

# 文字列の各文字位置を中心とした パラメタ化回文の最長半径を探索するアルゴリズム

---

2022/10/24

篠原・吉仲研究室 修士1年 松野 直也

# 概要と目次

---

- Manacherの回文の最長半径探索アルゴリズムを拡張した最長パラメタ化回文を探索アルゴリズムについて。

## 目次

- 導入
  - ・パラメタ化照合
  - ・パラメタ化回文
- Manacherのアルゴリズム
- 拡張版アルゴリズム
- まとめ

# パラメタ化照合 [Baker, 1993]

パラメタ化文字列:  $\Sigma \cup \Pi$  上の文字列

$\Sigma$ : 定数文字の集合  $\Pi$ : 変数文字の集合

## パラメタ化一致

2つのパラメタ化文字列 $S, T$ に対して、 $S$ と $T$ が等しくなるような変数文字同士の  
一対一対応が存在するならば、 $S$ と $T$ はパラメタ化一致するという。

## 例

$A, B, C \in \Sigma$        $x, y, z \in \Pi$

$S = A B x C x y$

$T = A B z C z x$

$x \rightarrow z$

$y \rightarrow x$



$S$  と  $T$  はパラメタ化一致する

$S \approx T$

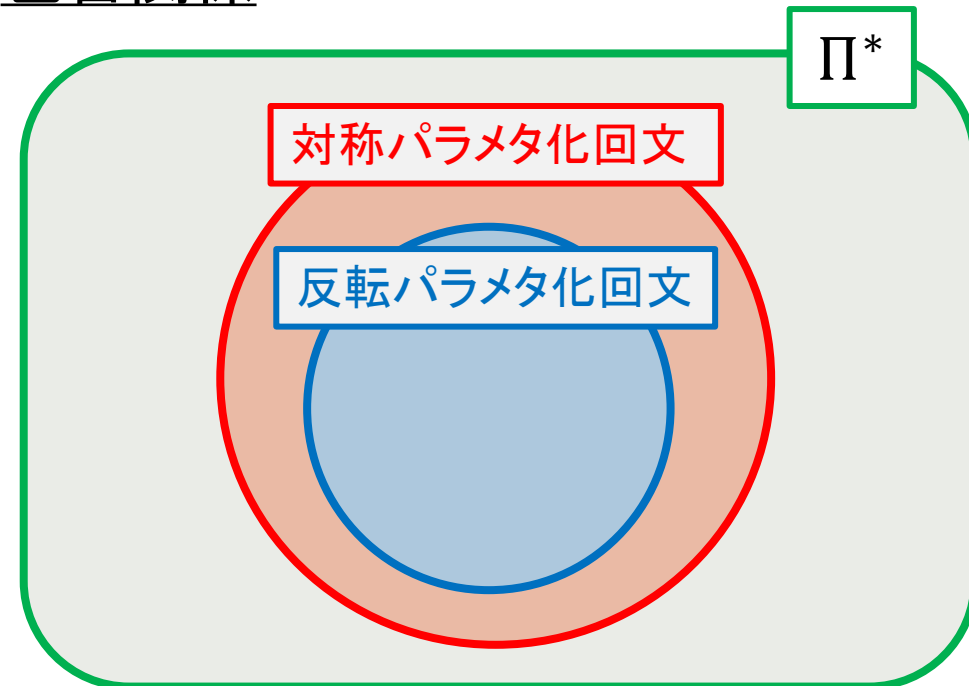
# パラメタ化回文

パラメタ化回文とは通常的回文を拡張したもので、パラメタ化一致を許した場合に回文となる文字列のことである。

定義としては以下の2つを定める。

- 反転パラメタ化回文
- 対称パラメタ化回文

包含関係



# 反転パラメタ化回文

## 定義

文字列  $w \in \Pi^*$  に対し、それを反転した文字列を  $w^R$  とすると  $w \approx w^R$  となるような文字列

## 例

$w = a b c a b$  のとき

$w^R = b a c b a$  となる。

このとき  $w$  と  $w^R$  はパラメタ化一致するので、 $w$  は反転パラメタ化回文である。

今回紹介するアルゴリズムはこれを見つけるもの。

# 対称パラメタ化回文

## 定義

文字列  $ucv \in \Pi^*$  に対し、 $|u| = |v|$  とすると  $u \approx v^R$  となるような文字列

## 例

$u = a b c a, v = d e f d, c = c$  のとき

$v^R = d f e d$  となる。

このとき  $u$  と  $u$  はパラメタ化一致するので、 $ucv = a b c a c d e f d$  は対称パラメタ化回文である。

# 回文の最長半径の探索 [Manacher, 1975]

---

- 文字列  $S$  に対して、各文字  $S[i]$  ( $1 \leq i \leq |S|$ ) を中心としたときの回文の最長半径を求めるアルゴリズム。
- 計算の再利用をすることで計算量  $O(|S|)$  で探索が可能。

