Compression Sensitivity of the Bijective Burrows-Wheeler transform

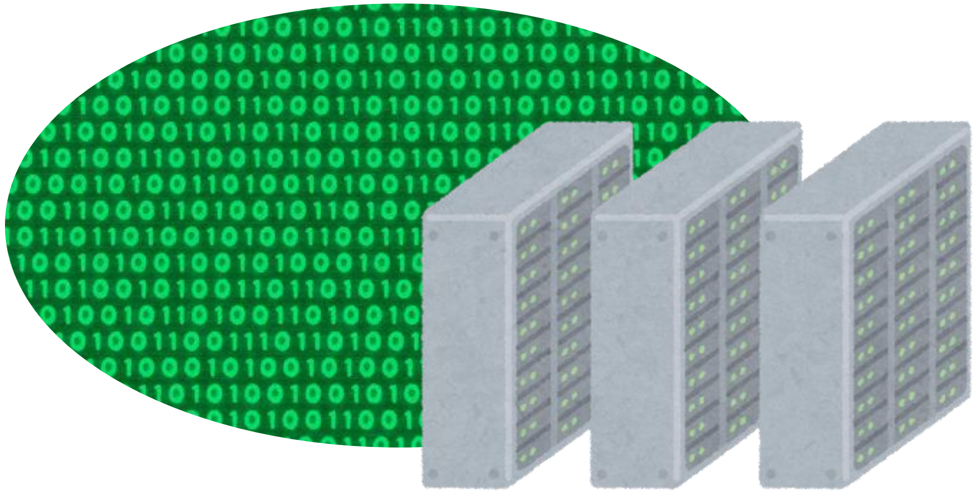
Hyodam Jeon、Dominik Köppl

Published in:
Mathematics 2025, 13, 1070

University of Yamanashi
Computer science and engineering

International Workshop on Discrete Mathematics and Algorithms 2025

Background



Setting: Need to store large text collections in compressed form

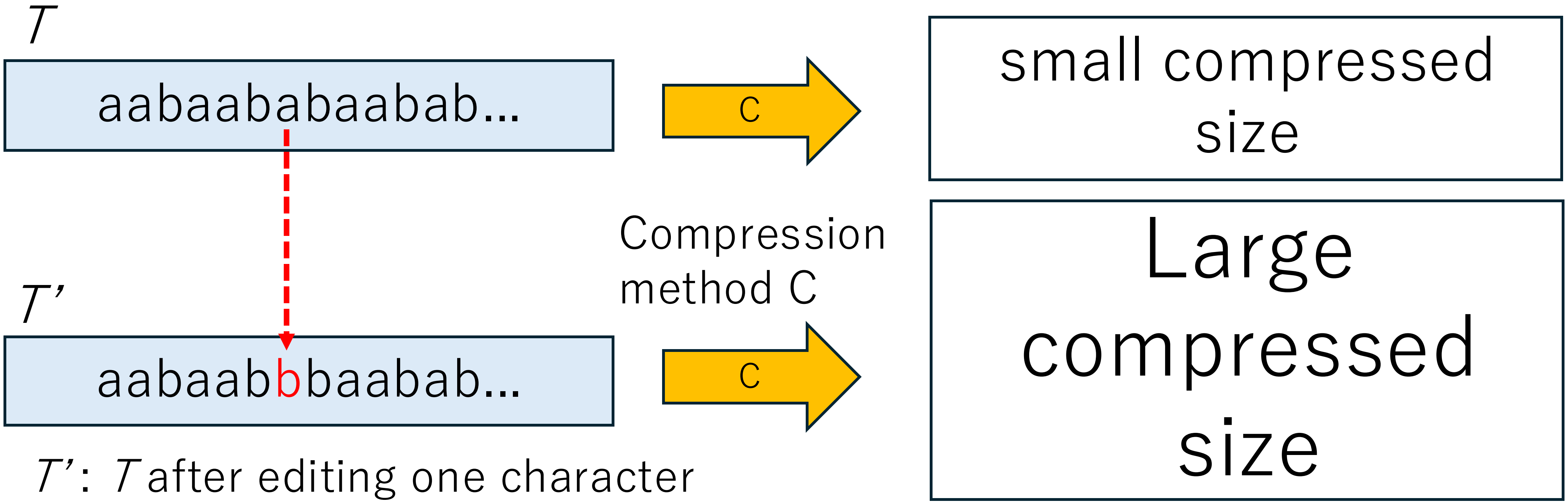
Examples:

- Biological data (ncbi),
- Source code (github),
- Websites (wayback machine)

- Expect that similar data can be stored compressed with similar sizes
- However: small changes of the input can cause large difference in compressed size!
- Difference can cause economic loss!
- Question: how bad can it get?

Research objective

How much impact has a single character edit of the input?



T': T after editing one character
Edit: insert/deletion/exchange
Here: exchange a with b

compression sensitivity = What is the max. difference between the compressed sizes of the edited text **C**(T') and the original **C**(T)?

