パラメタ化BW変換のオンライン構築に関する研究

修士学位論文

パラメタ化BW変換のオンライン構築に関する研究

東北大学 大学院情報科学研究科

システム情報科学専攻 篠原・吉仲研究室

博士課程前期2年の課程

橋本　大輝

2022年1月28日

目次

第1章

序論

1.1 はじめに

BW 変換とは，Burrows と Wheeler [2] によって提案された文字列の可逆変換手法の一種である．BW 変換によって生成される文字列は，元の文字列と比較して同じ文字が連続して出現しやすいという点から，連長圧縮が元の文字列よりも効果的なことが知られており，連長圧縮 BW 変換 [9] などの変種の変換手法の研究や bzip2 などの圧縮手法の一部で BW 変換が用いられているなど多くの場所で用いられている．また，BW 変換後の文字列と変換後の文字列を辞書式順序でソートした文字列を用いることで，変換前の文字列での厳密文字列照合問題が行える FM インデックス [4] なども知られている．FM インデックスは連長圧縮 BW 変換等と組み合わせて，DNA などの大量のデータを圧縮した状態で検索を行う時などに用いられる手法となっている．長さ n の文字列 T の BW 変換は，オフラインでは $O(n)$ 時間 [6]，オンラインでは $O(n\frac{\log n}{\log\log n})$ 時間 [10] での構築手法が提案されている．オンライン構築とは，変換対象となる文字列が 1 文字ずつ入力され，入力されるたびに現時点までに入力された文字列の情報をもとに文字列を変換していき，変換後の文字列を構築していく手法となる．

パラメタ化照合問題 [1] は，プログラムの剽窃検索や，DNA や RNA などの構造解析などの，文字列の厳密一致ではなく構造に着目し検索を行う問題である．

BW 変換には多くの変種が存在しており，パラメタ化 BW 変換 [5, 7] も BW 変換の変種の一種である．パラメタ化 BW 変換は BW 変換を用いた検索手法である FM インデックスと同様の手法でのパラメタ化照合問題の解答を可能とした文字列変換手法である．

Ganguly ら [5] によって提案されたパラメタ化 BW 変換は，パラメタ化接尾辞木を用い補助データ構造を構築し，検索を行うものである．その後，Kim ら [7] によって提案されたパラメタ化 BW 変換の構築手法では，直前符号化の定義を変更することで，検索をより簡潔に行えるようになった．長さ n の文字列 T のパラメタ化 BW 変換は，アルファベットサイズを σ とするときオフラインで $O(n \log \sigma)$ 時間での構築手法が提案されている．

本研究では，パラメタ化 BW 変換のオンライン構築を目的とする．

我々は，Kim ら [7] の補題で述べられている性質と，新たに開発した種類数符号化と $LCP_{\infty}(T)$ 配列を導入することでパラメタ化 BW 変換のオンライン構築を変数文字のアルファベットサイズを $|\Pi|$ とすると $O(n|\Pi|\frac{\log n}{\log \log n})$ 時間で行う初めての手法を提案した．本手法は，1 文字の入力に対して，$O(|\Pi|\frac{\log n}{\log \log n})$ 時間でパラメタ化 BW 変換を更新できるが，$O(\frac{\log n}{\log \log n})$ 時間は，使用するデータ構造 [8] によるものであり，1 文字の入力に対して更新箇所が最大 $|\Pi|$ 箇所生じることから，$O(|\Pi|\frac{\log n}{\log \log n})$ 時間は，現状ほぼ最適な更新であると考えられる．

1.2 本論文の構成

第 2 章では，基本的な記法や，BW 変換，パラメタ化 BW 変換について確認する．第 3 章では，パラメタ化 BW 変換のオンライン構築の手法について述べる．最後に，第 4 章で本論文の成果についてまとめる．

第2章

準備

2.1 表記と基本事項

本論文では，2つのアルファベットをΣとΠとする．Σの要素を**定数文字**，Πの要素を**変数文字**とする．自然数の集合を$\mathbb{N}$とする．長さnの文字列Tのi文字目を$T[i]$ $(1 \leq i \leq n)$で表し，i文字目からj文字目までの部分文字列を$T[i,j]$で表す．$i < j$のとき，$T[j,i] = \varepsilon$とし，空文字とする．また，文字列の長さを$|T|$で表す．文字列Tに出現する任意の文字cについてTでの最右出現する位置を$\mathcal{R}(T,c)$で表し，最左出現する位置を$\mathcal{L}(T,c)$で表す．文字列T中に出現する変数文字の種類数を$\pi(T)$で表す．T中の位置iの巡回右側は$i+1, i+2, ..., n, 1, 2, ..., i-1, i$のように，位置$i+1$から文末の位置$n$まで辿った後に先頭の1文字目に戻り，その後位置$i$までと定義する．また，2つの文字列$T, S$の結合を$T \circ S$で表す．$\Sigma$上の文字には順序関係が存在するものとし，2つの文字列$T, S \in \Sigma^*$において$T[1, i-1]$と$S[1, i-1]$が等しく，$T[i]$が$S[i]$よりもΣ上の順序関係が小さいとき，T, Sの**辞書式順序**は，$T \prec S$と定義し，$|T| < |S|$で$T[1, |T|] = S[1, |T|]$のときの辞書式順序は，$T \prec S$と定義する．また，$\$ \in \Sigma$は$\Sigma$内で辞書式順序が最小の文字とし，$T$中では文末のみに出現するものとする．

文字列Tの**巡回**を以下のように定義する．

定義 1. 長さnの文字列Tの，i回目 $(0 \leq i < n)$の**巡回**はT_iで表記され，以下の式で定義される．

$$T_i = T[i+1, n] \circ T[1, i]$$

文字列Tの巡回行列M_Tを以下のように定義する．

定義 2. 長さ n の文字列 T の**巡回行列** M_T は，$n \times n$ の行列であり，各行は文字列 T の相異なる巡回になっており，M_T は行方向にソートされている．つまり，巡回行列 M_T の i 行を $M_T[i]$ で表記するとき，M_T の i 行と j 行 $(i < j)$ の辞書式順序は，$M_T[i] \prec M_T[j]$ である．

文字列 $T \in (\Sigma \cup \Pi)^*$ を**パラメタ化文字列**といい，長さ n の 2 つの文字列 $T, S \in (\Sigma \cup \Pi)^*$ の全ての $1 \leq i \leq n$ に対して，$\alpha \in \Sigma$ の場合，$f(\alpha) = \alpha$ を満たし，$f(T[i]) = S[i]$ を満たす全単射の写像 $f : (\Sigma \cup \Pi) \to (\Sigma \cup \Pi)$ が存在するとき，文字列 T, S はパラメタ化一致といい，$T \sim S$ で表す．

このパラメタ化一致の判別を行う手法の 1 つに**直前符号化** [1] という符号化がある．

定義 3 (直前符号化)**.** 長さ n の文字列 $T \in (\Sigma \cup \Pi)^*$ に対して，直前符号化 $\langle T \rangle$ は以下のように定義される．

$$\langle T \rangle[i] = \begin{cases} T[i] & \text{if } T[i] \in \Sigma \\ \infty & \text{if } T[i] \in \Pi \text{ and } i = \mathcal{L}(T, T[i]) \\ i - k & \text{if } T[i] \in \Pi \text{ and } k = \mathcal{R}(T[1, i-1], T[i]) \end{cases}$$

また，$\Sigma \cup \mathbb{N} \cup \{\infty\}$ 上での順序は，$\alpha \in \Sigma, \beta \in \mathbb{N}$ のとき，$\alpha \prec \beta \prec \infty$ とし，$\mathbb{N}$ 内の辞書式順序は自然数の大小に従うものとする．

直前符号化には，次に示す補題が成立する．

補題 1 (直前符号化)**.** 長さ n の 2 つの文字列 $T, S \in (\Sigma \cup \Pi)^*$ に対して，$\langle T \rangle = \langle S \rangle$ のときに限り $T \sim S$ である．

例 1. $\Sigma = \{\texttt{\$,a}\}, \Pi = \{\texttt{X,Y,Z}\}$ のとき，$T = \texttt{aXaYXXZYa\$}$ の直前符号化は $\langle T \rangle = \texttt{a}\infty\texttt{a}\infty\texttt{31}\infty\texttt{4a\$}$ となる．

配列 C_T は，T に出現する文字 c について $C_T[c]$ に c よりも辞書式順序が小さい定数文字の T 中の出現回数を格納した配列である．

例 2. $\Sigma = \{\texttt{\$,a,b,c}\}, \Pi = \{\texttt{X,Y}\}, \texttt{a} \prec \texttt{b} \prec \texttt{c}$ のとき，$T = \texttt{aXbYaca\$}$ の配列 C_T は表 2.1 である．

表 2.1: $T =$ aXbYaca$ に対する配列 C_T

$	a	b	c
0	1	4	5

表 2.2: $T =$ abraca$ を BW 変換したときの様子

行番号	巡回文字列	ソート後	L
1	abraca$	$abraca	a
2	braca$a	a$abrac	c
3	raca$ab	abraca$	$
4	aca$abr	aca$abr	r
5	ca$abra	braca$a	a
6	a$abrac	ca$abra	a
7	$abraca	raca$ab	b