# 回文の最長半径の求め方

---

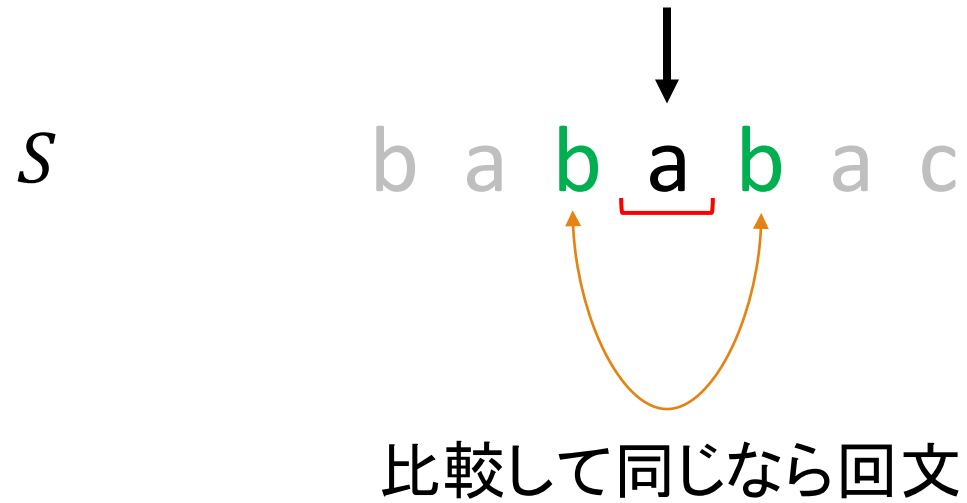
↓  
S            b a b a b a c

中心aから徐々に半径を伸ばしていくことで回文の最長半径を求める。



# 回文の最長半径の求め方

---



# 回文の最長半径の求め方

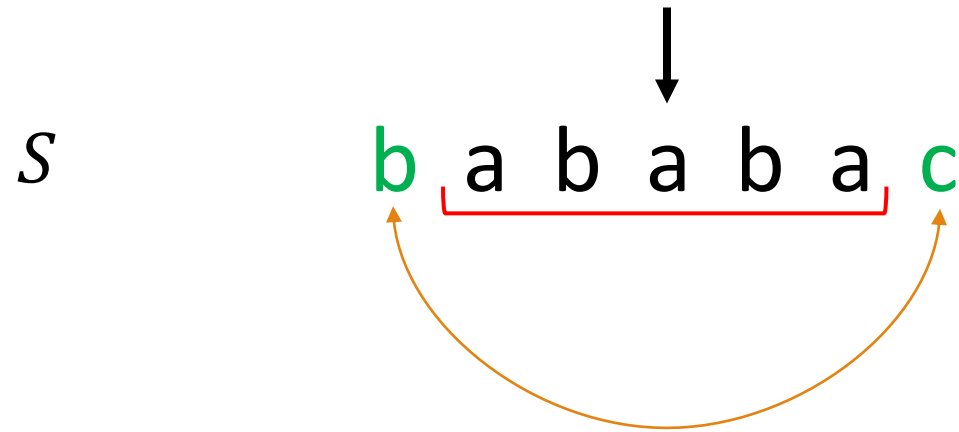
---

$s$                       b   a   b a b   a   c

↓

# 回文の最長半径の求め方

---




異なる文字なので回文にならない。  
よって最長半径は3。

# 計算の再利用

Manacherのアルゴリズムでは文字の先頭から順番に最長半径を求めていくが、その過程で計算量を減らすために既に計算した最長半径を利用するという工夫がなされている。

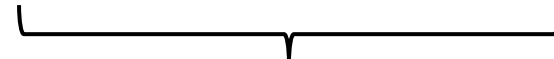
↓

$S$       a b a b b b b b a a b



最長半径

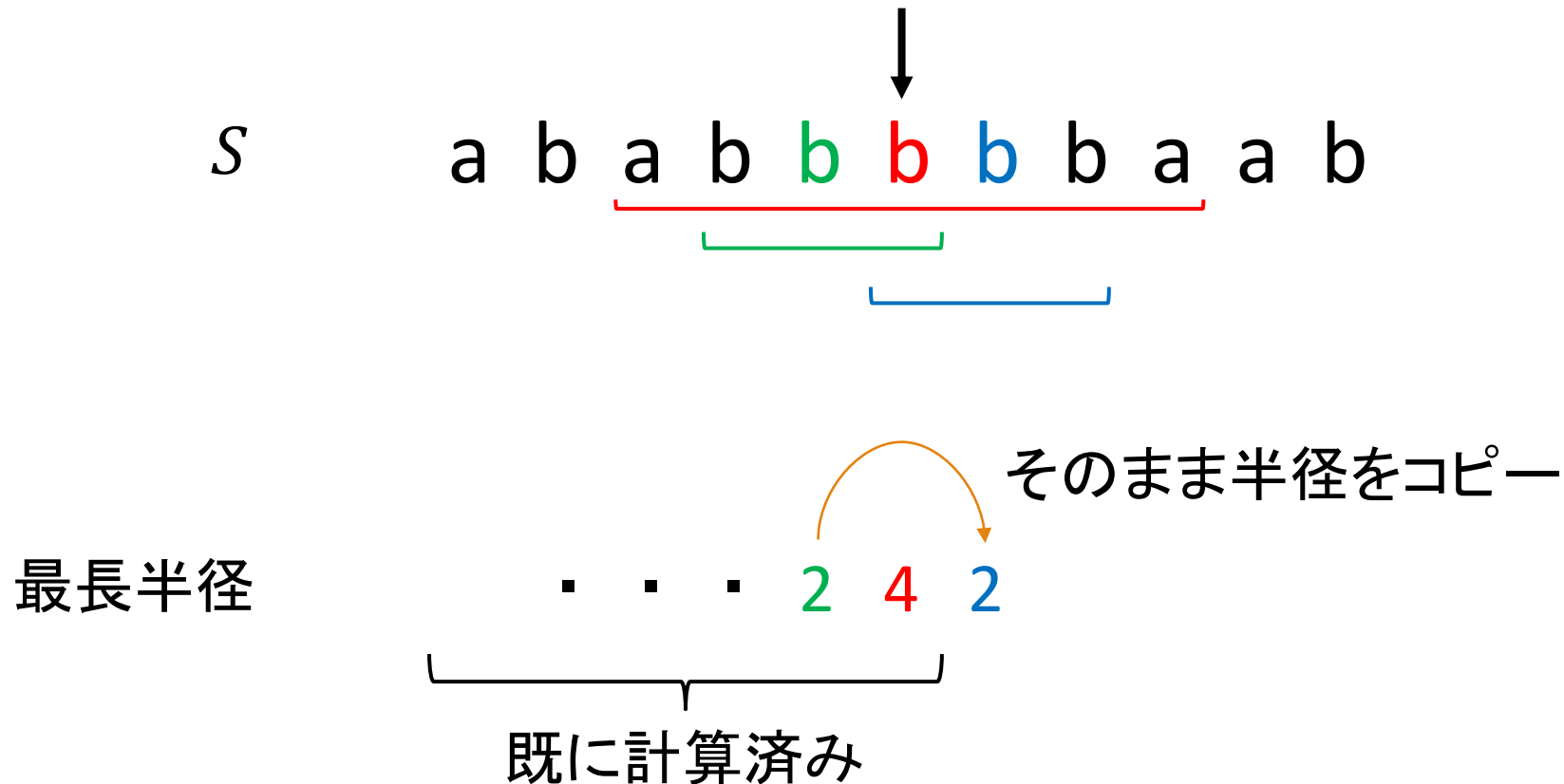
▪   ▪   ▪   2   4



既に計算済み

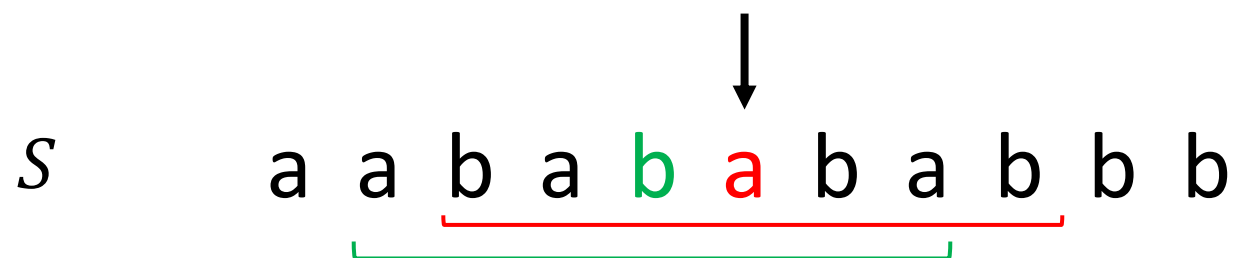
# 計算の再利用

Manacherのアルゴリズムでは文字の先頭から順番に最長半径を求めていくが、その過程で計算量を減らすために既に計算した最長半径を利用するという工夫がなされている。