Related work

Compression sensitivity has been studied for various compressors

Compression method	**Related work**
Lempel-Ziv 78 (LZ78) (gif image compression)	Lagarde&Perifel '18
Lempel-Ziv 77 (gzip/zip/etc.)	Akagi+ '23
BWT (bzip2, compressed indexes)	Giuliani+ '23
lex-parse	Nakashima+ '24
string attractor, bidirectional macro scheme	Fujie+ '24

however: the sensitivity of the **bijective BWT (BBWT)**
has not yet been studied

Clustering effect of the BBWT

T, r(T) = 110

aabaababaabaababaababaabaababaab
aababaababaabaababaababaabaababa
abaababaababaabaababaabaababaaba
baabaababaababaabaababaabaababaa
babaabaababaabab

BBWT

BBWT(*T*) , r(BBWT(T)) = **2**

bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaa

- BBWT arranges characters by previous context such that characters in the same context are grouped together
- BBWT(T) is likely to be better run-length compressible than the input
- **Repetitiveness measure r(*T*)** : number of maximal consecutive character occurrences in text T (size of the run-length encoding/compression)

Bijective BWT (BBWT) [Gil&Scott '12]

Lyndon words and factors

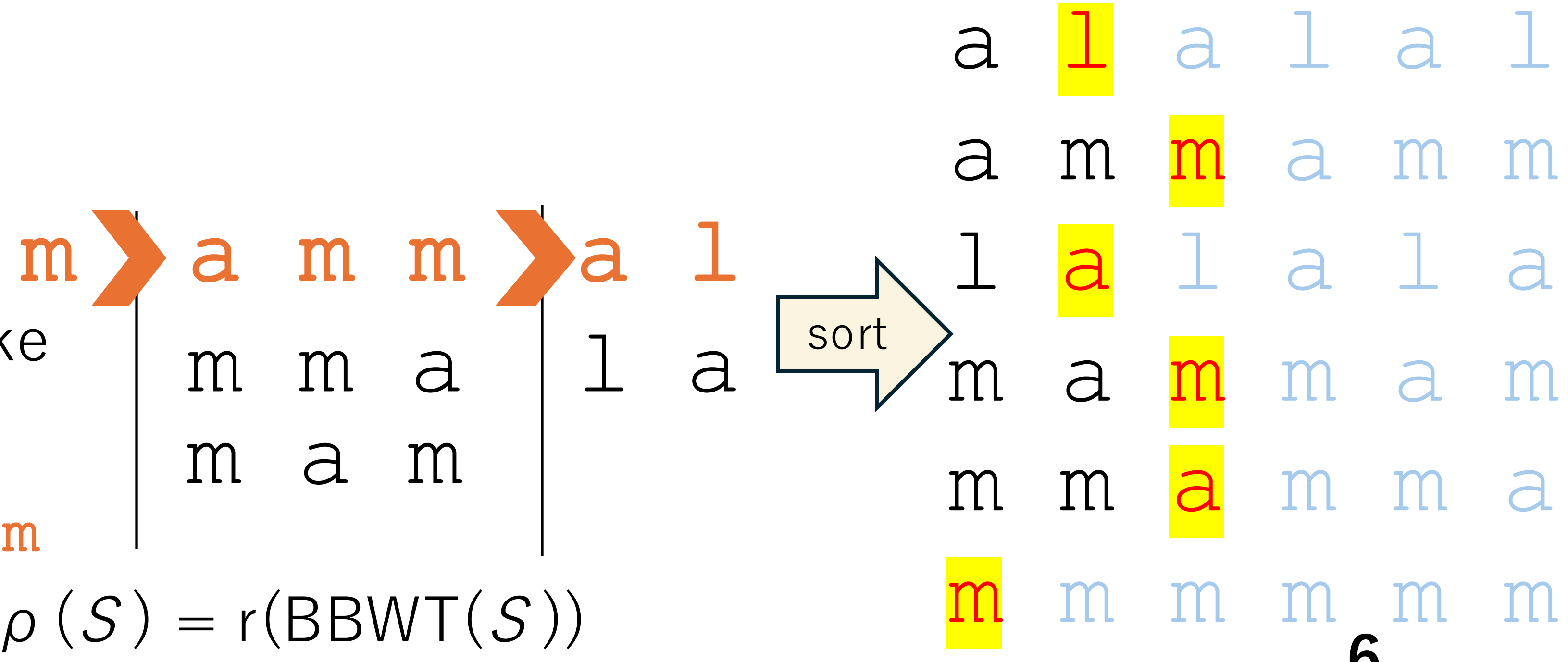
- A word S is **Lyndon** if it is smaller than all its conjugates
- If S is not Lyndon, we can factorize it uniquely into Lyndon factors L_1, L_2, L_3 … L_m such that the Lyndon factors are in lex. decreasing order $L_1 \geq L_2 \geq \cdots \geq L_m$

Order: a < l < m

a	l	m	a	m	m
a	m	m	a	l	m
l	m	a	m	m	a
m	a	l	m	a	m
m	a	m	m	a	l
m	m	a	l	m	a

Bijective BWT

- sort the conjugates of all Lyndon factors of S and take the last character of each
  - BBWT(mammal)= lmamam
  - compression measure : $\rho(S) = \mathrm{r}(\mathrm{BBWT}(S))$