2.2 BW 変換

BW 変換は，Burrows と Wheeler [2] によって提案された文字列の可逆変換手法の一種である．BW 変換の定義を以下に示す．

定義 4 (BW 変換)**.** BW 変換は，長さ n の文字列 T に対して，以下の操作を行い文字列 L を生成する．

1. 文字列 T の $n \times n$ の巡回行列 M_T を生成する．
2. 巡回行列 M_T の最終列を文字列 L として保存する．つまり，位置 i $(1 \leq i \leq n)$ について $L[i] = M_T[i][n]$ である．

例 3. $\Sigma =$ {$,a,b,c}, $T =$ abraca$ のときの BW 変換の様子は表 2.2 である．

BW 変換を復元する際に用いる文字列 F の定義を以下に示す．

定義 5. 文字列 F は，巡回行列 M_T の先頭列とする．つまり，位置 i $(1 \leq i \leq n)$ について $F[i] = M_T[i][1]$ である．

BW 変換によって生成された文字列 L と，文字列 F の間には 2 つの性質が成立する．

まず 1 つ目の性質が，$L[j] =$ $ のとき，j 以外のすべての i で $L[i] \circ F[i]$ が文字列 T の部分文字列になっている性質である．これは，F が T の巡回行列 M_T の先頭列と一致す

ることと，巡回行列 M_T の各行が文字列 T の巡回 $T_i = T[i+1, n] \circ T[1, i]$ で構成されているためである．

2 つ目の性質が，M_T の任意の 2 行 i, j $(i < j)$ について，$M_T[i][n] = M_T[j][n]$ を満たすとき，$M_T[i'] = M_T[i][n] \circ M_T[i][1, n-1]$，$M_T[j'] = M_T[j][n] \circ M_T[j][1, n-1]$ を満たす行 i', j' について，$i' < j'$ である性質である．巡回行列 M_T において，文末が同じ文字である 2 行 $M_T[i], M_T[j]$ $(i < j)$ を用いて考える．つまり，$M_T[i][n] = M_T[j][n] = \alpha$ である．2 つの文字列 $M_T[i], M_T[j]$ の辞書順は，末尾が同じ文字であることから末尾以外の部分で決まっていることとなるため，$M_T[i][1, n-1] \prec M_T[j][1, n-1]$ である．このとき，$M_T[i], M_T[j]$ の文末文字を文頭に巡回した文字列を $M_T[i'], M_T[j']$ について考える．つまり，$M_T[i'] = \alpha \circ M_T[i][1, n-1], M_T[j'] = \alpha \circ M_T[j][1, n-1]$ である．$M_T[i'], M_T[j']$ の辞書順は，$M_T[i][1, n-1] \prec M_T[j][1, n-1]$ であることから $M_T[i'] \prec M_T[j']$ であり，巡回行列 M_T は行方向にソートされているので，$i' < j'$ が成立する．したがって，M_T において，文末が同じ行の辞書式順序は，文末以外の部分で決定しているため，文頭に巡回した文字列でも，辞書式順序による位置関係は変化しない．

つまり，文字 α 同士の L 上での出現順序と F 上での出現順序が変化していないということである．

上記 2 つの性質と，$ が T の文末に一度しか出現しないことから文字列 T を復元することができる．

まず，$L[i] = \$$ とすると，$M_T[i] = T$ である，なぜならば，$ は T の文末に一度しか出現しない文字であり，文字列 L は巡回行列 M_T の最終列を保存した文字列なので，$L[i] = M_T[i][n]$ であり，$L[i] = M_T[i][n] = \$$ である．したがって，$M_T[i]$ は末尾が $ である文字列となるため，$M_T[i]$ は T 自身である．またこのことから，$F[i] = T[1]$ であることも言える．図 2.1 の例では，$L[3] = \$$ であることから $T[1] = F[3] = \mathtt{a}$ であることがわかる．

一文字目が判明した後は，文字 α に対して，F 上での出現順と L 上での出現順が変わらないという 2 つ目の性質から，$F[i]$ の L 上での出現位置を求める．図 2.1 の例では，$F[3] = \mathtt{a}$ は F 上で 2 番目の a であるため，L 上で 2 番目に出現する a が T 中で同じ位置の文字である．したがって $F[3] = L[5]$ であることがわかる．

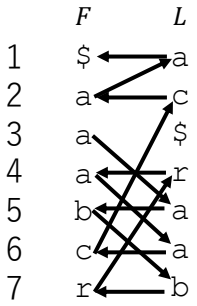

図 2.1: 文字列 $T =$ abraca$ を文字列 L, F を用いた復元の様子

その後，$L[i] =$ \$ のとき，i 以外のすべての j で $L[j] \circ F[j]$ が文字列 T の部分文字列になっているという性質から，次の文字が判明する．図 2.1 の例では，先ほど $L[5] =$ a が $T[1]$ であることがわかったので，$L[5] \circ F[5] =$ ab が部分文字列となることから，$F[5] = T[2]$ となることがわかる．

以上の操作を繰り返すことで，文字列 T を復元することができる．

2.3 FM インデックス

FM **インデックス**とは，Ferragina ら [4] によって提案された BW 変換を用いた文字列検索のためのアルゴリズムである．FM インデックスでは，文字列 T の BW 変換後の文字列 L，パターン P，配列 C_T を入力として，文字列 T の巡回行列 M_T において P が接頭辞となる行の区間 (i, j) を出力する．M_T では，各行が T の各位置から始まる巡回文字列となっており，各行の先頭から \$ までの部分文字列は T の接尾辞となっており，M_T が辞書順でソートされていることから，T 中に P が複数回出現している場合は，複数行の接頭辞としてまとまって出現することとなる．よって，区間 (i, j) を出力することで，T 中での P の出現を検索することができる．

FM インデックスの具体的な検索について記す．FM インデックスでは，パターンの後方から M_T での出現を検索する．これは，BW 変換後の文字列 L と文字列 F の間の性質で述べた，$L[j] =$ \$ のとき，j 以外のすべての i で $L[i] \circ F[i]$ が文字列 T の部分文字列になっている性質が理由である．P で既に検索を終えている部分文字列を $P[k, |P|]$，その結果得られた区間を (i, j) としたとき，M_T の行 l $(i \leq l \leq j)$ の接頭辞が $P[k, |P|]$ となっている．このとき，次に検索する文字は $P[k-1]$ であり，今回検索するのは，部分文字列 $L[i, j]$ である．

これは，M_T の各行が T の巡回で構成されていることから $L[i,j]$ 中に出現した $P[k-1]$ が M_T の行 l $(i \leq l \leq j)$ の接頭辞 $P[k,|P|]$ と部分文字列 $P[k-1] \circ P[k,|P|] = P[k-1,|P|]$ を生成するためである．$L[i,j]$ 以外に出現している $P[k-1]$ は，出現している行を i' $(i' < i$ または $j < i')$ とすると，$M_T[i']$ の接頭辞が $P[k,|P|]$ ではないため，検索の対象外となる．$L[i,j]$ 中に出現した $P[k-1]$ を検索した後は，2 つ目の性質である，任意の文字 α に対して，F 上での出現順順序と L 上での出現順序が変化しない性質を用いることで，M_T の接頭辞として $P[k-1,|P|]$ が出現している区間を新たに更新する．

FM インデックスを Algorithm 1 に記す．なお， $rank$ 関数 $rank_S(\alpha, p)$ は，$S[1,p]$ 中に出現した文字 α の出現回数を返す関数とする．

Algorithm 1: FM インデックス

input : 文字列 T の BW 変換後の文字列 L，パターン P，配列 C_T
output: 出現区間 (i,j)

```
1  k = |P|, c = P[k];
2  i = C_T[c] + 1;
3  j = C_T[c] + rank_L(c, |L|);
4  while i ≤ j and k ≥ 2 do
5  |   c = P[k − 1];
6  |   i = C_T[c] + rank_L(c, i − 1) + 1;
7  |   j = C_T[c] + rank_L(c, j);
8  |   k = k − 1;
9  if i ≤ j then
10 |   return (i, j);
11 else
12 |   return not found;
```

2.4 種類数符号化

パラメタ化BW変換の定義の準備として，**種類数符号化**を導入する．文字列 $T \in (\Sigma \cup \Pi)^*$ に対する種類数符号化 $[\![T]\!]$ は以下のように定義される．

定義 6 (種類数符号化)**.** 長さ n の文字列 $T \in (\Sigma \cup \Pi)^*$ の種類数符号化 $[\![T]\!][i]$ は，$T[i]$ から巡回右側に確認し，$T[i]$ と同じ変数文字が出現するまでに出現した変数文字の種類数

であり，次のように定義される．

$$[\![T]\!][i] = \begin{cases} T[i] & \text{if } T[i] \in \Sigma \\ \pi(T_i[1, \mathcal{L}(T_i, T[i])]) & \text{otherwise} \end{cases}$$

例 4. $\Sigma = \{\texttt{\$,a}\}, \Pi = \{\texttt{X,Y,Z}\}$ のとき，$T = \texttt{aXaYXXZYa\$}$ の種類数符号化は $[\![T]\!] = \texttt{a2a31332a\$}$ となる．