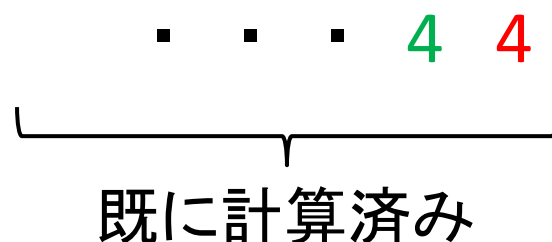


# 計算の再利用

単純にコピーできない場合もある。

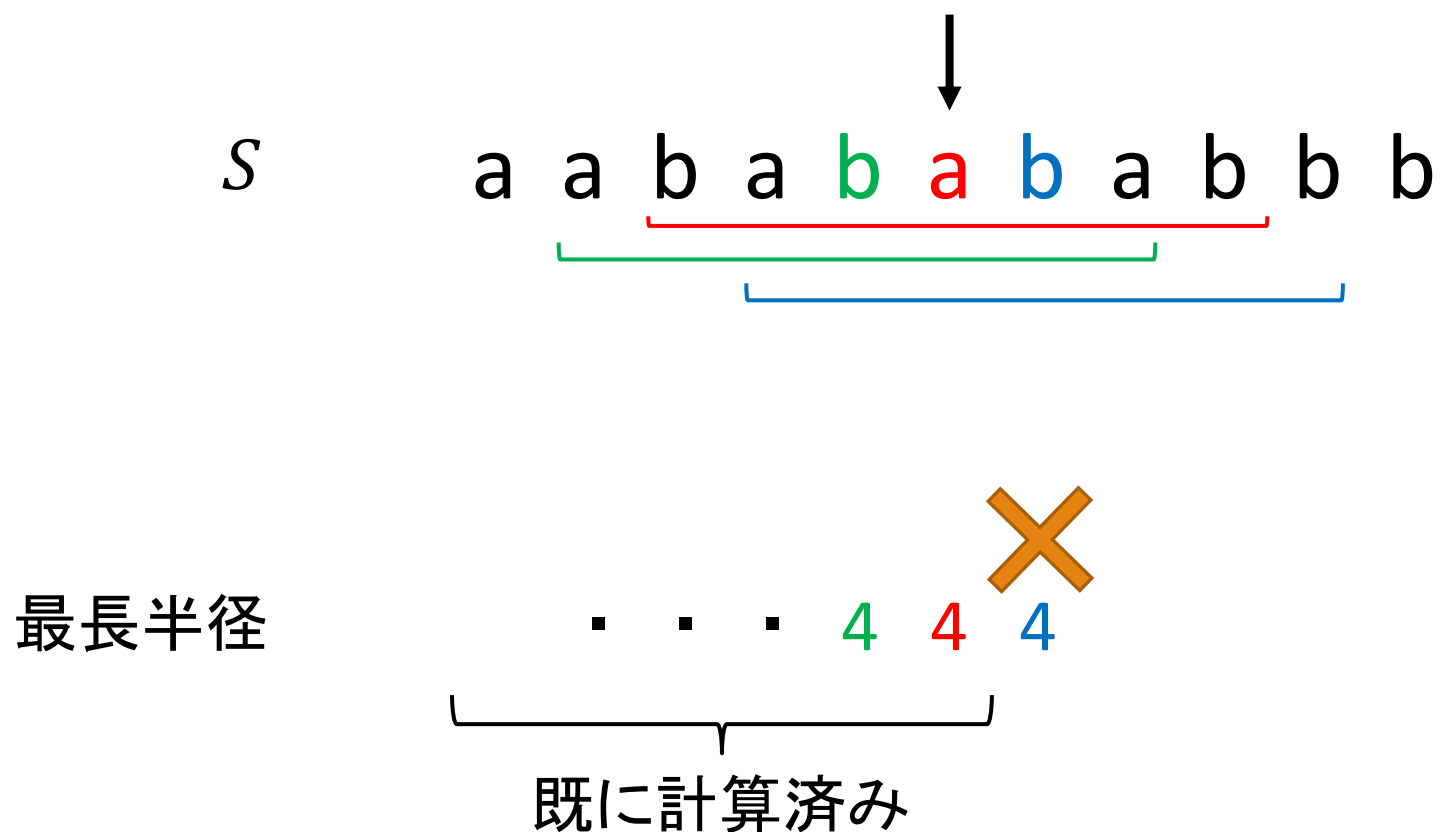


最長半径



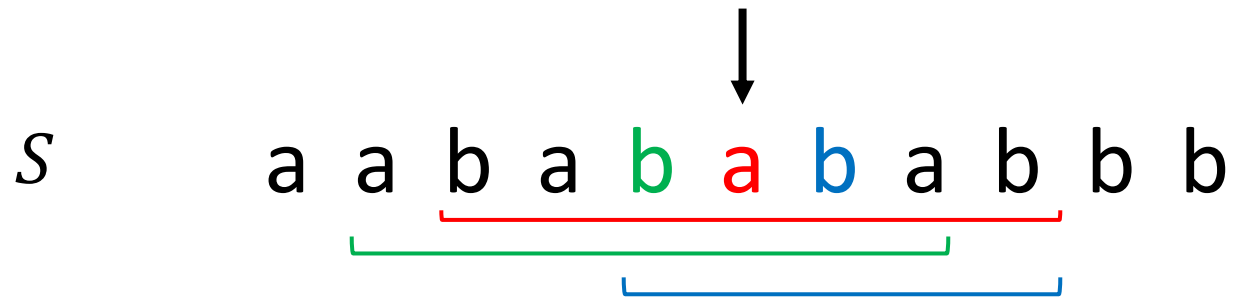
# 計算の再利用

単純にコピーできない場合もある。



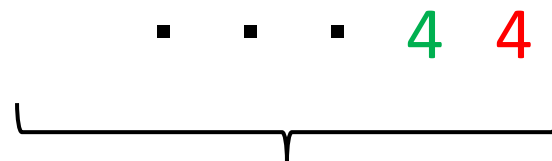
# 計算の再利用

単純にコピーできない場合もある。



半径3までは回文なので、  
bを中心とし半径4から愚直に計算。

最長半径



既に計算済み



# アルゴリズムの動作例

---

$s$       ↓  
a b c b a b a b a a b

最長半径

# アルゴリズムの動作例

---


$s$       ↓  
a b c b a b a b a a b

最長半径      1

# アルゴリズムの動作例

---

$s$       a   **b**   c   b   a   b   a   b   a   a   b



最長半径      1

# アルゴリズムの動作例

---


$s$       a b c b a b a b a a b

最長半径      1

# アルゴリズムの動作例

---

$s$       a b c b a b a b a a b




最長半径      1   1

# アルゴリズムの動作例

---

$s$       a b c b a b a b a a b

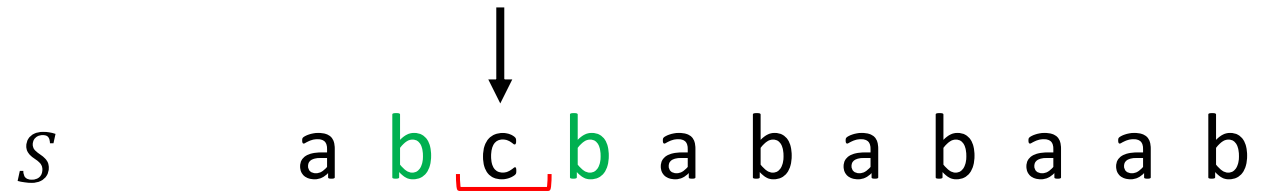


最長半径      1 1

# アルゴリズムの動作例

---

$s$       a   **b**   c   **b**   a   b   a   b   a   a   b

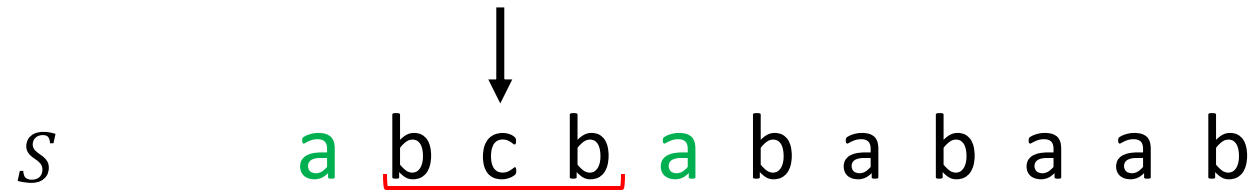


最長半径      1   1

# アルゴリズムの動作例

---

$s$       a b c b a b a b a a b



最長半径      1   1



# アルゴリズムの動作例

---

$s$       a b c b a b a b a a b

↓

最長半径      1   1   3

# アルゴリズムの動作例

---

$s$       a b c b a b a b a a b

↓

└─┘

最長半径      1   1   3   1

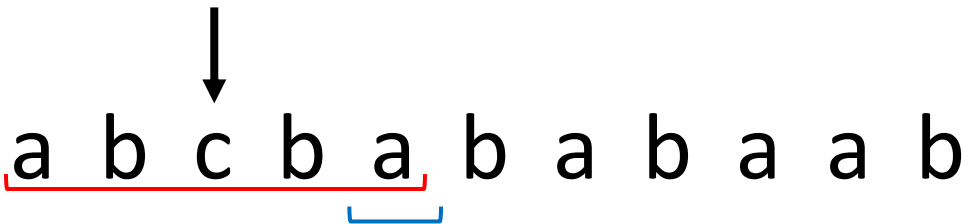
↪

計算の再利用  
半径は1

# アルゴリズムの動作例

---

$s$       a b c b a b a b a a b



最長半径      1   1   3   1



計算の再利用  
半径は1以上

# アルゴリズムの動作例

---

$s$       a b c **b** a **b** a b a a b

最長半径      1 1 3 1

# アルゴリズムの動作例

---

$s$       a b c b a b a b a a b

↓

最長半径      1 1 3 1

# アルゴリズムの動作例

---

↓

$s$       a b c b a b a b a a b

最長半径      1 1 3 1 2

# アルゴリズムの動作例

---

$s$       a b c b a b a b a a b

最長半径    1 1 3 1 2 3 3 2 1 1 1

# 反転パラメタ化回文への応用

---

Manacherのアルゴリズムを拡張して、各文字位置を中心とした最長のパラメタ化回文を見つけられないか？