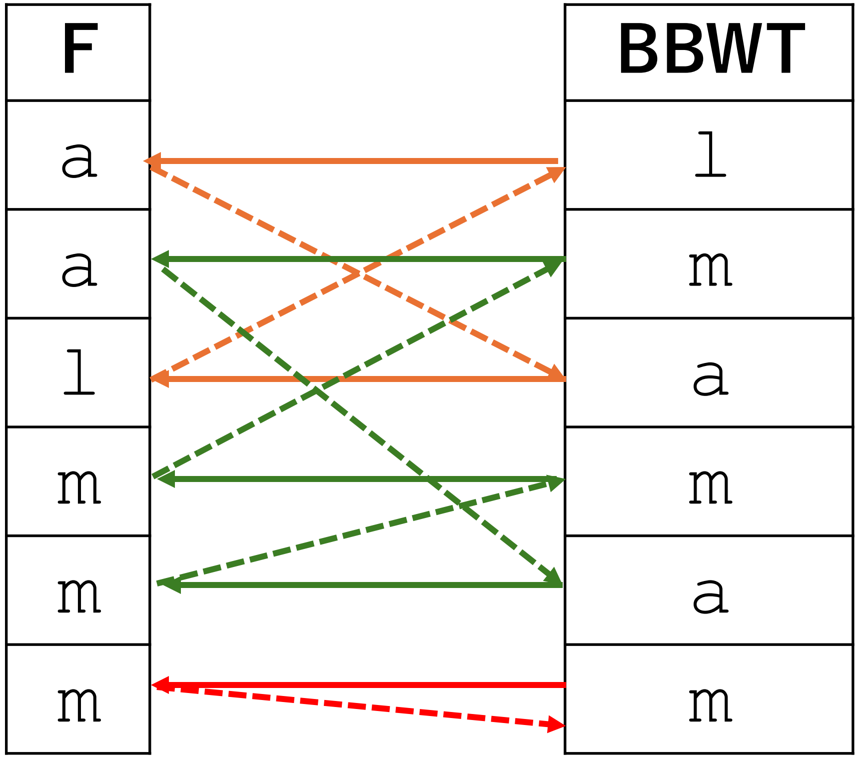


Inverting the BBWT

To obtain S from BBWT(S), we follow the cycles in the BBWT → obtain S's Lyndon factors

First sorted characters **F**						BBWT(mammal)					
a	l	a	l	a	l	a	l	a	l	a	l
a	m	m	a	m	m	a	m	m	a	m	m
l	a	l	a	l	a	l	a	l	a	l	a
m	a	m	m	a	m	m	a	m	m	a	m
m	m	a	m	m	a	m	m	a	m	m	a
m	m	m	m	m	m	m	m	m	m	m	m

Loop to retrieve Lyndon factors



m
a m m
a l

a l
a m m
m

Sort in descending order

Sensitivity : Results

Red: new results, (*x) : edit operation with character x
Black: known results due to Giuliani+ '23

method / Input text	Edit operation	BWT	BBWT
Fibonacci word ・r(BWT(F_{2k}))=2 ・ρ(Lyndon conjugate of F_{2k}) =2	remove last letter	**2*k***	**≥*k***
	change last letter	**2*k* +2(*a)**	**≥*k* + 1(*#)**
		2*k* +2(*#)	**≥*k* (*c)**
	insert at specific positions	-	**≥ *k*** **≥*k* +1**

Fibonacci

Fibonacci words

- $F_0 = \mathrm{b}$, $F_1 = \mathrm{a}$, $F_k = F_{k-1}F_{k-2}$
- $F_2 = \mathrm{ab}$
- $F_3 = \mathrm{aba}$
- $F_4 = \mathrm{abaab}$
- $F_5 = \mathrm{abaababa}$
- $F_6 = \mathrm{abaababaabaab}$
- $F_7 = \mathrm{abaababaabaababaababa}$

Fibonacci numbers

- $f_0 = 1$, $f_1 = 1$, $f_k = f_{k-1} + f_{k-2}$
- $f_2 = 2$
- $f_3 = 3$
- $f_4 = 5$
- $f_5 = 8$
- $f_6 = 13$
- $f_7 = 21$

Fibonacci

Fibonacci words

- $F_0 = \mathrm{b}$, $F_1 = \mathrm{a}$, $F_k = F_{k-1}F_{k-2}$
- $F_2 = \mathrm{ab}$
- $F_3 = \boxed{\mathrm{a}}\mathrm{ba}$
- $F_4 = \boxed{\mathrm{aba}}\mathrm{ab}$
- $F_5 = \boxed{\mathrm{abaaba}}\mathrm{ba}$
- $F_6 = \boxed{\mathrm{abaababaaba}}\mathrm{ab}$
- $F_7 = \boxed{\mathrm{abaababaabaababaaba}}\mathrm{ba}$

X_k: palindrome

$$F_k = X_k\mathrm{ab} \text{ if } k \text{ is even}$$
$$= X_k\mathrm{ba} \text{ if } k \text{ is odd}$$

Fibonacci numbers

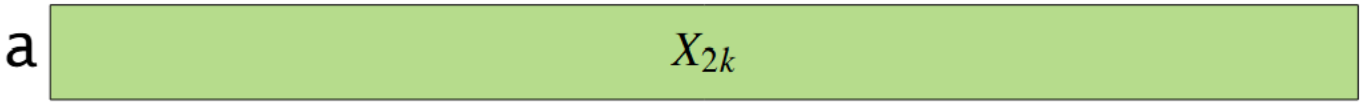
- $f_0 = 1$, $f_1 = 1$, $f_k = f_{k-1} + f_{k-2}$
- $f_2 = 2$
- $f_3 = 3$
- $f_4 = 5$
- $f_5 = 8$
- $f_6 = 13$
- $f_7 = 21$

Logarithmic increase of ρ

What we did: Remove last character of Lyndon conjugate aX_{2k}b of F_{2k} .