種類数符号化と直前符号に関して，以下の定理を示す．

定理 1. 2 つの文字列 S, T について，以下が成立する．

$$[\![S]\!] = [\![T]\!] \Leftrightarrow S \sim T$$

証明. まず，$[\![S]\!] = [\![T]\!] \Rightarrow S \sim T$ を示す．したがって，仮定より $[\![S]\!] = [\![T]\!]$ なので，位置 i $(1 \leq i)$ について $[\![S]\!][i] = [\![T]\!][i]$ である．まず，$S[i], T[i] \in \Sigma$ のときを考える．定義 3 より，$\langle S \rangle[i] = S[i]$，$\langle T \rangle[i] = T[i]$ であり，$[\![S]\!][i] = S[i] = T[i] = [\![T]\!][i]$ なので $\langle S \rangle[i] = \langle T \rangle[i]$ である．次に，$S[i], T[i] \in \Pi$ のときを考える．定義 6 より，$[\![S]\!][i] = \pi(S_i[1, \mathcal{L}(S_i, S[i])])$，$[\![T]\!][i] = \pi(T_i[1, \mathcal{L}(T_i, T[i])])$ である．$[\![S]\!][i] = [\![T]\!][i]$ なので，$\pi(S_i[1, \mathcal{L}(S_i, S[i])] = \pi(T_i[1, \mathcal{L}(T_i, T[i])])$ であるので，$\mathcal{L}(S_i, S[i]) = \mathcal{L}(T_i, T[i])$ である．したがって，$\langle S[i + \mathcal{L}(S_i, S[i])] \rangle = \mathcal{L}(S_i, S[i])$，$\langle T[i + \mathcal{L}(T_i, T[i])] \rangle = \mathcal{L}(T_i, T[i])$ となるので，$\langle S \rangle[i + \mathcal{L}(S_i, S[i])] = \langle T \rangle[i + \mathcal{L}(T_i, T[i])]$ である．したがって，$[\![S]\!] = [\![T]\!] \Rightarrow S \sim T$ である．

次に，$[\![S]\!] = [\![T]\!] \Leftarrow S \sim T$ を対偶を用いて示す．$[\![S]\!] \neq [\![T]\!]$ と仮定する．特に，$[\![S]\!][i] \neq [\![T]\!][i]$ と仮定する．まず，$S[i], T[i] \in \Sigma$ のときを考える．定義 6 より $[\![S]\!][i] = S[i]$，$[\![T]\!][i] = T[i]$ なので，$[\![S]\!][i] \neq [\![T]\!][i]$ から $S[i] \neq T[i]$ である．定義 3 より，$\langle S \rangle[i] = S[i]$，$\langle T \rangle[i] = T[i]$ である．したがって，仮定より $S[i] \neq T[i]$ なので，$\langle S \rangle[i] \neq \langle T \rangle[i]$ である．次に，$S[i] \in \Sigma$ かつ $T[i] \in \Pi$ のときを考える．定義 3 より，$\langle S \rangle[i] = S[i]$，$\langle T \rangle[i] = \infty$ または $\langle T \rangle[i] = i - k$ である．ただし，$k = \mathcal{R}(T[1, i-1], T[i])$ である．したがって，$\langle S \rangle[i] \neq \langle T \rangle[i]$ である．また，$T[i] \in \Sigma$ かつ $S[i] \in \Pi$ のときも同様である．最後に，$S[i], T[i] \in \Pi$ のときを考える．定義 6 より $[\![S]\!][i] = \pi(S_i[1, \mathcal{L}(S_i, S[i])])$，$[\![T]\!][i] = \pi(T_i[1, \mathcal{L}(T_i, T[i])])$ であ

る．$[\![S]\!][i] \neq [\![T]\!][i]$ と仮定しているので，$\pi(S_i[1, \mathcal{L}(S_i, S[i])]) \neq \pi(T_i[1, \mathcal{L}(T_i, T[i])])$ である．$\pi(S_i[1, \mathcal{L}(S_i, S[i])]) \neq \pi(T_i[1, \mathcal{L}(T_i, T[i])])$ を満たす状況は，$\mathcal{L}(S_i, S[i]) \neq \mathcal{L}(T_i, T[i])$ のときと，$\mathcal{L}(S_i, S[i]) = \mathcal{L}(T_i, T[i])$ のときで場合分けして考える．まず，$\mathcal{L}(S_i, S[i]) \neq \mathcal{L}(T_i, T[i])$ のときを考える．文字列 S の巡回文字列 S_i での文字 $S[i]$ の最左出現位置を文字列 S での位置で表記すると，$\mathcal{L}(S_i, S[i])+i$ である．なので，文字列 S の位置 $\mathcal{L}(S_i, S[i])+i$ と文字列 T の位置 $\mathcal{L}(T_i, T[i])+i$ の直前符号化は，$\langle S\rangle[\mathcal{L}(S_i, S[i])+i] = \mathcal{L}(S_i, S[i])$，$\langle T\rangle[\mathcal{L}(T_i, T[i])+i] = \mathcal{L}(T_i, T[i])$ である．よって，$\mathcal{L}(S_i, S[i]) \neq \mathcal{L}(T_i, T[i])$ であることから $\langle S\rangle[\mathcal{L}(S_i, S[i])+i] \neq \langle T\rangle[\mathcal{L}(T_i, T[i])+i]$ となるので，$\langle S\rangle \neq \langle T\rangle$ である．次に，$\mathcal{L}(S_i, S[i]) = \mathcal{L}(T_i, T[i])$ のときを考える．仮定より，$\pi(S_i[1, \mathcal{L}(S_i, S[i])]) \neq \pi(T_i[1, \mathcal{L}(T_i, T[i])])$ なので，$\pi(S[i+1, i+\mathcal{L}(S_i, S[i])]) \neq \pi(T[i+1, i+\mathcal{L}(T_i, T[i])])$ である．今回は，$\mathcal{L}(S_i, S[i]) = \mathcal{L}(T_i, T[i])$ であることを考えている．したがって，$S[i+1, i+\mathcal{L}(S_i, S[i])]$ と $T[i+1, i+\mathcal{L}(T_i, T[i])]$ は同じ長さの文字列である．よって，同じ長さの文字列で文字列中に含まれる変数文字の種類数が異なるため，$\langle S[i+1, i+\mathcal{L}(S_i, S[i])]\rangle \neq \langle T[i+1, i+\mathcal{L}(T_i, T[i])]\rangle$ である．したがって，$[\![S]\!] = [\![T]\!] \Leftarrow S \sim T$ である．

よって，$[\![S]\!] = [\![T]\!] \Leftrightarrow S \sim T$ である． □

2.5 パラメタ化 BW 変換

パラメタ化 BW 変換は，Ganguly ら [5] によって提案された文字列変換手法の一種であり，パラメタ化照合問題を FM インデックスと同様の手法で解くことを目的として開発された．パラメタ化 BW 変換は，２つの文字列 $F(T), pBWT(T)$ から成る．パラメタ化 BW 変換は以下のように定義される．

定義 7 (パラメタ化 BW 変換)**.** 最初に，長さ n の文字列 T を巡回させ直前符号化しソートした行列 $rot(T)$ を生成する．$rot(T)$ の各行 i について，$rot(T)[i] = \langle T_k\rangle$ を用いて，行列 $\overline{rot}(T)$ を以下のように定義する．

$$\overline{rot}(T) = T_k$$

なお，任意の行 i について $\langle\overline{rot}(T)[i]\rangle = rot(T)[i] = \langle T_k\rangle$ が成立する．

表 2.3: 文字列 $T =$ XYZZaYYZ$ のパラメタ化 BW 変換

	巡回	直前符号化		$rot(T)$	$\overline{rot}(T)$	$F(T)$	$[\![\overline{rot}(T)]\!]$	$pBWT(T)$
1	XYZZaYYZ$	∞∞∞1a414$		$∞∞∞1a414	$XYZZaYYZ	$	$3212a133	3
2	YZZaYYZ$X	∞∞1a414$∞		a∞1∞$∞441	aYYZ$XYZZ	a	a133$3212	2
3	ZZaYYZ$XY	∞1a∞14$∞4		∞$∞∞41a41	Z$XYZZaYY	3	3$3212a13	3
4	ZaYYZ$XYZ	∞a∞14$∞44		∞a∞14$∞44	ZaYYZ$XYZ	2	2a133$321	1
5	aYYZ$XYZZ	a∞1∞$∞441	→	∞1a∞14$∞4	ZZaYYZ$XY	1	12a133$32	2
6	YYZ$XYZZa	∞1∞$∞441a	整列	∞1∞$∞441a	YYZ$XYZZa	1	133$3212a	a
7	YZ$XYZZaY	∞∞$∞441a4		∞∞$∞441a4	YZ$XYZZaY	3	33$3212a1	1
8	Z$XYZZaYY	∞$∞∞41a41		∞∞1a414$∞	YZZaYYZ$X	2	212a133$3	3
9	$XYZZaYYZ	$∞∞∞1a414		∞∞∞1a414$	XYZZaYYZ$	3	3212a133$	$

