全単射なBurrowsーWheeler変換の感度指標について

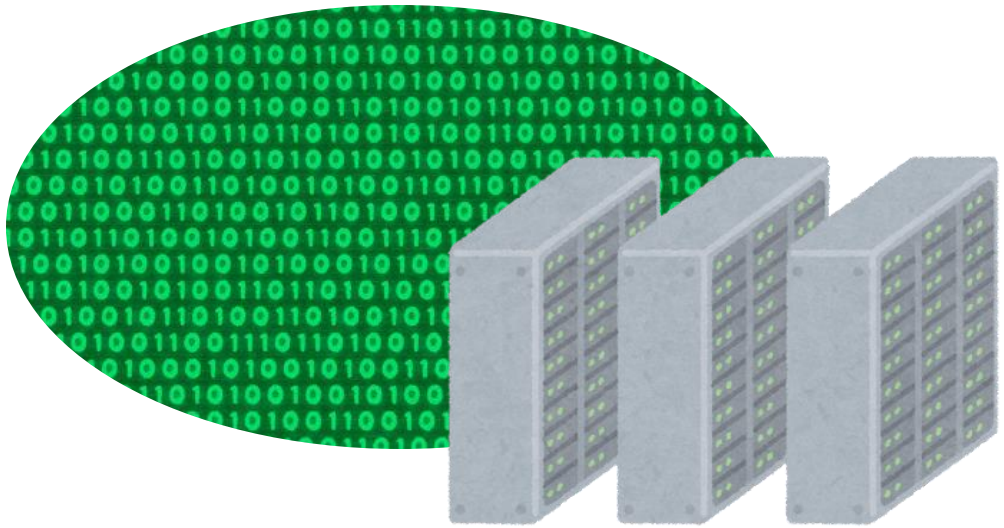
ジョン　ヒョダム、クップル　ドミニク

山梨大学、工学部

第202回AL研究発表会（5）

研究背景

Google, GitHubなどの大きな会社で



大量のデータ（コード, 遺伝子データ)は
コスト削減のために圧縮して保存

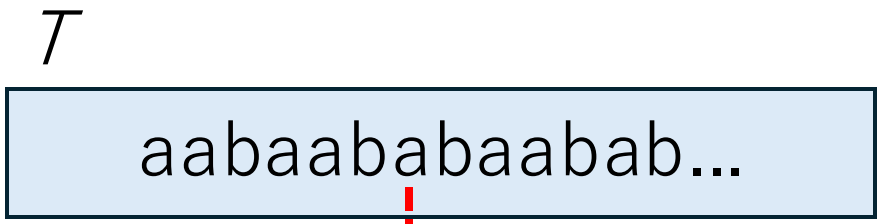
そのとき、予め圧縮後のサイズを予測して保存

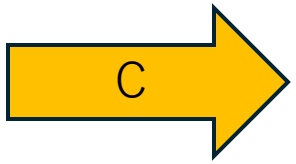
似たデータはほぼ同じ圧縮サイズであるが,
中のわずかな文字の違いで圧縮サイズが大幅に増加し,
予測以上にハードディスクを消費する場合がある

➡問題を起こすパターンを事前に把握するのは，リスク回避に繋がる

研究目的

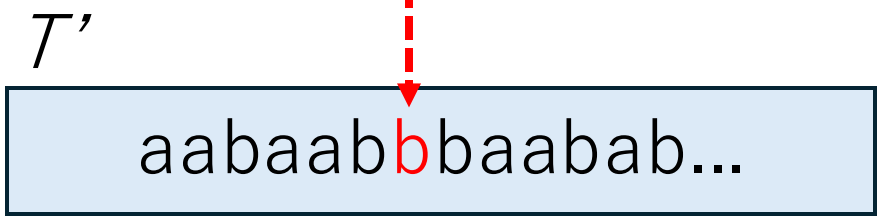
入力テキストの一文字編集は圧縮手法に対してどの程度の影響があるか？

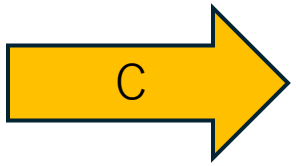




圧縮サイズが小さい
(容量が減る)