とりあえず最長の反転パラメタ化回文を探索するようにアルゴリズムを考えてみた。

$$w \approx w^R$$

アルゴリズム自体は考え付き、証明を行っていかうとしていたところ以下の論文で先に発表されてしまった。

## 論文

Funakoshi Mitsuru, Takuya Mieno, Yuto Nakashima et al.

Computing maximal generalized palindromes.

*arXiv preprint arXiv:2210.02067*, 2022.



# 直前符号化 [Baker, 1993]

文字列  $S$  の  $i$  番目の文字  $S[i]$  に対する直前符号化

$$P(S)[i] = \begin{cases} S[i] & \text{if } S[i] \in \Sigma \\ 0 & \text{if } S[i] \in \Pi \wedge S[i] \neq S[j] \text{ for all } 1 \leq j < i \\ i - \max\{j \mid S[i] = S[j] \wedge 1 \leq j < i\} & \text{otherwise} \end{cases}$$

例  $A, B, C \in \Sigma$        $x, y, z \in \Pi$

$S = A B x C x y z x$   
                    ↑          ↑  
                  2文字前  3文字前

$P(S) = A B 0 C 2 0 0 3$

補題

$$P(S) = P(T) \Leftrightarrow S \approx T$$

# 直後符号化

文字列  $S$  の  $i$  番目の文字  $S[i]$  に対する直後符号化

$$N(S)[i] = \begin{cases} S[i] & \text{if } S[i] \in \Sigma \\ 0 & \text{if } S[i] \in \Pi \wedge S[i] \neq S[j] \text{ for all } i < j \leq |S| \\ \min\{j \mid S[i] = S[j] \wedge i < j \leq |S|\} - i & \text{otherwise} \end{cases}$$

例  $A, B, C \in \Sigma \quad x, y, z \in \Pi$

$S = x z y x C x B A$   
3文字後    2文字後

$N(S) = 3 \ 0 \ 0 \ 2 \ C \ 0 \ B \ A$

$S$ の直後符号化列を反転したものは  
 $S^R$ の直前符号化列と等価。

$$(N(S))^R = P(S^R)$$

$$P(S) = P(S^R) \Leftrightarrow S \approx S^R$$



$$P(S) = (N(S))^R \Leftrightarrow S \approx S^R$$

# 反転パラメタ化回文の最長半径の探索

---

## 全体の流れ

- まず前処理として文字列  $S$  全体を直前符号化および直後符号化する。  
アルファベットサイズが定数であればそれぞれ  $O(|S|)$  で処理可能。
- $S$  の先頭から愚直に最長半径を求めていく。(文字の比較処理が通常的回文のものと異なる)
- 計算の再利用が可能な場合は利用する。(ここの処理は同じ)

# 半径を伸ばす過程での比較

## 命題

反転パラメタ化回文  $u$  と  $x, y \in \Pi$  に対して  $S = xuy$  とすると、  
 $P(S)[|S|] = N(S)[1]$  ならば  $S$  は反転パラメタ化回文である。