このとき，$pBWT(T)[i]$ は以下のように定義する．

$$pBWT(T)[i] = [\![\overline{rot}(T)[i]]\!][n]$$

また，$F(T)[i]$ を以下のように定義する．

$$F(T)[i] = [\![\overline{rot}(T)[i]]\!][1]$$

パラメタ化BW変換で得られた$pBWT(T), F(T)$の間にも，文字α同士において，$pBWT(T)$上での出現順序と，$F(T)$ 上での出現順序が変化しない性質が成り立つことが以下のようにして示される．

文字 $\alpha \in \mathbb{N}$ について考える．$pBWT(T)[i] = pBWT(T)[j] = \alpha$ $(i < j)$ とする．定義 6 と定義 7 より，$pBWT(T)[i] = [\![\overline{rot}(T)[i]]\!][n] = \pi(\overline{rot}(T)[i, k_i])$，$pBWT(T)[j] = [\![\overline{rot}(T)[j]]\!][n] = \pi(\overline{rot}(T)[j, k_j])$ である．ただし，$k_i = \mathcal{L}(\overline{rot}(T)[i], \overline{rot}(T)[i][n])$，$k_j = \mathcal{L}(\overline{rot}(T)[j], \overline{rot}(T)[j][n])$ とする．$\overline{rot}(T)[i], \overline{rot}(T)[j]$ の文末の文字を先頭に巡回した文字列を $\overline{rot}(T)[i'], \overline{rot}(T)[j']$ とする．つまり，以下の関係式が成立する．

$$\overline{rot}(T)[i'] = \overline{rot}(T)[i][n] \circ \overline{rot}(T)[i][1, n-1]$$

$$\overline{rot}(T)[j'] = \overline{rot}(T)[j][n] \circ \overline{rot}(T)[j][1, n-1]$$

	rot(T)		$\overline{\text{rot}}(T)$
3	∞11∞2$3∞2	3	YYYXY$XZX
6	∞∞1142$3∞	6	XYYYXY$XZ

図 2.2: 文字列 $T =$ XZXYYYXY$ において，$rot(T)[3], \overline{rot}(T)[3]$ と，$rot(T)[6], \overline{rot}(T)[6]$. $\overline{rot}(T)[3]$ の文末文字 X が，$\overline{rot}(T)[3]$ で最左に出現している位置 k_3 は，$k_3 = 4$ である．このとき，$rot(T)[6] = \infty \circ rot(T)[3][1, k_3 - 1] \circ k_3 \circ rot(T)[3][k_3 + 1, 8] = \infty \circ rot(T)[3][1, 3] \circ 4 \circ rot(T)[3][5, 8]$ となっていることがわかる．

また，$rot(T)[i'], rot(T)[j']$ に関して以下の関係式が成立する．

$$rot(T)[i'] = \infty \circ rot(T)[i][1, k_i - 1] \circ k_i \circ rot(T)[i][k_i + 1, n - 1]$$

$$rot(T)[j'] = \infty \circ rot(T)[j][1, k_j - 1] \circ k_j \circ rot(T)[j][k_j + 1, n - 1]$$

上記のことを踏まえ，$\overline{rot}(T)[i'], \overline{rot}(T)[j']$ について，$i' < j'$ となることを示す．

$i' < j'$ となることを示すうえで，k_i, k_j の関係によって場合分けをして考える．まず，$k_i < k_j$ のときを考える．$rot(T)[i], rot(T)[j]$ の辞書順が $rot(T)[i][1, k_i - 1]$ と $rot(T)[j][1, k_i - 1]$ の比較で決定していた場合について考える．$i < j$ であることから $rot(T)$ の定義より，$rot(T)[i] \prec rot(T)[j]$ であるので，$rot(T)[i][1, k_i - 1] \prec rot(T)[j][1, k_j - 1]$ であり，$\infty \circ rot(T)[i][1, k_i - 1] \prec \infty \circ rot(T)[j][1, k_i - 1]$ が成立するので，$rot(T)[i'] \prec rot(T)[j']$ である．したがって，$rot(T)$ は行方向にソートされているので，$i' < j'$ となる．続いて，$rot(T)[i][1, k_i - 1]$ と $rot(T)[j][1, k_i - 1]$ の比較で決定しない場合，つまり，$rot(T)[i][1, k_i - 1] = rot(T)[j][1, k_i - 1]$ であるときについて考える．まず，$rot(T)$ が行方向にソートされていることと，$rot(T)[i][k_i] = \infty$ であることから，$rot(T)[i]$ と $rot(T)[j]$ は，k_i 文字目で辞書順が決定していることはないため，$rot(T)[j][k_i] = \infty$ である．また，$k_i = \mathcal{L}(\overline{rot}(T)[i], \overline{rot}(T)[i][n])$ なので，$rot(T)[i']$ では $rot(T)[i'][k_i + 1] = k_i$ となる．一方，$k_i < k_j$ であることから，$rot(T)[j'][k_i + 1] = rot(T)[j][k_i] = \infty$ である．つまり，$rot(T)[i'][k_i + 1] = k_i, rot(T)[j'][k_i + 1] = \infty$ となるので，$rot(T)[i'][1, k_i + 1] \prec rot(T)[j'][1, k_i + 1]$ であり，$i' < j'$ となる．

次に，$k_i = k_j$ のときについて考える．$rot(T)[i], rot(T)[j]$ の辞書順が $rot(T)[i][1, k_i - 1]$

と $rot(T)[j][1,k_j-1]$ の比較で決定していた場合は，$k_i < k_j$ のときと同様に $rot(T)[i'] \prec rot(T)[j']$ が成立し $i' < j'$ となる．$rot(T)[i][1,k_i-1]$ と $rot(T)[j][1,k_j-1]$ の比較で決定していない場合は，$rot(T)[i'][k_i+1] = k_i, rot(T)[j'][k_j+1] = k_j$ となるが，$k_i = k_j$ なので $\infty \circ rot(T)[i][1,k_i-1] \circ k_i = \infty \circ rot(T)[j][1,k_j-1] \circ k_j$ である．よって，$rot(T)[i'][1,k_i+1] = rot(T)[j'][1,k_j+1]$ である．したがって，$rot(T)[i'], rot(T)[j']$ の辞書順は，k_i+2 文字目以降で決定するが k_i+2 文字目以降は，$rot(T)[i], rot(T)[j]$ を用いて以下のように表記できる．

$$rot(T)[i'][k_i+2,n] = rot(T)[i][k_i+1,n-1]$$
$$rot(T)[j'][k_j+2,n] = rot(T)[j][k_j+1,n-1]$$

したがって，$rot(T)[i'], rot(T)[j']$ の辞書順は，$rot(T)[i][k_i+1,n-1]$ と $rot(T)[j][k_j+1,n-1]$ の辞書順によって決定する．一方，$rot(T)[i], rot(T)[j]$ について考えると，$rot(T)[i][1,k_i-1] = rot(T)[j][1,k_j-1]$ かつ $rot(T)[i][k_i] = \infty$，$rot(T)[j][k_j] = \infty$，$k_i = k_j$ であることから，$rot(T)[i][k_i+1,n] \prec rot(T)[j][k_j+1,n]$ であることがわかる．したがって，$rot(T)[i'] \prec rot(T)[j']$ となるので，$i' < j'$ が成立する．

最後に，$k_i > k_j$ のときについて考える．$rot(T)[i], rot(T)[j]$ の辞書順が $rot(T)[i][1,k_j-1]$ と $rot(T)[j][1,k_j-1]$ の比較で決定していた場合は今までと同様に $rot(T)[i'] \prec rot(T)[j']$ が成立し $i' < j'$ となる．$rot(T)[i][1,k_j-1]$ と $rot(T)[j][1,k_j-1]$ の比較で決定していない場合は，$rot(T)[i][1,k_j-1] = rot(T)[j][1,k_j-1]$ である．このとき，$\pi(\overline{rot}(T)[j][1,k_j]) = \alpha$ であり，$\pi(\overline{rot}(T)[i][1,k_i]) = \alpha$ である．$rot(T)[i][1,k_j] = rot(T)[j][1,k_j]$ であるので，$\pi(\overline{rot}(T)[i][1,k_j]) = \alpha$ である．よって，$\pi(\overline{rot}(T)[i][1,k_j]) = \pi(\overline{rot}(T)[i][1,k_i]) = \alpha$ という結果が得られた．これは，$\overline{rot}(T)[i][k_j+1,k_i]$ 中に $\overline{rot}(T)[i][1,k_j]$ で出現していない変数文字が出現していないことを意味するが，$rot(T)[i][k_i] = \infty$ であることから，$\overline{rot}(T)[i][k_j+1,k_i]$ 中に $\overline{rot}(T)[i][1,k_j]$ で出現していない変数文字が出現していることとなり，矛盾が生じる．したがって，$k_i > k_j$ のときは，$rot(T)[i][1,k_j-1]$ と $rot(T)[j][1,k_j-1]$ の比較で辞書順が決定していない場合はない．

したがって，パラメタ化 BW 変換で得られた $pBWT(T), F(T)$ の間にも，文字 α 同士において，$pBWT(T)$ 上での出現順序と，$F(T)$ 上での出現順序が変化しない性質が成り