圧縮手法 C





圧縮サイズが大きい
(容量が増える)

T' は *T* に１文字の編集操作
(挿入・削除・置換) を
行って得られる文字列
この例では置換

**C(*T'*) は 編集によってC(*T*)とは
別の分解になり、圧縮サイズが変わる.**

**差分感度＝ 圧縮サイズがどの程度大きく
なり得るのかが知りたい**

既存研究

現在，様々な圧縮手法の差分感度の研究が行われている

圧縮手法	既存研究
Lempel-Ziv 78 (LZ78) (画像のデータ圧縮)	Lagarde&Perifel ‘18
Lempel-Ziv 77 (gzipなどで利用)	赤木ら ‘23
BWT (bzip2, 圧縮索引構造)	Giulianiら ‘23
lex-parse	中島ら ‘24
string attractor, bidirectional macro scheme	藤江ら ‘24

しかし，**全単射BWT(BBWT)の圧縮感度**は未だ研究されていない

BBWT の(連長) “クラスタリング効果”

$\boldsymbol{T}$, r(T) = 110

aabaababaabaababaababaabaababab
aababaababaabaababaababaabaababa
abaababaababaabaababaababaababa
baabaababaababaabaababaababaaba
babaabaababaabab

BBWT

BBWT($\boldsymbol{T}$) , r(BBWT(T)) = **2**

bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
bbbbbbbbbbbbbbbbbbbbbbbbbaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaa

- 同じ文字が連続しやすい形に並べ替えることで，データを圧縮しやすくする手法
- BBWT(T) は T と比べて同じ文字が続きやすい→「圧縮しやすく」なる
- 圧縮サイズの指標 r(T)：文字列 T の最大の同一文字の連続する個数 (連超圧縮)

全単射 BWT(BBWT)[Gil&Scott '12]

辞書順 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

Lyndon語

- 文字列 S の巡回文字列の中で辞書式順序が最も小さい文字列
- S がLyndon語でない場合は、
  S をさらにLyndon要素$L_1, L_2, L_3 \ \ldots L_m$ に分解する➡**Lyndon 分解**
  Lyndon要素は辞書式順序の降順で並べられる➡$L_1 \geq L_2 \geq \cdots \geq L_m$

全単射 BWT

- S の全てのLyndon要素の巡回文字列をソートし、最後の文字を連結して得られる
  - BBWT(mammal)= lmamam
  - 圧縮サイズの指標： $\rho(S) = r(\mathrm{BBWT}(S))$

m ❯ a m m ❯ a l

m m a l a

m a m

整列

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

全単射 BWT(BBWT) の逆変換

BBWT(S)からもSに復元できる

復元は、Lyndon要素ごとに復元される

巡回文字列の一番目の要素F	BBWT(mamma$)
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

Lyndon要素の始点と終点は同じ

F	BBWT
a	l
a	m
l	a
m	m
m	a
m	m

m

a m m

a l

a l

a m m

m

降順でソートする

圧縮量の変化

赤字が本研究の成果, (*x)：既存の文字からxに変更又はxを追加
黒字は[Giulianiら 2023]より引用

対象文字列＼圧縮法	編集	BWT	全単射 BWT
Fibonacci文字列 ・$r(\mathrm{BWT}(F_{2k}))=2$ ・$\rho(F_{2k}$のLyndon語$)=2$	最後の文字を削除	**$2k$**	**$\geq k$**
	最後の文字を置換	**$2k$ +2(*a)**	**$\geq k+1$(*#)**
		$2k$ +2(*#)	**$\geq k$ (*c)**
	特定の位置に追加	-	**$\geq k$** **$\geq k+1$**