回文を伸ばしていく過程での比較処理1つ1つが定数時間で処理できる。

# 証明

$$S = xuy$$

$P(S)[|S|] = N(S)[1] = k$ として考える。

$k = 0$ のとき

これは  $x, y$  が  $u$  に含まれていない。

このとき  $u$  にあたる部分の符号化は影響を受けない。

よって  $P(S) = (N(S))^R$  であるといえるので、 $S$  は反転パラメタ化回文である。

	x	a	b	c	b	d	y
直前符号化		0	0	0	2	0	
	0	0	0	0	2	0	0
直後符号化		0	2	0	0	0	
	0	0	2	0	0	0	0

# 証明

$$S = xuy$$

$P(S)[|S|] = N(S)[1] = k$ として考える。

$0 < k < |S|/2$ のとき

$x, y$  が  $u$  に含まれている。

$u = u_1 \text{ } x \text{ } u_2 \text{ } y \text{ } u_3$ とする。(ただし  $x \notin u_1$  かつ  $y \notin u_3$ )

このとき  $S = x \text{ } u_1 \text{ } x \text{ } u_2 \text{ } y \text{ } u_3 \text{ } y$

$|u_1| = |u_3|$ であるので、 $P(S)[k] = N(S)[|S| - k + 1]$ といえる。

またこれ以外の部分も条件より  $P(S)[i] = N(S)[|S| - i + 1]$ といえる。

よって  $P(S) = (N(S))^R$  が成り立つので、 $S$  は反転パラメタ化回文である。

	$x$	$a$	$x$	$b$	$y$	$c$	$y$
直前符号化		0	0	0	0	0	
	0	0	2	0	0	0	2
直後符号化		0	0	0	0	0	
	2	0	0	0	2	0	0

# 証明

$$S = xuy$$

$P(S)[|S|] = N(S)[1] = k$ として考える。

$k = |S|/2$ のとき

$x, y$  が  $u$  に含まれているかつ  $x = y$ 。

$u = u_1 \text{ } x \text{ } u_2$ とする。(ただし  $x \notin u_1$  かつ  $x \notin u_2$ )

このとき  $S = x \text{ } u_1 \text{ } x \text{ } u_2 \text{ } x$

$|u_1| = |u_2|$ であるので、 $P(S)[k] = N(S)[|S| - k + 1]$ といえる。

またこれ以外の部分も条件より  $P(S)[i] = N(S)[|S| - i + 1]$ といえる。

よって  $P(S) = (N(S))^R$  が成り立つので、 $S$  は反転パラメタ化回文である。

	$x$	$a$	$b$	$x$	$b$	$c$	$x$
直前符号化		0	0	0	2	0	
	0	0	0	3	2	0	3
直後符号化		0	2	0	0	0	
	3	0	2	3	0	0	0

# 証明

$$S = xuy$$

$P(S)[|S|] = N(S)[1] = k$ として考える。

$k > |S|/2$ のとき

$x, y$  が  $u$  に含まれている。

$u = u_1 \textcolor{red}{y} u_2 \textcolor{red}{x} u_3$ とする。(ただし  $x \notin u_1 \cup u_2$  かつ  $y \notin u_3 \cup u_2$ )

このとき  $S = x u_1 \textcolor{red}{y} u_2 \textcolor{red}{x} u_3 y$

$|u_1 \textcolor{red}{y} u_2| = |u_2 \textcolor{red}{x} u_3|$ であるので、 $P(S)[k] = N(S)[|S| - k + 1]$ といえる。

またこれ以外の部分も条件より  $P(S)[i] = N(S)[|S| - i + 1]$ といえる。

よって  $P(S) = (N(S))^R$  が成り立つので、 $S$  は反転パラメタ化回文である。

	$\textcolor{red}{x}$	$a$	$y$	$b$	$x$	$c$	$\textcolor{blue}{y}$
直前符号化	0	0	0	0	0	0	
	$\textcolor{red}{0}$	0	0	0	$\textcolor{red}{4}$	0	$\textcolor{blue}{4}$
直後符号化		0	0	0	0	0	
	$\textcolor{red}{4}$	0	$\textcolor{red}{4}$	0	0	0	$\textcolor{blue}{0}$

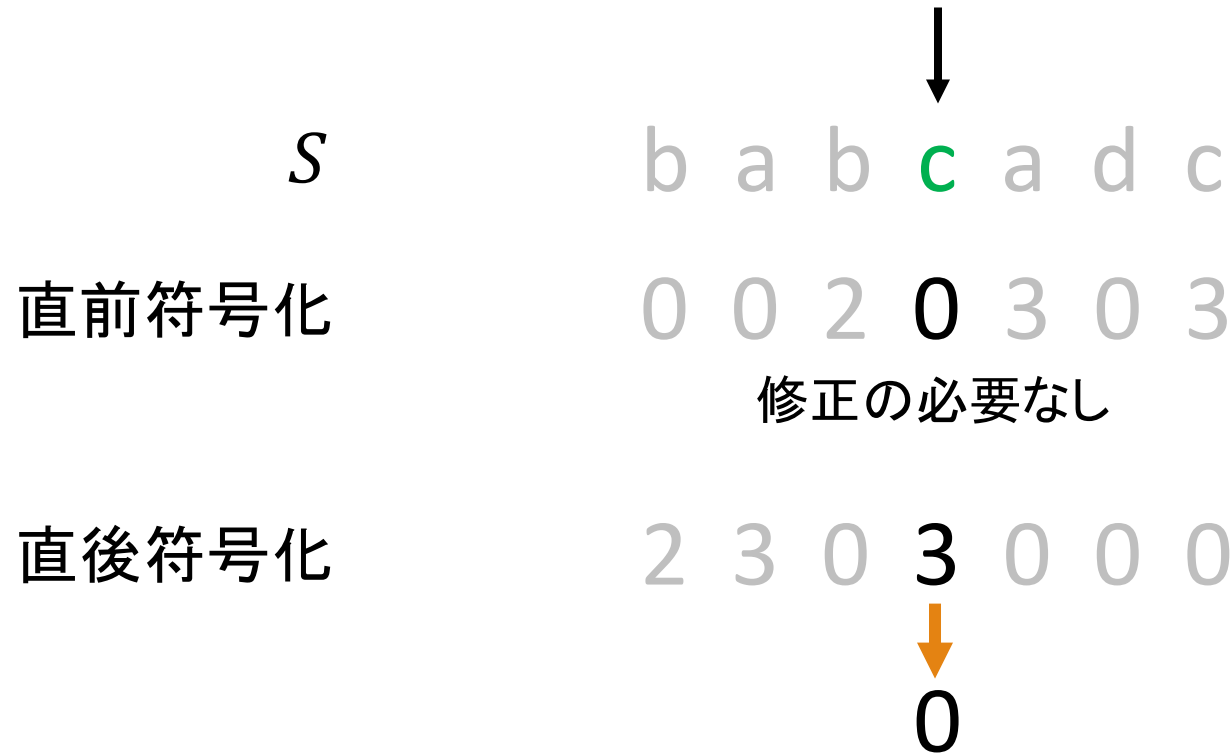


# 反転パラメタ化回文の最長半径の求め方

				↓			
$S$	b	a	b	c	a	d	c
直前符号化	0	0	2	0	3	0	3
直後符号化	2	3	0	3	0	0	0

$S$  の部分文字列  $T$  の最後の文字の直前符号化  $P(T)[|T|]$  と  
 $S$  の部分文字列  $T$  の最初の文字の直後符号化  $N(T)[1]$  を比較する。  
比較の際には全体の符号化を修正する必要がある。