立つ．

また，パラメタ化 BW 変換での検索手法は，BW 変換での FM インデックスを基本軸としており，文字列 $\alpha \circ P$ が $\overline{rot}(T)$ において接頭辞とパラメタ化一致する行の区間 (i', j') を，$\overline{rot}(T)$ において接頭辞が P とパラメタ化一致する行の区間 (i, j)，文字 α，文字列 $pBWT(T), F(T)$ を入力として求める．違う点としてあげられるのは，変数文字の検索である．既に検索された文字列を P，次に検索する文字を α，既に検索した結果として得られた区間を (i, j) としたとき，検索は次の 2 つの場合に分けられる．まず，α が定数文字である，または，α が変数文字であり，かつ P 中に α が出現している場合について考える．α が定数である場合は，FM インデックスと同様に $pBWT(T)[i, j]$ に出現する α が今回探す α である．α が変数文字で既に検索された文字列 P 中に出現している場合は，P 中での α の最左出現位置を l とすると，$P[1, l]$ に出現している変数文字の種類数 $\pi(P[1, l])$ が $pBWT(T)$ 上での α の符号化となる．したがって，$pBWT(T)[i, j]$ に出現する $\pi(P[1, l])$ を検索することで，$\alpha \circ P$ の検索ができる．

次に，α が変数文字であり，かつ P 中に α が出現していない場合について考える．このとき，$\pi(P)$ より大きい自然数を検索する．なぜなら，P 中に α が出現しているときの α の符号化で考えられる最大値は，P 中に出現する変数文字の種類数である．したがって，今回は，α が P 中に出現していないため，$\pi(P)$ よりも大きい自然数が対象となり，$pBWT(T)[i, j]$ に出現する $\pi(P)$ より大きいの自然数を検索することとなる．

以上を踏まえた検索アルゴリズムが，Algorithm 2 である．なお，$select$ 関数は文字列 T と文字 α，自然数 p を受け取り，文字列 T において p 回目の出現である文字 α の位置を返す関数とし，$rankrange$ 関数は文字列 T，区間 $(i, j), (x, y)$ を受け取り，文字列 T において $T[i, j]$ 中に出現する x 以上 y 以下の文字の出現回数を返す関数とする．RMQ は配列 H と区間 (i, j) を受け取り，$H[i, j]$ 中の最大値を返す関数である．配列 pLF は，i 番目の要素に次の条件を満たす位置 j を格納する配列である．

$$rot(T)[i] = \langle T_l \rangle \text{ のとき}, rot(T)[j] = \langle T_{l-1} \rangle$$

つまり，$\overline{rot}(T)[i][n] = T_l[n] = T[l]$，$\overline{rot}(T)[j][1] = T_{l-1}[1] = T[l]$ としたとき $pLF[i] = j$ となる．ただし，$l = 0$ のときは，$\overline{rot}(T)[j][1] = T_{n-1}[1] = T[n] = \$$ を満たす位置 j を格

納する.

Algorithm 2: パラメタ化 BW 変換を用いた FM インデックス

input : 文字列 $F(T), pBWT(T)$，パターン P，パターン長 m，配列 pLF
output: 出現区間 (i, j)

```
1  c = P[m], k = m;
2  while i ≤ j and k ≥ 1 do
3      c = P[k];
4      if c ∈ Σ or c appears in P[k + 1, m] then
5          if c ∈ Π then
6              t = 1;
7              for l = k to m do
8                  if P[l] = c then
9                      break;
10                 else if P[l] ∈ Π and P[l] dose not appear in P[k, l − 1] then
11                     t = t + 1;
12             c = t;
13         i = select_{F(T)}(rank_{pBWT(T)}(i − 1, c) + 1, c);
14         j = select_{F(T)}(rank_{pBWT(T)}(j, c), c);
15     else
16         t = 1;
17         for l = k to m do
18             if P[l] ∈ Π and P[l] dose not appear in P[k, l − 1] then
19                 t = t + 1;
20         j' = RMQ(pLF, (i, j));
21         i = j' − rankrange_{pBWT(T)}((i, j), (t, |Π|)) + 1;
22         j = j';
23     k = k − 1;
24 if i ≤ j then
25     return (i, j);
26 else
27     return not found;
```

第3章

提案手法

3.1 はじめに

本章では，提案手法であるパラメタ化BW変換のオンライン構築について記す．すなわち，文字列Tのパラメタ化BW変換$pBWT(T)$とそれに付随する文字列$F(T)$が与えられているとし，その後Tの先頭に文字αを追加して新たな文字列$T' = \alpha \circ T$が得られたとする．このとき，T'のパラメタ化BW変換$pBWT(T')$と$F(T')$の構築を，$pBWT(T), F(T)$を必要最小限更新することで行う．

提案アルゴリズムでは，Navarroら [8] が提案したデータ構造を$pBWT(T), F(T)$の保存に用いる．本データ構造は，文字列Sに対する次に示す関数や操作を$O(\frac{\log n}{\log \log n})$時間で行える．

定義 8. 1. $rank_S(\alpha, p)$：$S[1, p]$中に出現した文字αの出現回数を返す関数

2. $select_S(\alpha, p)$：文字列Sにおける，文字αのp回目の出現位置を返す関数

3. $insert_S(\alpha, p)$：文字列Sのp文字目に文字αを挿入し，Sを$S[1, p-1] \circ \alpha \circ S[p, n]$に更新する．

4. $delete_S(p)$：文字列Sのp文字目を削除し，Sを$S[1, p-1] \circ S[p+1, n]$に更新する．

3.2 T'の種類数符号化

$T' = \alpha \circ T$とし，まず，$\alpha \in \Sigma$の場合の，種類数符号化の更新について，補題2に示す．

$$T = \overset{1\,2\,3\,4\,5\;6\,7\,8\,9}{\texttt{WZYXaXYW\$}} \qquad [\![T]\!] = \overset{1\,2\,3\,4\,5\;6\,7\,8\,9}{\texttt{4421a431\$}}$$

$$T' = \overset{1\,2\,3\,4\;5\,6\,7\,8\,9\,10}{\texttt{aWZYXaXYW\$}} \qquad [\![T']\!] = \overset{1\,2\,3\,4\;5\,6\,7\,8\,9\,10}{\texttt{a4421a431\$}}$$

図 3.1: 文字列 $T = \texttt{WZYXaXYW\$}$ に，文字 a を追加したときの種類数符号化の様子．$[\![T']\!] = \texttt{a} \circ [\![T]\!]$ になっている．

補題 2. $\alpha \in \Sigma$ のとき，$[\![T']\!] = \alpha \circ [\![T]\!]$ である．

証明. 文字列 T の位置 i における種類数符号化を考える．$T[i] \in \Sigma$ の場合，$[\![T']\!][i+1] = [\![T]\!][i] = T[i]$ なので，他の文字によらず符号化が行われる．$T[i] \in \Pi$ の場合，定義 6 より，$[\![T]\!][i] = \pi(T_i[1, \mathcal{L}(T_i, T[i])])$ であり，$[\![T']\!][i+1] = \pi(T'_{i+1}[1, \mathcal{L}(T'_{i+1}, T'[i+1])])$ である．したがって，先頭に追加した文字 α が定数文字の場合，$T'[i+1]$ の種類数符号化に影響しない．よって，$[\![T']\!][i+1] = [\![T]\!][i]$ である．

したがって，先頭に追加した文字 α が定数文字の場合は, 文字列 T の位置 i において $[\![T]\!][i] = [\![T']\!][i+1]$ が成立するので，$[\![T']\!] = \alpha \circ [\![T]\!]$ である． □

補題 2 の例を，図 3.1 に示す．

次に，$\alpha \in \Pi$ の場合の，種類数符号化の更新について示す．まず，$[\![T']\!][1]$ に関する補題を補題 3 に示す．

補題 3. $\alpha \in \Pi$ のとき，$[\![T']\!][1]$ は以下の通りである．

$$[\![T']\!][1] = \begin{cases} \pi(T) + 1 & \text{if } \mathcal{R}(T', \alpha) = 1 \\ \pi(T[1, \mathcal{L}(T, \alpha)]) & \text{otherwise} \end{cases}$$

証明. まず，$\mathcal{R}(T', \alpha) = 1$ のとき，つまり α が T 中に出現していないときについて考える．$T'[1] \in \Pi$ なので，$[\![T']\!][1]$ は，定義 6 より，位置 j を $j = \mathcal{L}(T'_1, T'[1])$ としたとき，$[\![T']\!][1] = \pi(T'_1[1, j])$ である．このとき，$T'[1] = \alpha$ は T 中に出現していないため，巡回文字列 $T'_1 = T'[2, n+1] \circ T'[1]$ で α が出現している位置は T'_1 の文末 $T'_1[n+1]$ のみである．したがって，位置 j は，$j = n+1$ となる．よって，$[\![T']\!][1] = \pi(T'_1[1, n+1])$ となる．また，このとき，T' 中に出現する変数文字の種類数 $\pi(T')$ は，$\pi(T') = \pi(T) + 1$ を満たす．なぜなら，α が，T' 中に出現し T 中に出現しない変数文字だからである．したがって，$\mathcal{R}(T', \alpha) = 1$ のとき，$[\![T']\!][1] = \pi(T) + 1$ である．