Fibonacci 文字列

Fibonacci 文字列

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

$F_k = X_k$ab(kは偶数)
$= X_k$ba(kは奇数)

Fibonacci 数列

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

Fibonacci 文字列

Fibonacci 文字列

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

X_k: 回文 palindrome

$F_k = X_k$ab(kは偶数)
$= X_k$ba(kは奇数)

Fibonacci 数列

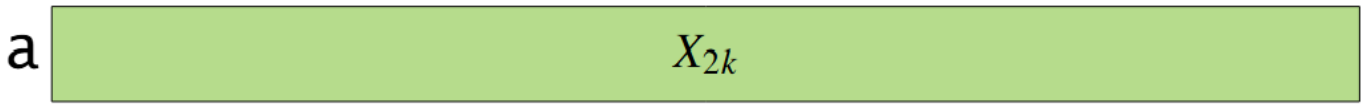
- $f_0 = 1, f_1 = 1, f_k = f_{k-1} + f_{k-2}$
- $f_2 = 2$
- $f_3 = 3$
- $f_4 = 5$
- $f_5 = 8$
- $f_6 = 13$
- $f_7 = 21$

全単射BWTでρが対数的に増える例

編集対象：F_{2k}のLyndon巡回文字列 aX_{2k}b の最後の文字bを削除した文字列 aX_{2k}

定理：$\rho(\mathrm{a}X_{2k}\mathrm{b}) = 2$ であるが，$\rho(\mathrm{a}X_{2k}) \geq k$

証明：異なるLyndon 要素の数が最も多い分解を考える