# 反転パラメタ化回文の最長半径の求め方

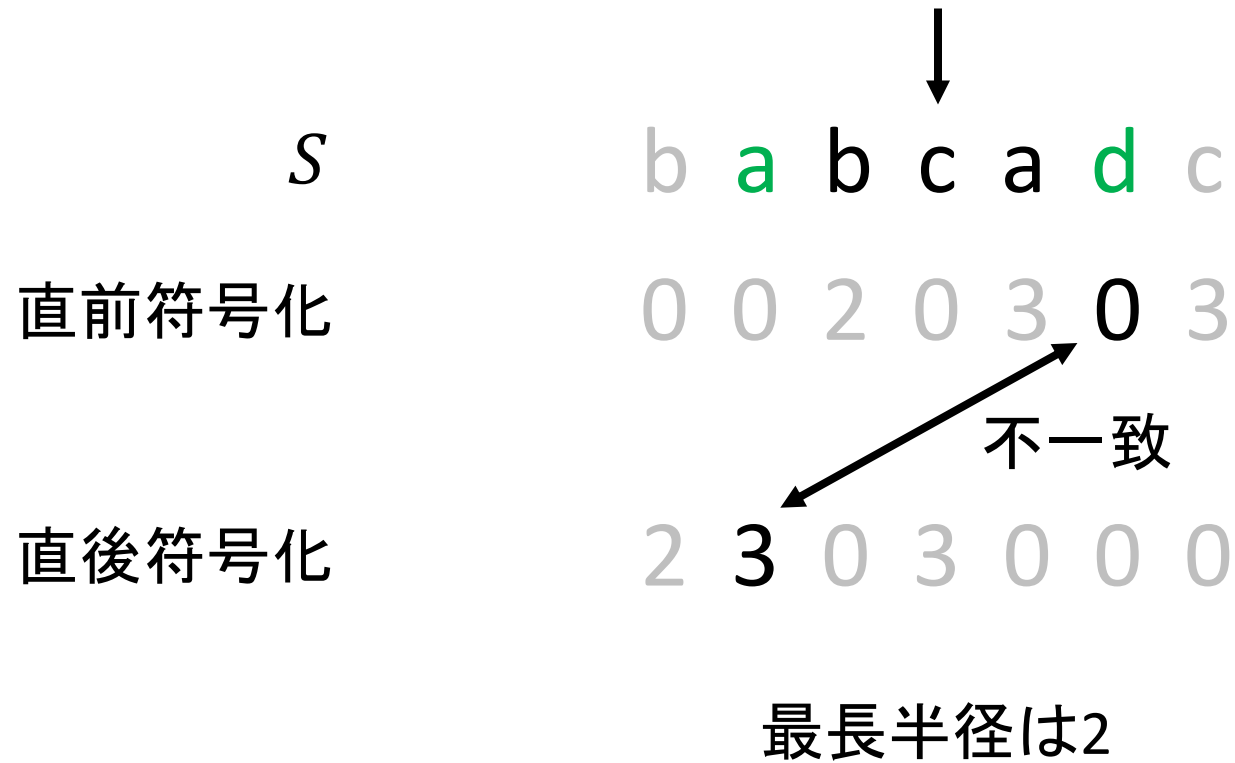


比較する場所の符号化は部分文字列の大きさ  $|T|$  以上にはならない。  
もし  $|T|$  以上になっていたときは0に修正する。

# 反転パラメタ化回文の最長半径の求め方



# 反転パラメタ化回文の最長半径の求め方



# アルゴリズムの動作例

---

	↓										
$s$	a	b	c	b	a	d	a	b	a	c	b
直前符号化	0	0	0	2	4	0	2	4	2	7	4
直後符号化	4	2	7	4	2	0	2	3	0	0	0
	↓										
	0										
最長半径											



# アルゴリズムの動作例

---

	↓										
$s$	<u>a</u>	b	c	b	a	d	a	b	a	c	b
直前符号化	0	0	0	2	4	0	2	4	2	7	4
直後符号化	4	2	7	4	2	0	2	3	0	0	0
最長半径	1										

# アルゴリズムの動作例

---

$s$	a	 b	c	b	a	d	a	b	a	c	b
直前符号化	0	0	0	2	4	0	2	4	2	7	4
直後符号化	4	2	7	4	2	0	2	3	0	0	0
											
最長半径	1	0									

# アルゴリズムの動作例

---

$s$	a	b	c	b	a	d	a	b	a	c	b
		↓									
直前符号化	0	0	0	2	4	0	2	4	2	7	4
直後符号化	4	2	7	4	2	0	2	3	0	0	0
	↓										
最長半径	0										
	1										