Theorem： ρ (aX_{2k}b) $= 2$ but ρ (aX_{2k}) $\geq k$

Proof Idea: Number of distinct Lyndon factors is a lower bound of ρ



$= \mathrm{a}X_{2k}$

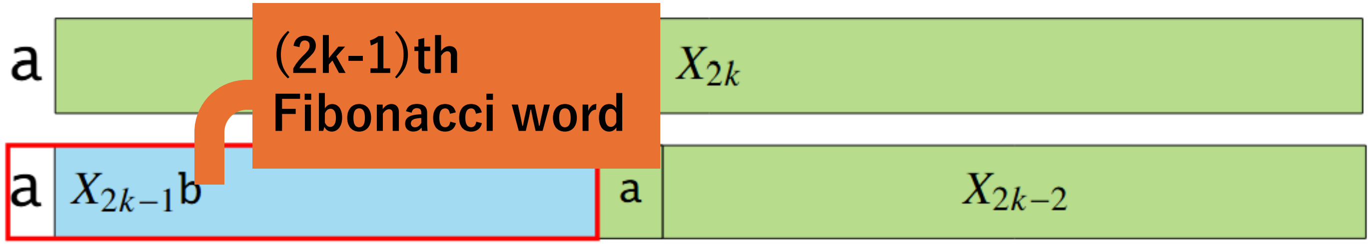
Green parts are not Lyndon, blue parts are Lyndon factors

Logarithmic increase of ρ

What we did: Remove last character of Lyndon conjugate aX_{2k}b of F_{2k} .

Theorem：$\boldsymbol{\rho\,(\mathrm{a}X_{2k}\mathrm{b}) = 2}$ **but** $\boldsymbol{\rho\,(\mathrm{a}X_{2k}) \geq k}$

Proof Idea: Number of distinct Lyndon factors is a lower bound of ρ



$= \mathrm{a}X_{2k}$

$= \boxed{\mathrm{a}\ X_{2k-1}\ \mathrm{b}}\mathrm{a}\ X_{2k-2}$

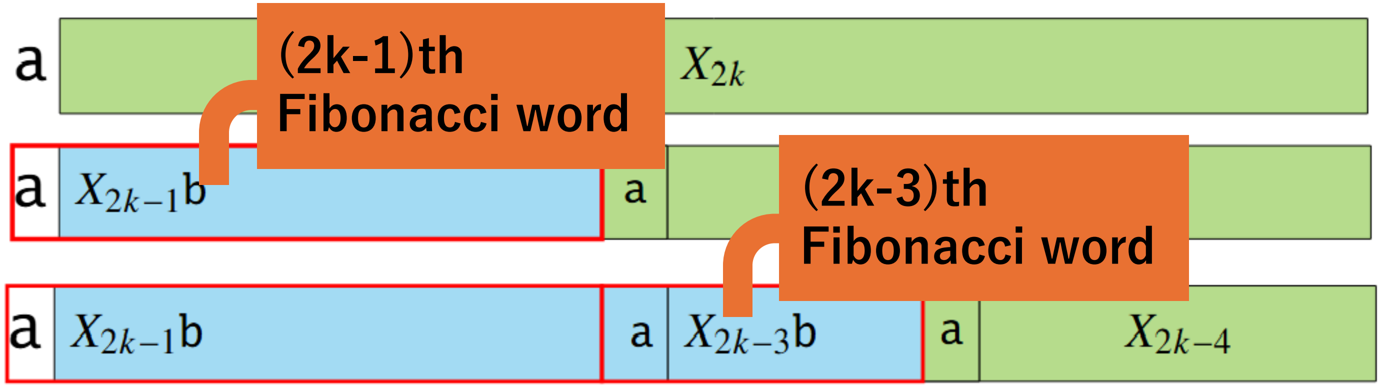
Green parts are not Lyndon, blue parts are Lyndon factors

Logarithmic increase of ρ

What we did: Remove last character of Lyndon conjugate aX_{2k}b of F_{2k} .

Theorem： ρ (aX_{2k}b) = 2 but ρ (aX_{2k}) ≥ k

Proof Idea: Number of distinct Lyndon factors is a lower bound of ρ



$= \text{a}X_{2k}$

$= \boxed{\text{a } X_{2k-1} \text{ b}}\text{a } X_{2k-2}$

$= \boxed{\text{a } X_{2k-1} \text{ b}}\boxed{\text{a } X_{2k-3} \text{ b}}\text{a } X_{2k-4}$

