensitivity of string compressors and repetitiveness measures.
   T. Akagi, M. Funakoshi, S. Inenaga
   (Information & Computation 2023)
2. Novel results on the number of runs of the Burrows-Wheeler-transform.
   S. Giuliani,. S. Inenaga, Z. Lipták, N. Prezza, M. Sciortino, A. Toffanello
   (SOFSEM 2021)
3. Bit Catastrophes for the Burrows-Wheeler Transform.
   S. Giuliani, S. Inenaga, Z. Lipták, G. Romana, M. Sciortino, C. Urbina
   (DLT 2023)
4. On Sensitivity of Compact Directed Acyclic Word Graphs.
   H. Fujimaru, Y. Nakashima, S. Inenaga
   (WORDS 2023)
5. Sensitivity of CDAWGs is constant.
   R. Hamai, S. Inenaga
   (Unpublished)