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

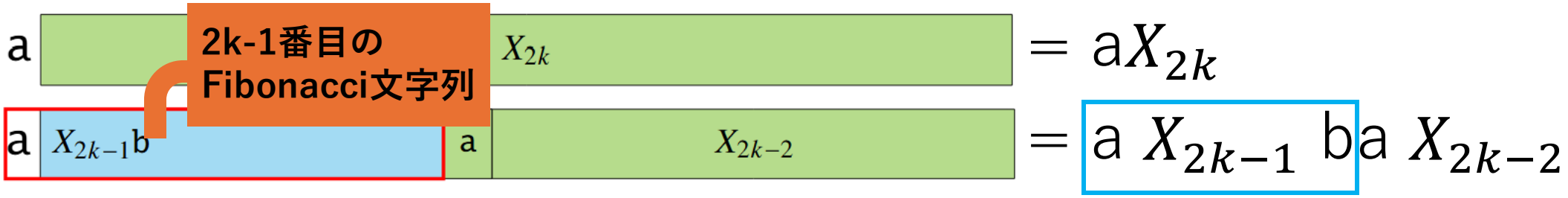
緑はLyndon要素ではない. 青はLyndon要素

全単射BWTでρが対数的に増える例

編集対象：F_{2k}のLyndon巡回文字列 aX_{2k}b の最後の文字bを削除した文字列 aX_{2k}

定理：$\rho(aX_{2k}b) = 2$ であるが，$\rho(aX_{2k}) \geq k$

証明：異なるLyndon 要素の数が最も多い分解を考える



$= aX_{2k}$

$= a\ X_{2k-1}\ ba\ X_{2k-2}$

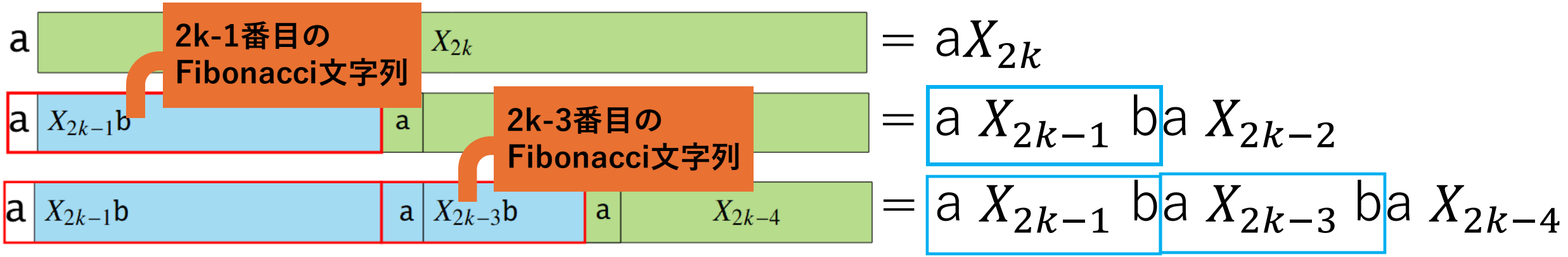
緑はLyndon要素ではない. 青はLyndon要素

全単射BWTでρが対数的に増える例

編集対象：F_{2k}のLyndon巡回文字列 aX_{2k}b の最後の文字bを削除した文字列 aX_{2k}

定理：$\rho(aX_{2k}b) = 2$ であるが，$\rho(aX_{2k}) \geq k$

証明：異なるLyndon 要素の数が最も多い分解を考える



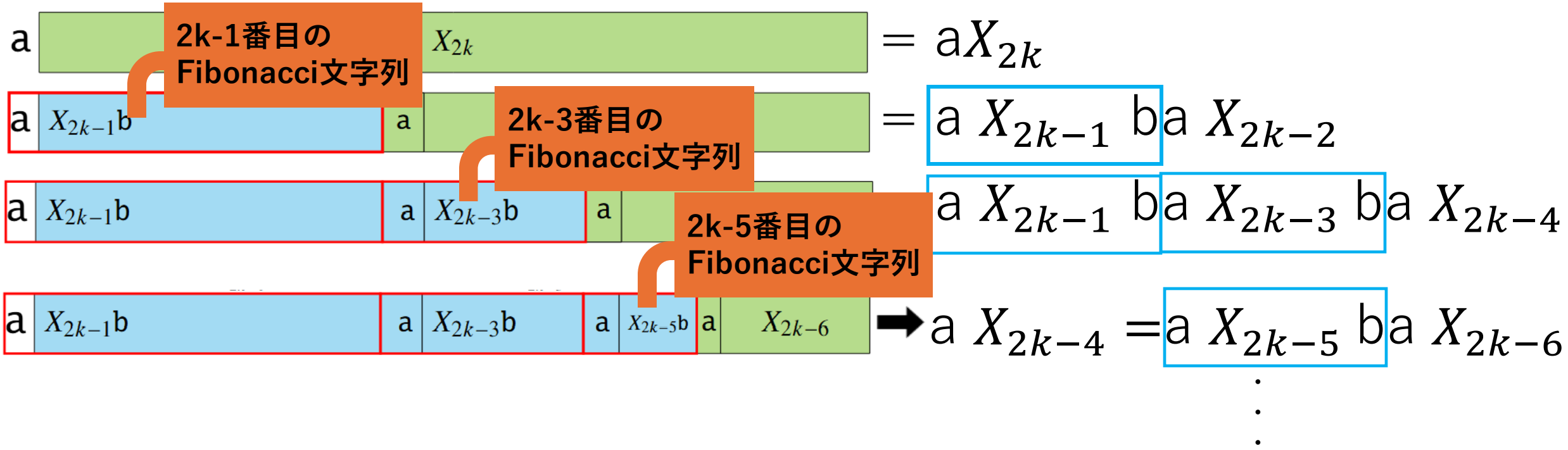
緑はLyndon要素ではない. 青はLyndon要素

全単射BWTでρが対数的に増える例