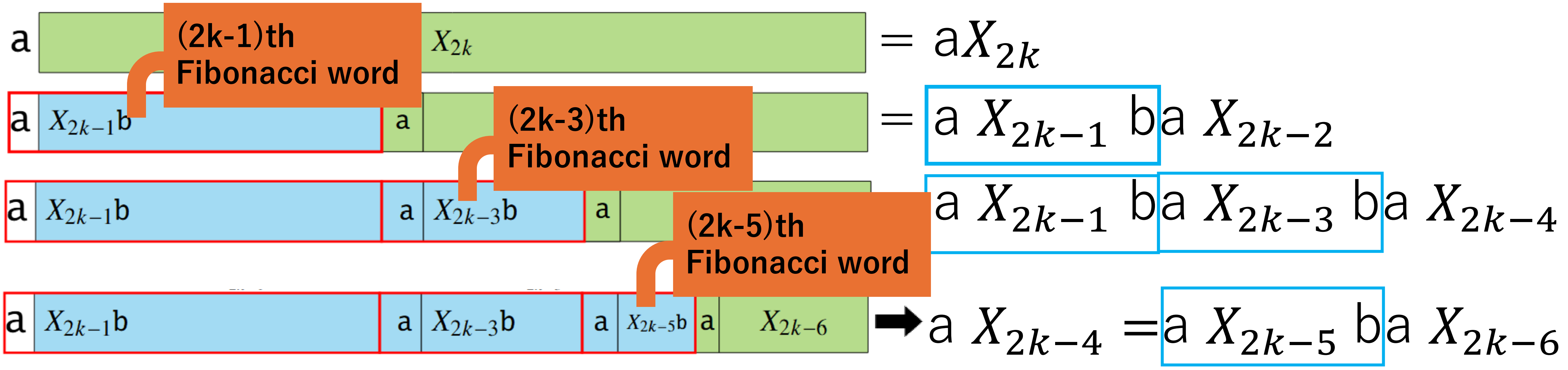
Green parts are not Lyndon, blue parts are Lyndon factors

Logarithmic increase of ρ

What we did: Remove last character of Lyndon conjugate aX_{2k}b of F_{2k} .

Theorem： ρ (aX_{2k}b) = 2 but ρ (aX_{2k}) $\geq k$

Proof Idea: Number of distinct Lyndon factors is a lower bound of ρ



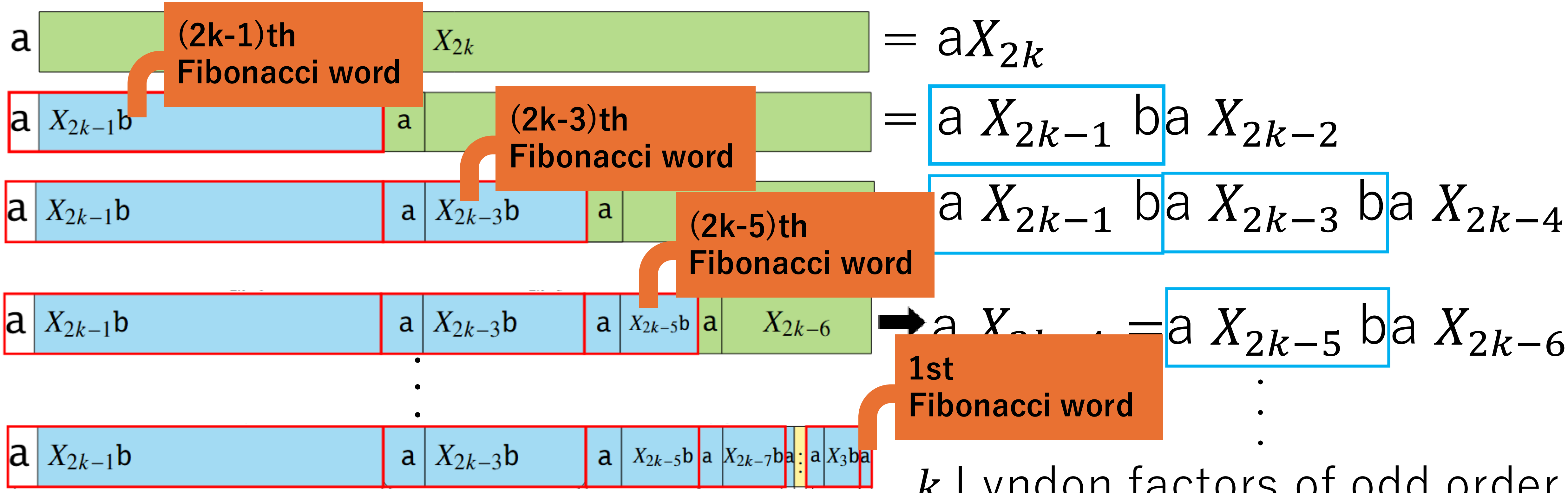
Green parts are not Lyndon, blue parts are Lyndon factors

Logarithmic increase of ρ

What we did: Remove last character of Lyndon conjugate aX_{2k}b of F_{2k} .

Theorem：ρ(aX_{2k}b) = 2 but ρ(aX_{2k}) ≥ k

Proof Idea: Number of distinct Lyndon factors is a lower bound of ρ



k Lyndon factors of odd order

Green parts are not Lyndon, blue parts are Lyndon factors

Recap and future work

recap

1. compression measure ρ can increase by a factor of $\Theta(\log n)$
2. not shown but proved: size can increase by $+\Theta(\sqrt{n})$

➡ a single edit can change the compressed size dramatically

➡ **BWT and BBWT are not well-designed measures for compressibility**

future work

・experiments suggest that $\rho(S)=2k$ in the proof of the previous slides, but we could not prove it

・improve the bounds: is there a non-trivial upper bound for ρ?