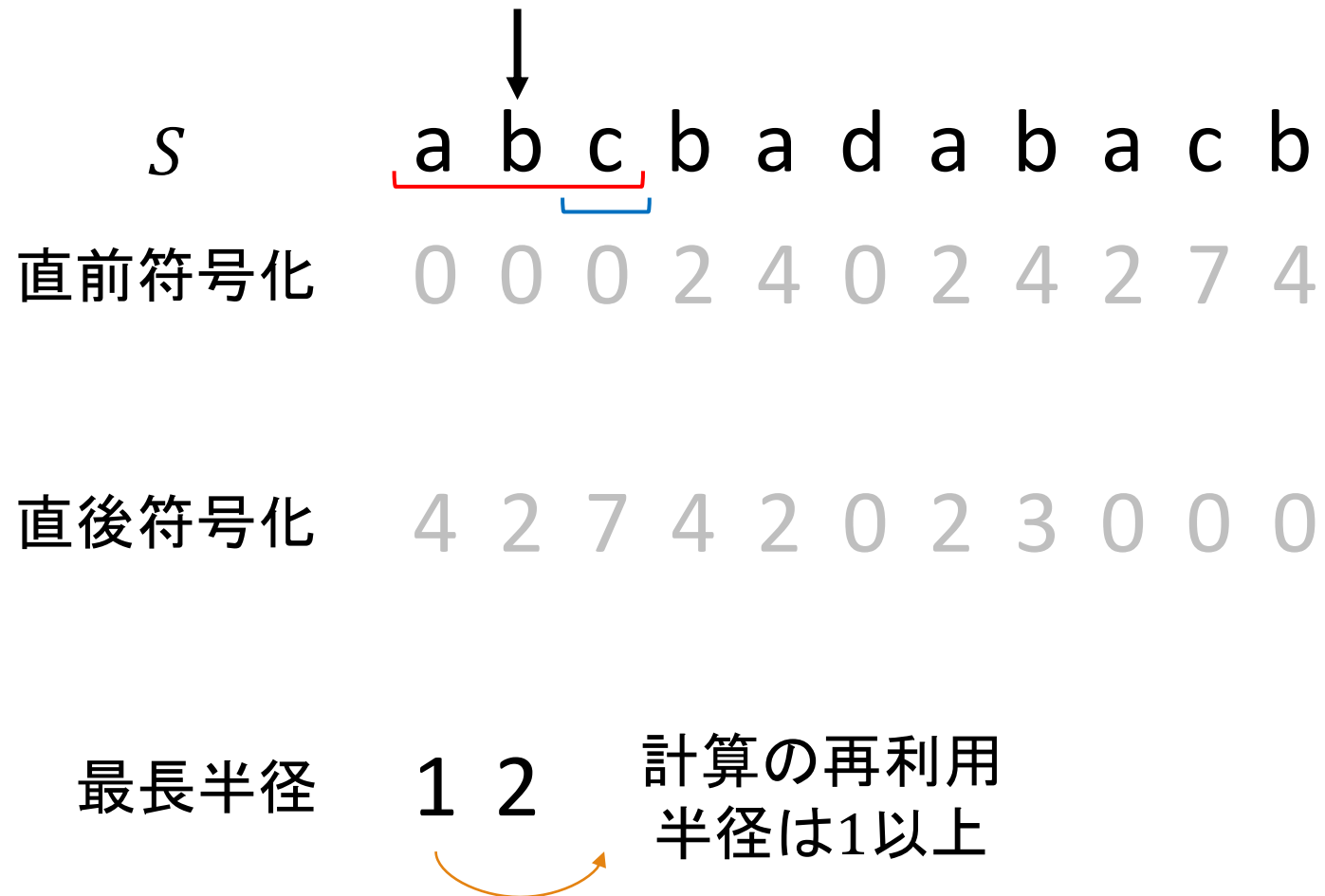


# アルゴリズムの動作例

---

			↓									
$S$		<u>a</u>	<u>b</u>	<u>c</u>	b	a	d	a	b	a	c	b
直前符号化		0	0	0	2	4	0	2	4	2	7	4
直後符号化		4	2	7	4	2	0	2	3	0	0	0
最長半径		1	2									

# アルゴリズムの動作例



# アルゴリズムの動作例

---

			↓								
$s$	a	b	<u>c</u>	b	a	d	a	b	a	c	b
直前符号化	0	0	0	2	4	0	2	4	2	7	4
直後符号化	4	2	7	4	2	0	2	3	0	0	0
最長半径	1	2									

# アルゴリズムの動作例

---

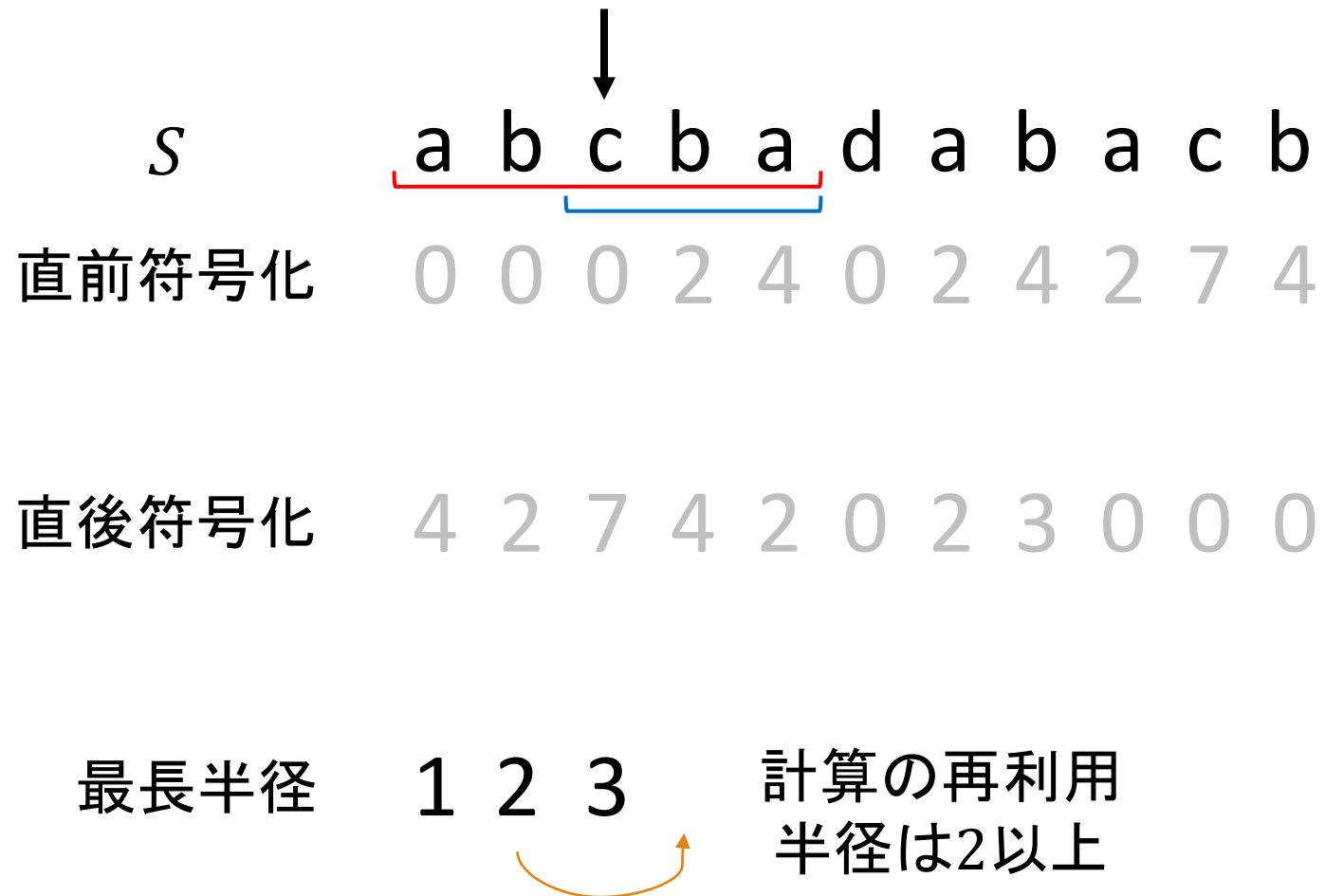
			↓								
$s$	a	b	c	b	a	d	a	b	a	c	b
直前符号化	0	0	0	2	4	0	2	4	2	7	4
直後符号化	4	2	7	4	2	0	2	3	0	0	0
最長半径	1	2									

# アルゴリズムの動作例

---

			↓								
$s$	<u>a</u>	b	c	b	a	d	a	b	a	c	b
直前符号化	0	0	0	2	4	0	2	4	2	7	4
直後符号化	4	2	7	4	2	0	2	3	0	0	0
最長半径	1	2	3								