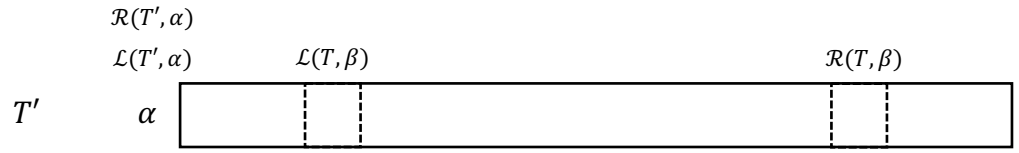

図 3.2: 文字列 T の先頭に，文字 α を追加したときの様子．α が変数文字，かつ T 中に出現していない場合，$\mathcal{R}(T',\alpha) = \mathcal{L}(T',\alpha) = 1$ となるので，$[\![T']\!][1]$ は，T 中に出現する変数文字の種類数に，α の分の 1 を足した自然数となる．

次に，α が T 中に出現しているときについて考える．このとき，$[\![T']\!][1]$ は，定義 6 より，位置 j を $j = \mathcal{L}(T'_1, T'[1])$ としたとき，$[\![T']\!][1] = \pi(T'_1[1,j])$ である．$T'_1 = T'[2,n+1] \circ T'[1] = T \circ T'[1]$ であるので，位置 j は T を用いて，$j = \mathcal{L}(T, T'[1])$ と表せる．したがって，$[\![T']\!][1]$ は，$T[1,j]$ 中に出現する変数文字の種類数となる．

□

補題 3 のイメージ図を，図 3.2 に示す．

次に，$[\![T]\!]$ と $[\![T']\!]$ に関する補題を補題 4 に示す．

補題 4. $\alpha \in \Pi$ のとき，位置 i $(1 \leq i)$ について，$T[i] \in \Sigma$ または，$T[i] \in \Pi$ かつ $i \neq \mathcal{R}(T, T[i])$ であるとき，以下が成立する．

$$[\![T']\!][i+1] = [\![T]\!][i]$$

証明. まず，$T[i] \in \Sigma$ の場合，$[\![T]\!][i] = T[i]$，$[\![T']\!][i+1] = T'[i+1]$ であり，$T[i] = T'[i+1]$ なので，$[\![T']\!][i+1] = T'[i+1] = T[i] = [\![T]\!][i]$ である．

次に，$T[i] \in \Pi$ の場合，$[\![T]\!][i] = \pi(T_i[1, \mathcal{L}(T_i, T[i])])$，$[\![T']\!][i+1] = \pi(T'_{i+1}[1, \mathcal{L}(T'_{i+1}, T[i])])$ である．また，$T_i = T[i+1,n] \circ T[1,i]$，$T'_{i+1} = T'[i+2,n+1] \circ T'[1,i+1] = T[i+1,n] \circ \alpha \circ T[1,i]$ である．

$[\![T']\!][i+1] \neq [\![T]\!][i]$ とすると，$\pi(T_i[1, \mathcal{L}(T_i, T[i])]) \neq \pi(T'_{i+1}[1, \mathcal{L}(T'_{i+1}, T[i])])$ である．このとき，巡回文字列 T_i と T'_{i+1} は，1 文字目から $n-i$ 文字目が $T[i+1,n]$ で同じである．したがって，$\mathcal{L}(T_i, T[i]) \leq n-i$ を満たすとき，$\mathcal{L}(T'_{i+1}, T[i])$ も $\mathcal{L}(T'_{i+1}, T[i]) \leq n-i$ を満たし，$[\![T']\!][i+1] = [\![T]\!][i]$ となる．よって，$[\![T']\!][i+1] \neq [\![T]\!][i]$ の仮定から，$\mathcal{L}(T_i, T[i]) > n-i$ である．$\mathcal{L}(T_i, T[i]) > n-i$ ということは，T_i の $n-i$ 文字目以降，$T[1,i]$ 中で $T[i]$ と同じ変数文字が出現したということであり，$T[i+1,n]$ 中には $T[i]$ と同じ変数文字が出現し

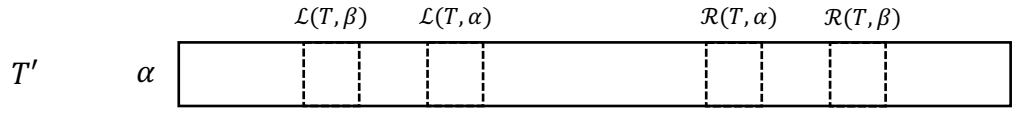

図 3.3: 文字列 T に，文字 α を追加したときの様子．T 中での α の最右位置 $\mathcal{R}(T,\alpha)$ に着目すると，$\mathcal{R}(T,\alpha)$ の巡回右側に α が出現している位置が，$\mathcal{L}(T,\alpha)$ から T' の先頭に変化していることが見て取れる．また，α 以外の変数文字 β の最右位置 $\mathcal{R}(T,\beta)$ に着目すると，先頭に α が追加される前は，$\mathcal{R}(T,\beta)$ の右側と，$\mathcal{L}(T,\beta)$ の左側に α は出現していないことがわかる．

ていないということである．つまり，位置 i は，文字 $T[i]$ の T 中での最右出現位置となるので，$i = \mathcal{R}(T, T[i])$ である．

□

したがって，補題 4 より，位置 i $(1 \leq i)$ について，$[\![T']\!][i+1] \neq [\![T]\!][i]$ の場合，$T[i] \in \Pi$ かつ $i = \mathcal{R}(T, T[i])$ である．補題 4 のイメージ図を，図 3.3 に示す．

続いて，各変数文字の最右出現位置での符号化の変化について補題 5, 6 に示す．

補題 5. α が T 中に出現していない変数文字のとき，α 以外の変数文字 β の T 中での最右位置を，$i = \mathcal{R}(T,\beta)$ としたとき，以下が成立する．

$$[\![T']\!][i+1] = [\![T]\!][i] + 1$$

証明. α 以外の変数文字 β の T 中での最右出現位置を，$i = \mathcal{R}(T,\beta)$ としたとき，$[\![T]\!][i] = \pi(T_i[1, \mathcal{L}(T_i, T[i])])], T_i = T[i+1, n] \circ T[1, i]$ である．$i = \mathcal{R}(T,\beta)$ なので，$T[i+1, n]$ 中に β は出現しない．したがって，$T[1, i]$ 中で β が最左に出現する位置を j とすると，$[\![T]\!][i] = \pi(T[i{+}1, n] \circ T[1, j])$ である．よって，$[\![T']\!][i{+}1] = \pi(T[i{+}1, n] \circ \alpha \circ T[1, j])$ である．α は T 中に出現していないので，$\pi(T[i{+}1, n] \circ T[1, j]){+}1 = \pi(T[i{+}1, n] \circ \alpha \circ T[1, j])$ である．したがって，$[\![T']\!][i{+}1] = \pi(T[i{+}1, n] \circ \alpha \circ T[1, j]) = \pi(T[i{+}1, n] \circ T[1, j]){+}1 = [\![T]\!][i]{+}1$ となる．

□

補題 3, 5 の例を，図 3.4 に示す．

$$T = \overset{1\,2\,3\,4\,5\,6\,7\,8\,9}{\texttt{WZYXaXYW\$}} \qquad [\![T]\!] = \overset{1\,2\,3\,4\,5\,6\,7\,8\,9}{4421a431\$}$$

$$T' = \overset{1\,2\,3\,4\,5\,6\,7\,8\,9\,10}{\texttt{VWZYXaXYW\$}} \qquad [\![T']\!] = \overset{1\,2\,3\,4\,5\,6\,7\,8\,9\,10}{54521a542\$}$$

図 3.4: 文字列 $T = \texttt{WZYXaXYW\$}$ の先頭に，変数文字 V を追加したときの種類数符号化の様子．V は T 中に出現していない変数文字なので，T 中に出現する変数文字の種類数が 4 であることから，$[\![T']\!][1] = \pi(T) + 1 = 5$ となっている．また，各変数文字の最右出現位置 $\mathcal{R}(T, \beta)$ に着目してみると，$[\![T]\!][\mathcal{R}(T, \texttt{W})] = 1$ であり，$[\![T']\!][\mathcal{R}(T', \texttt{W})] = 2$ となっており，$[\![T']\!][\mathcal{R}(T', \texttt{W})] = [\![T']\!][\mathcal{R}(T, \texttt{W}) + 1] = [\![T]\!][\mathcal{R}(T, \texttt{W})] + 1$ となっている．

続いて，α が T 中に出現しているときの各変数文字の最右出現位置での種類数符号化について，補題 6 に示す．

補題 6. α が T 中に出現している変数文字のとき，ある変数文字 β の T 中での最右出現位置を，$i = \mathcal{R}(T, \beta)$ としたとき，以下が成立する．

$$[\![T']\!][i+1] = \begin{cases} \pi(T[i+1, n]) + 1 & \text{if } \alpha = \beta \\ [\![T]\!][i] + 1 & \text{if } \mathcal{L}(T, \beta) < \mathcal{L}(T, \alpha) \text{ and } \mathcal{R}(T, \beta) > \mathcal{R}(T, \alpha) \\ [\![T]\!][i] & \text{otherwise} \end{cases}$$