・determine the sensitivity of other compression methods

W_k文字列

- $W_k = \prod_{i=2}^{k-1}(P_i E_i) Q_k, \left(P_i = ab^i aa,\ E_i = ab^i aba^{i-2}, Q_k = ab^k a\right)$
- $\overline{W_k} = \prod_{i=2}^{k-1}(\overline{P_i}\,\overline{E_i}) \overline{Q_k}, \left(\overline{P_i} = ba^i bb, \overline{E_i} = ba^i bab^{i-2}, \overline{Q_k} = ba^k b\right)$
- $|W_k| = (3k^2 + 7k - 18)/\,2 = \Theta(k^2)$

a bを反転

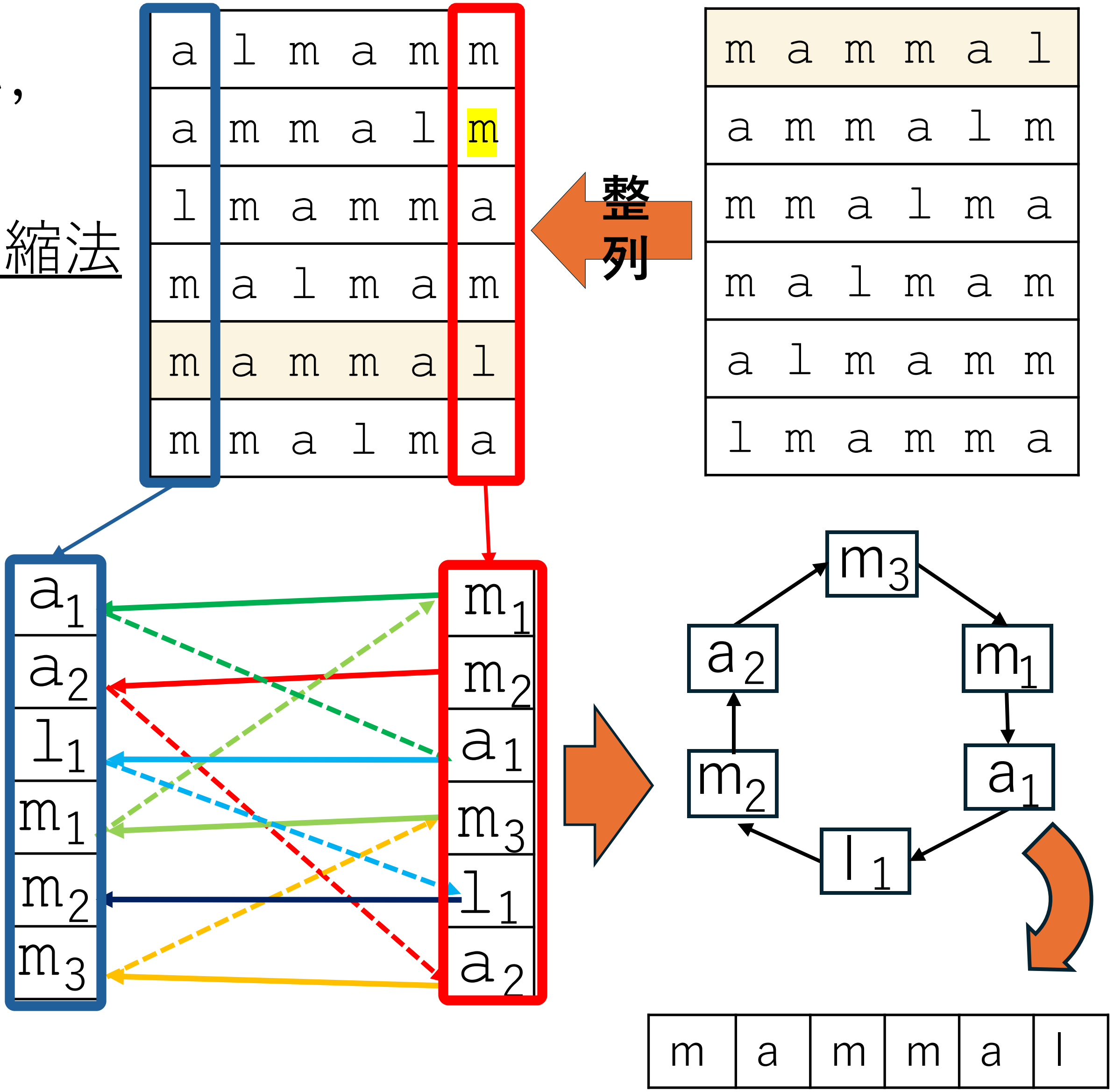
k	W_k
3	abbaaabbababbba
4	abbaaabbababbbaaabbbabaabbbba
5	abbaaabbababbbaaabbbabaabbbbaaabbbbabaaabbbbba

k	$\overline{W_k}$
3	baabbbaababaaab
4	baabbbaababaaabbbaaababbaaaab
5	baabbbaababaaabbbaaababbaaaabbbaaaababbbaaaaab

Burrows-Wheeler 変換(BWT) [Burrows&Wheeler 1994]

- 対象文字列のすべての巡回文字列を生成し，
  それらをアルファベット順序でソートし、
各行の最後の文字を連結して得られる<u>可逆圧縮法</u>
  - BWT(mammal)= mmamla