# アルゴリズムの動作例



# アルゴリズムの動作例

---

			↓								
$S$	a	b	<u>c</u>	b	a	d	a	b	a	c	b
直前符号化	0	0	0	2	4	0	2	4	2	7	4
直後符号化	4	2	7	4	2	0	2	3	0	0	0
最長半径	1	2	3								

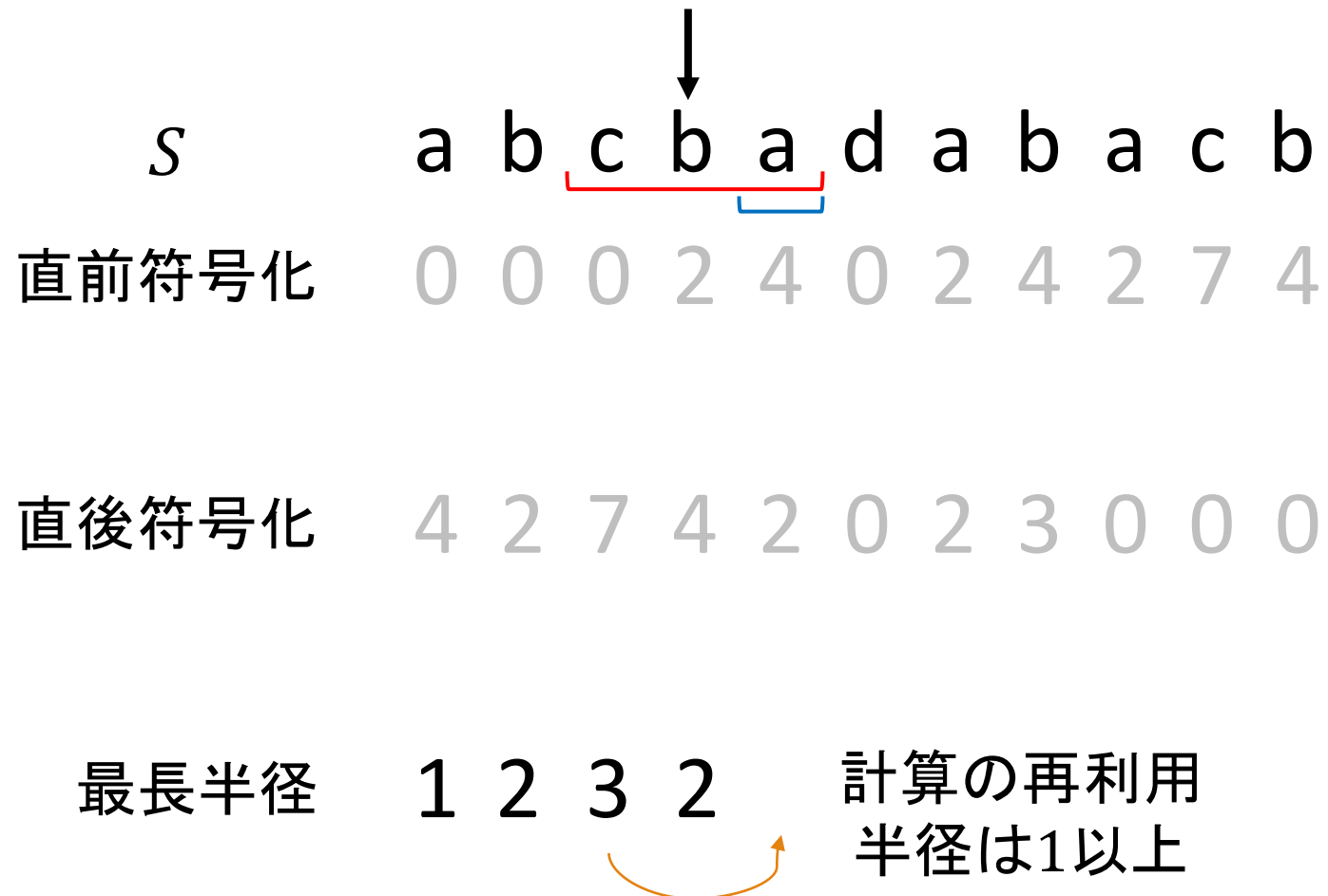
# アルゴリズムの動作例

---

			↓								
$s$	a	b	<u>c</u>	<u>b</u>	<u>a</u>	d	a	b	a	c	b
直前符号化	0	0	0	2	4	0	2	4	2	7	4
直後符号化	4	2	7	4	2	0	2	3	0	0	0
最長半径	1	2	3	2							



# アルゴリズムの動作例



# アルゴリズムの動作例

---

				↓							
$S$	a	b	c	b	<u>a</u>	d	a	b	a	c	b
直前符号化	0	0	0	2	4	0	2	4	2	7	4
直後符号化	4	2	7	4	2	0	2	3	0	0	0
				↓							
				0							
最長半径	1	2	3	2							

# アルゴリズムの動作例

---

$s$	a	b	c	<u>b</u>	a	d	a	b	a	c	b
直前符号化	0	0	0	2	4	0	2	4	2	7	4
直後符号化	4	2	7	4	2	0	2	3	0	0	0
			↓								
			0								
最長半径	1	2	3	2							

# アルゴリズムの動作例

---

				↓							
$S$	a	b	c	<u>b</u>	a	d	a	b	a	c	b
直前符号化	0	0	0	2	4	0	2	4	2	7	4
直後符号化	4	2	7	4	2	0	2	3	0	0	0
最長半径	1	2	3	2	2						

# アルゴリズムの動作例

---

$s$	a	b	c	b	a	d	a	b	a	c	b
直前符号化	0	0	0	2	4	0	2	4	2	7	4
直後符号化	4	2	7	4	2	0	2	3	0	0	0
最長半径	1	2	3	2	2	3	3	3	2	2	1

# まとめと今後の方針

---

- Manacherの最長回文探索アルゴリズムをパラメタ化回文に適応させたアルゴリズムを紹介した。
- 論文中ではもう一つの定義である対称パラメタ化回文をManacherアルゴリズムで探索することはやっていなかったなのでそれを考えていたが、計算の再利用部分が使えない場合が多くあり、 $O(|S|)$ で実現するのは難しい。
- 現在研究テーマを別のものにすることを検討中。

# 参考

---

[1] Brenda S. Baker

Parameterized duplication in strings: Algorithms and an application to software maintenance.

SIAM Journal on Computing 26.5 (1997): 1343-1362.

[2] Manacher Glenn.

A New Linear-Time "On-Line" Algorithm for Finding the Smallest Initial Palindrome of a String.

*Journal of the ACM (JACM)*, 1975, 22.3: 346-351.

[3] Funakoshi Mitsuru, Takuya Mieno, Yuto Nakashima et al.

Computing maximal generalized palindromes.

*arXiv preprint arXiv:2210.02067*, 2022.