証明. まず，$\alpha = \beta$ のときを考える．このとき，$[\![T']\!][i+1] = \pi(T'_{i+1}[1, \mathcal{L}(T'_{i+1}, T[i])]), T'_{i+1} = T[i+1, n] \circ \alpha \circ T[1, i]$ である．$i = \mathcal{R}(T, \beta)$ なので，$T[i+1, n]$ 中に β は出現しない．したがって，巡回文字列 T'_{i+1} での $\beta = \alpha$ の最初の出現は $n - i + 1$ 文字目である．よって，$[\![T']\!][i+1] = \pi(T'_{i+1}[1, n-i+1]) = \pi(T[i+1, n] \circ \alpha)$ であるので，$[\![T']\!][i+1] = \pi(T[i+1, n] \circ \alpha) = \pi(T[i+1, n]) + 1$ である．

次に，$\alpha \neq \beta$ のときを考える．位置 j を，T 中での β の最左出現位置とする．つまり，$j = \mathcal{L}(T, \beta)$ である．このとき，$[\![T']\!][i+1] = \pi(T'_{i+1}[1, n-i+j+1]), T'_{i+1}[1, n-i+j+1] = T[i+1, n] \circ \alpha \circ T[1, j]$ であり，$[\![T]\!][i] = \pi(T_i[1, n-i+j]), T_i[1, n-i+j] = T[i+1, n] \circ T[1, j]$ である．したがって，$[\![T]\!][i]$ と $[\![T']\!][i+1]$ の差異は，α が含まれるか否かである．よって，$T[i+1, n] \circ T[1, j]$ 中に α が含まれないときに $[\![T']\!][i+1] = [\![T]\!][i] + 1$ で，α が含まれるときに $[\![T']\!][i+1] = [\![T]\!][i]$ である．$T[i+1, n] \circ T[1, j]$ 中に α が含まれないときというのは，T において，位置 i よりも右側に α が出現しない，かつ位置 j よりも左側に α が出

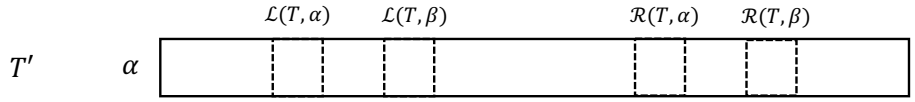

図 3.5: 文字列 T に，文字 α を追加したときの様子．$\mathcal{L}(T,\alpha) < \mathcal{L}(T,\beta)$ なので，補題 6 の 3 つめの条件に相当する．T の時点で $T[\mathcal{R}(T,\beta)]$ の種類数符号化に，α の分も含まれているため，$[\![T']\!][\mathcal{R}(T,\beta)+1] = [\![T]\!][\mathcal{R}(T,\beta)]$ である．

$$T = \overset{1\,2\,3\,4\,5\;6\,7\,8\,9}{\texttt{WZYXaXYW\$}} \qquad [\![T]\!] = \overset{1\,2\,3\,4\,5\;6\,7\,8\,9}{4421a431\$}$$

$$T' = \overset{1\,2\,3\,4\;5\,6\,7\,8\,9\,10}{\texttt{YWZYXaXYW\$}} \qquad [\![T']\!] = \overset{1\,2\,3\,4\;5\,6\,7\,8\,9\,10}{34421a422\$}$$

図 3.6: 文字列 $T = \texttt{WZYXaXYW\$}$ に，文字 $\texttt{Y}$ を追加したときの種類数符号化の様子．$T[4] = \texttt{X}$ は $T[6]$ が同じ変数文字となっており，$[\![T]\!][4]$ は $T[4,6]$ 中に出現する変数文字の種類数 1 で符号化されており，$[\![T']\!][5] = [\![T]\!][4]$ であることが見て取れる．

現しないことである．よって，$\mathcal{L}(T,\beta) < \mathcal{L}(T,\alpha)$ かつ $\mathcal{R}(T,\beta) > \mathcal{R}(T,\alpha)$ を満たすとき，$T[i+1,n] \circ T[1,j]$ 中に α が含まれない．

□

補題 6 のイメージ図を，条件が $\alpha = \beta$ のときと，$\mathcal{L}(T,\beta) < \mathcal{L}(T,\alpha)$ かつ $\mathcal{R}(T,\beta) > \mathcal{R}(T,\alpha)$ を満たすときは図 3.3，その他のときを図 3.5 に示し，具体例を図 3.6 に示す．

種類数符号化をオンラインで行うために，配列 L_T, R_T を導入する．配列 L_T は $L_T[c]$ に文字列 T に出現する変数文字 c の最左出現位置と $\texttt{\$}$ との距離を格納し，配列 R_T は $R_T[c]$ に文字列 T に出現する変数文字 c の最右出現位置と $\texttt{\$}$ との距離を格納する．つまり，$L_T[c] = |T| - \mathcal{L}(T,c), R_T[c] = |T| - \mathcal{R}(T,c)$ である．

例 5. $\Sigma = \{\texttt{\$},\texttt{a}\}, \Pi = \{\texttt{X},\texttt{Y},\texttt{Z}\}$ のとき，$T = \texttt{aXaYXXZYa\$}$ の配列 L_T, R_T は表 3.1 である．

種類数符号化のオンラインでの更新に関するアルゴリズムを Algorithm 3 に記す．T 中に出現する変数文字集合を Π' とする．また，配列 L_T, R_T の更新アルゴリズムを Algorithm 4 に記す．

配列 L_T, R_T と文字列 T の種類数符号化 $[\![T]\!]$ を用い，T の先頭に文字 α を追加した文字列 T' の種類数符号化 $[\![T']\!] = [\![\alpha \circ T]\!]$ の更新を行う．

定理 2. Algorithm 3, 4 を用いることで，変数文字のアルファベットサイズを $|\Pi|$ とすると，種類数符号化の更新と配列 L_T, R_T の更新は，共に $O(|\Pi|)$ 時間で行える．

Algorithm 3: T' の種類数符号化

input : 文字 α，文字列 T の種類数符号化 $[\![T]\!]$，配列 L_T, R_T，T 中に出現する変数文字集合 Π'

output: 文字列 $T' = \alpha \circ T$ の種類数符号化 $[\![T']\!]$

```
if α ∈ Σ then
    [[T']] = α ∘ [[T]];
else
    cou = 1, cou_2 = 0;
    if α ∈ Π' then
        while Π' ≠ ∅ do
            Let c be an element of Π';
            Π' = Π' \ {c};
            if L_T[c] > L_T[α] then
                cou = cou + 1;
                if R_T[c] < R_T[α] then
                    [[T]][|T| − R_T[c]] = [[T]][|T| − R_T[c]] − 1;
            if R_T[c] < R_T[α] then
                cou_2 = cou_2 + 1;
        [[T]][|T| − R_T[α]] = cou_2 + 1;
    else
        cou = |Π'| + 1;
        while Π' ≠ ∅ do
            Let c be an element of Π';
            Π' = Π' \ {c};
            [[T]][|T| − R_T[c]] = [[T]][|T| − R_T[c]] + 1;
    [[T']] = cou ∘ [[T]];
return [[T']];
```

Algorithm 4: 配列 L_T, R_T の更新

input : 文字 α，配列 L_T, R_T，T 中に出現する変数文字集合 Π'

output: 配列 $L_{T'}, R_{T'}$

```
L_T' = L_T,  R_T' = R_T;
if α ∈ Π then
    if α ∉ Π' then
        L_T'[α] = |T'| − 1;
        R_T'[α] = |T'| − 1;
    else
        L_T'[α] = |T'| − 1;
return L_T', R_T';
```

表 3.1: 文字列 $T =$ aXaYXXZYa\$ の (a) 配列 L_T，(b) 配列 R_T.

(a) 配列 L_T

	X	Y	Z
L_T	8	6	3

(b) 配列 R_T

	X	Y	Z
R_T	4	2	3

証明. 文字列 T' の種類数符号の更新は，先頭に追加した文字 α が定数文字の場合は，補題 2 より，先頭に α を追加するだけのため，$O(1)$ 時間で行える．α が変数文字の場合について考える．α が T 中に出現していない場合は，補題 5 より，各変数文字の最右位置の値を 1 増加させるため，配列 R_T を走査し各変数文字の最右位置を求めることで，種類数符号の更新ができるため，配列 R_T の要素数である変数文字のアルファベットサイズ $|\Pi|$ に線形時間の $O(|\Pi|)$ 時間で行える．α が T 中に出現している場合は，補題 6 より，α の最右位置の種類数符号化に関しては，α の最右位置から文末に出現する変数文字の種類数であるため，配列 R_T において，$R_T[\alpha] > R_T[\beta]$ を満たす変数文字 β が，α の最右位置から文末に出現する変数文字である．よって，配列 R_T を一度走査し，$R_T[\alpha] > R_T[\beta]$ を満たす変数文字 β の個数を数えることで，α の最右位置の種類数符号化を更新できる．また，α 以外の変数文字の最右位置の種類数符号化は，同じく補題 6 より，$L_T[\beta] > L_T[\alpha]$ と $R_T[\beta] < R_T[\alpha]$ を満たす変数文字 β では，$[\![T']\!][|T'| - R_T[\beta]] = [\![T]\!][|T| - R_T[\beta]] + 1$ となり，満たさない場合は, $[\![T']\!][|T'| - R_T[\beta]] = [\![T]\!][|T| - R_T[\beta]]$ となるため，配列 R_T, L_T を走査し，α との比較を行うことで更新できる．よって，配列 L_T, R_T のサイズとなる変数文字のアルファベットサイズを $|\Pi|$ とすると，$O(|\Pi|)$ 時間で種類数符号化の更新が行える．