- 逆変換は，<u>文字の安定ソート</u>が
  逆巡回の<u>最初の文字に対応</u>する
  ことを利用する (LF-mapping)
  - 正確には「環状文字列 ➔ 文字列」の可逆変換。
  ➔ 復元には開始位置が必要



W_kの全反射BWTー最後の文字を「＃」に置換する場合

$C_k^b = W_k$のLyndon語の最後の文字を削除した文字列

$C_k^b\# = C_k^b$の最後に#を追加した文字列 = W_kの最後の文字を#に置換した文字列

1. **文字列をLyndon分解する** $C_k^b\# = \underbrace{\boxed{\mathrm{a}^{k-2}\mathrm{b}^k\mathrm{a}\cdot\left(\prod_{i=2}^{k-2}\mathrm{ab}^i\mathrm{aaab}^i\mathrm{aba}^{i-2}\right)\cdot\mathrm{ab}^{k-1}\mathrm{aaab}^{k-1}}}_{\rho(D_K)=8k-18}\boxed{\mathrm{a}}\boxed{\#}$

2. **各Lyndon要素の圧縮サイズを求める**

①$D_k = C_k^b$からさらに最後の文字を削除した文字列

D_kはLyndon語なので、既存研究を応用し、BWTと同じ方法で圧縮サイズを求めると、

8k-18になる.

② aの圧縮サイズは1

③ #の圧縮サイズ1

3. **Lyndon要素を辞書順で並べ替えて、足し合わせる**

$\rho\,(C_k^b) =$ #の圧縮サイズ**1** ＋ aの圧縮サイズ**1** + D_kの圧縮サイズ**8k-18**

= 8k-16

例　質問された時

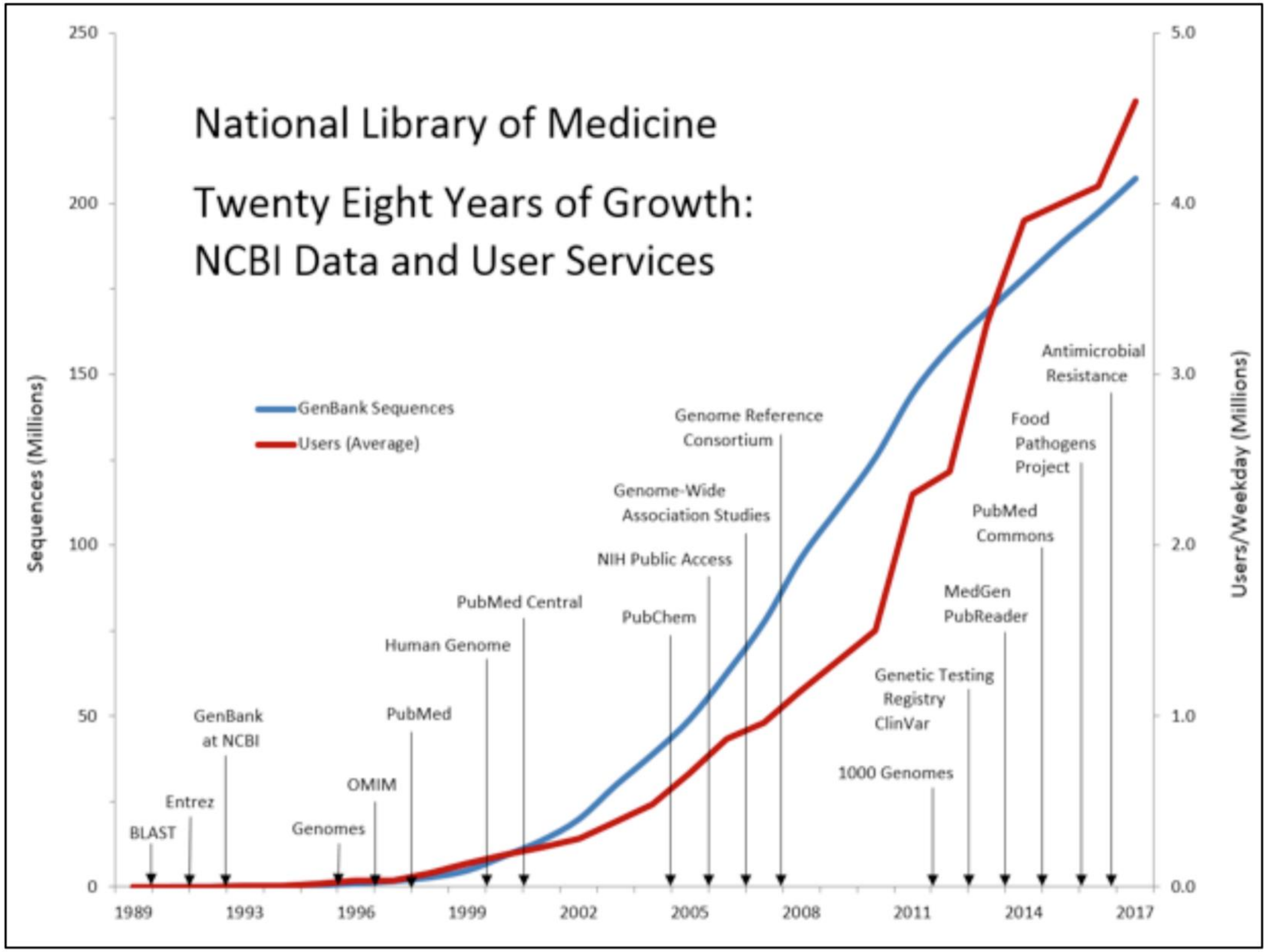
Q.**何故圧縮をしようとしたか？？**
A. 今のテーマは、私たちにとって身近なものだと感じました。普段よく使う文字列が変形されたとき、どこまで最悪のケースに至るのかを数学的に証明できるのは、とても面白いと思いました。