編集対象：F_{2k}のLyndon巡回文字列 aX_{2k}b の最後の文字bを削除した文字列 aX_{2k}

定理：$\rho(aX_{2k}b) = 2$ であるが，$\rho(aX_{2k}) \geq k$

証明：異なるLyndon 要素の数が最も多い分解を考える



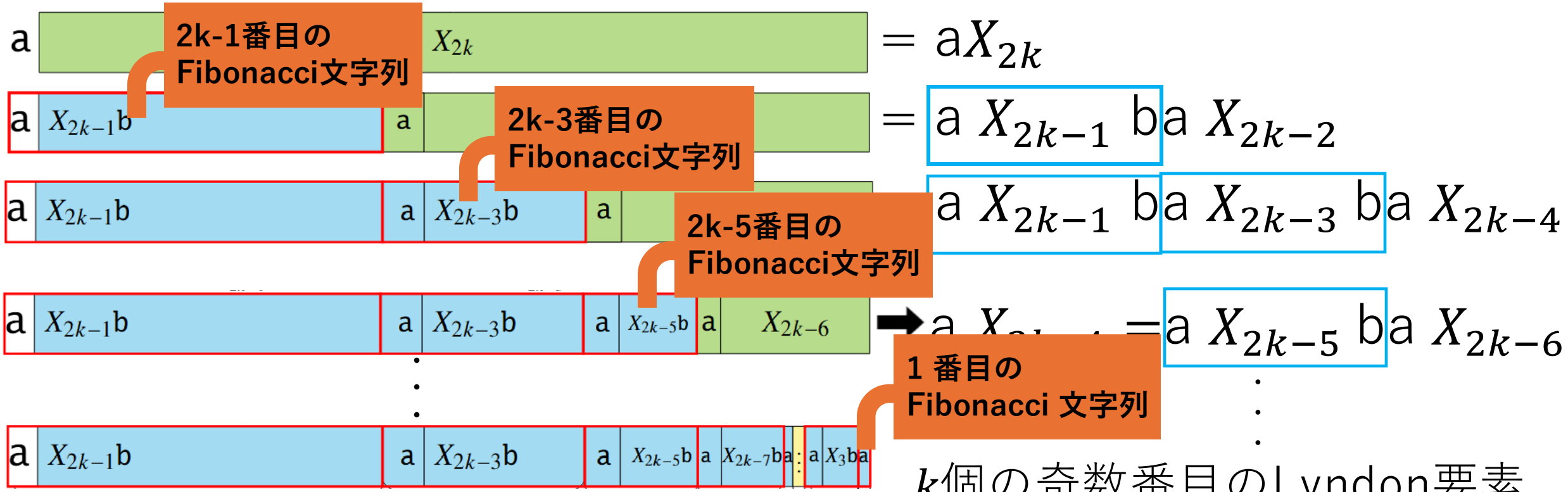
緑はLyndon要素ではない．青はLyndon要素

全単射BWTでρが対数的に増える例

編集対象：F_{2k}のLyndon巡回文字列 aX_{2k}b の最後の文字bを削除した文字列 aX_{2k}

定理：$\rho(aX_{2k}b) = 2$ であるが，$\rho(aX_{2k}) \geq k$

証明：異なるLyndon 要素の数が最も多い分解を考える



緑はLyndon要素ではない．青はLyndon要素

まとめと今後の課題

まとめ

1. 圧縮サイズが$\Theta(\log n)$倍になる文字列があった
2. 発表しない追加結果：圧縮サイズが$+\Theta(\sqrt{n})$増える文字列があった

➡一文字編集で圧縮サイズが大きく変化したことから

➡類似するデータ同士であっても、BWT と全単射 BWT を適用した際に、得られる圧縮サイズが必ずしも一致するとは限らないという可能性がある

今後の課題

・BBWTでFibonacciに変更を加えた結果, Lyndonになる文字列Sの確定
（実験より$\rho(S)=2k$になることは確認済み）

・網羅的に知られている圧縮手法の上下界を求めること