Q. **Fibonacciとwkは良く使われる文字列なの？**
Fibonacci文字列とW_k文字列はよく使われてはいませんが、
規則性があるので追跡しやすい性質があり、下限(少なくてもどれくらい悪くなるか）が特定しやすい。

Q. **圧縮効率が悪い文字列はどうなるの？**
容量が多くかかるので、保存しにくい

圧縮研究

データの量



年度

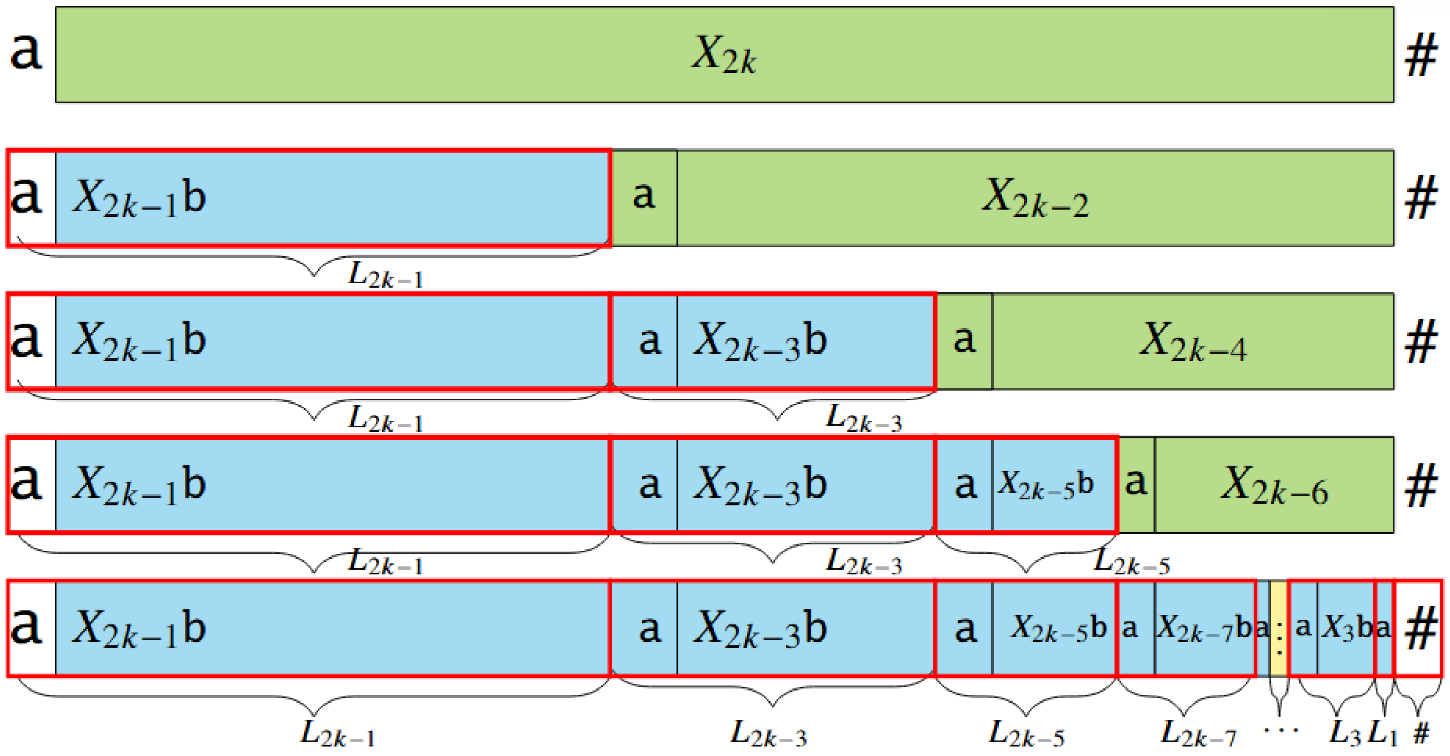
BBWTのrが増加するFibonacci文字列の編集(2)

定理:

- 2kが十分大きいとき， L_{2k}の最後の文字を#に変更した文字列vのBBWTのrの下限はk

$r(L_{2k} = \mathrm{a}X_{2k}\mathrm{b}) = 2$ ➔ $r(v) \geq k + 1$

証明



BBWTのrが増加するFibonacci文字列の編集(3)

定理:

- 2kが十分大きいとき， L_{2k}の最後の文字をcに変更した文字列v'のBBWTのrの下限はk

$r(L_{2k}=\mathrm{a}X_{2k}\mathrm{b})=2$ ➔ $r(v')\geq k$

証明