また，配列 L_T, R_T の更新について考える．配列 $L_{T'}, R_{T'}$ は，T' 中に出現する変数文字 β に対して次のように定義される．

$$L_{T'}[\beta] = |T'| - \mathcal{L}(T', \beta),\ R_{T'}[\beta] = |T'| - \mathcal{R}(T', \beta)$$

文字列 T' は，$T' = \alpha \circ T$ なので $|T'| = |T| + 1$ であり，ある変数文字 β に関して，$\beta \neq \alpha$ のときは，$\mathcal{L}(T', \beta) = \mathcal{L}(T, \beta) + 1$，$\mathcal{R}(T', \beta) = \mathcal{R}(T, \beta) + 1$ である．したがって，$\beta \neq \alpha$

を満たす変数文字 β に関して $L_{T'}[\beta], R_{T'}[\beta]$ は，次のように表せる．

$$L_{T'}[\beta] = (|T|+1) - (\mathcal{L}(T,\beta)+1) = |T| - \mathcal{L}(T,\beta) = L_T[\beta]$$
$$R_{T'}[\beta] = (|T|+1) - (\mathcal{R}(T,\beta)+1) = |T| - \mathcal{R}(T,\beta) = R_T[\beta]$$

また，α が変数文字，かつ T 中に出現しているときは，$\mathcal{L}(T',\alpha) = 1$ となるので，$L_{T'}[\alpha], R_{T'}[\alpha]$ は，次のように表せる．

$$L_{T'}[\alpha] = (|T|+1) - 1 = |T|$$
$$R_{T'}[\alpha] = (|T|+1) - (\mathcal{R}(T,\alpha)+1) = |T| - \mathcal{R}(T,\alpha) = R_T[\alpha]$$

最後に α が変数文字，かつ T 中に出現していないときは，$\mathcal{L}(T',\alpha) = 1$，$\mathcal{R}(T'\alpha) = 1$ となるので，$L_{T'}[\alpha], R_{T'}[\alpha]$ は，次のように表せる．

$$L_{T'}[\alpha] = (|T|+1) - 1 = |T|$$
$$R_{T'}[\alpha] = (|T|+1) - 1 = |T|$$

したがって，配列 $L_{T'}, R_{T'}$ は，α 以外の要素は配列 L_T, R_T の要素をそのまま格納し，α の要素に関しては，文字列 T の長さを格納するため，$O(|\Pi|)$ 時間で配列 $L_{T'}, R_{T'}$ を構築できる． □

3.3 $pBWT(T'), F(T')$ の構築

3.3.1 $pBWT(T)$ と $pBWT(T')$ の違いについて

文字列 T のパラメタ化 BW 変換 $pBWT(T)$ と，文字列 $T' = \alpha \circ T$ のパラメタ化 BW 変換 $pBWT(T')$ の違いについて考える．

まず，$rot(T)$ の構成要素である文字列 T の巡回文字列 T_i の辞書式順序に関する補題を，補題 7 に示す．

補題 7. 文字列 T の巡回文字列 T_i, T_j $(i < j)$ の直前符号化後の文字列 $\langle T_i\rangle, \langle T_j\rangle$ の辞書式順序は，$\langle T_i[1, n-j]\rangle, \langle T_j[1, n-j]\rangle$ の辞書順に従う．

証明. 巡回文字列 $T_i, T_j (i < j)$ は，それぞれ $n-i$ 文字目，$n-j$ 文字目に \$ が出現する．つまり，$T_i = T[i+1, n-1] \circ \$ \circ T[1, i]$，$T_j = T[j+1, n-1] \circ \$ \circ T[1, j]$ である．\$ は辞書順最小の文字で，かつ T の文末に一度のみ出現する．したがって，巡回文字列 T_i, T_j 中にも \$ は一度のみ出現し，それぞれ，$n-i$ 文字目と $n-j$ 文字目に出現する．

今回は，$i < j$ なので，$n-i > n-j$ である．したがって，$\langle T_i \rangle, \langle T_j \rangle$ の辞書式順序は，$\langle T_j \rangle[n-j] \prec \langle T_i \rangle[n-j]$ より，必ず $n-j$ 文字目までで決定する． □

補題 7 と，文字列 T の先頭に文字 α を追加したとき，T の巡回文字列 T_i では，\$ の直後に α が追加されるという性質から，T の巡回文字列と T' の巡回文字列について，以下の補題が示すことができる．

補題 8. 文字列 T の巡回文字列 T_l, T_m と T'_{l+1}, T'_{m+1} に関して，以下のことが成り立つ．

$$\langle T_l \rangle \prec \langle T_m \rangle \Rightarrow \langle T'_{l+1} \rangle \prec \langle T'_{m+1} \rangle$$

証明. T の巡回文字列 T_l と T_m について考える．このとき，T_l, T_m の直前符号化は，$\langle T_l \rangle = \langle T[l+1, n] \circ T[1, l] \rangle$，$\langle T_m \rangle = \langle T[m+1, n] \circ T[1, m] \rangle$ である．補題 7 より，$\langle T_l \rangle$ と $\langle T_m \rangle$ の辞書式順序は，$l > m$ のときは $\langle T_l[1, n-l] \rangle$ と $\langle T_m[1, n-l] \rangle$ の辞書順に従う．つまり，$\langle T_l \rangle \prec \langle T_m \rangle$ であることから，$\langle T[l+1, n] \rangle \prec \langle T[m+1, m+n-l] \rangle$ である．また，T の巡回文字列 T_i では，\$ の後ろに α が追加される性質があるので，T_l と T_m の \$ の後ろに α を追加した文字列の直前符号化後の辞書順について考える．つまり，$\langle T'_{l+1} \rangle$ と $\langle T'_{m+1} \rangle$ の辞書順を考える．このとき，$\langle T'_{l+1} \rangle$ と $\langle T'_{m+1} \rangle$ は，次のように表記できる．

$$\langle T'_{l+1} \rangle = \langle T'[l+2, n+1] \circ T'[1, l+1] \rangle = \langle T[l+1, n] \circ \alpha \circ T[1, l] \rangle$$
$$\langle T'_{m+1} \rangle = \langle T'[m+2, n+1] \circ T'[1, m+1] \rangle = \langle T[m+1, n] \circ \alpha \circ T[1, m] \rangle$$

$\langle T[l+1, n] \rangle \prec \langle T[m+1, m+n-l] \rangle$ であることから，$\langle T'_{l+1} \rangle \prec \langle T'_{m+1} \rangle$ である．$l < m$ のときも同様である． □

補題 8 の説明図を，図 3.7 に示す．

補題 8 より，$rot(T)$ の各行は T の巡回文字列となっており，行方向にソートされてい

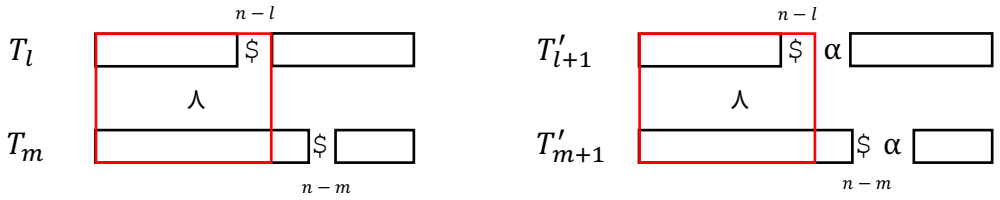

図 3.7: T の巡回文字列 T_l, T_m と，T'_{l+1}, T'_{m+1} の様子．ただし，$l > m$ である．T_l, T_m の辞書式順序は，赤枠部分で決定している．T'_{l+1}, T'_{m+1} の赤枠部分は T_l, T_m と変化していないことから，T'_{l+1}, T'_{m+1} の辞書式順序も赤枠部分で決定している．

	rot(T)	rot′(T')
1	$∞a∞∞1a252a	$∞∞a3∞1a252a
2	a$∞a∞∞1a252	a$∞∞a3∞1a252
3	a∞∞1a252a$∞	a∞∞1a252a$4∞
4	a∞∞2a$∞a661	a∞∞2a$4∞a661
5	∞a$∞a∞61a25	∞a$∞∞a361a25
6	∞a2∞2a$∞a66	∞a2∞2a$4∞a66
7	∞a∞∞1a252a$	∞a∞∞1a252a$4
8	∞1a2∞2a$∞a6	∞1a2∞2a$4∞a6
9	∞∞a$∞a661a2	∞∞a$4∞a661a2
10	∞∞1a252a$∞a	∞∞1a252a$4∞a
11	∞∞2a$∞a661a	∞∞2a$4∞a661a

図 3.8: 文字列 $T =$ XaYZZaZYZa$ の $rot(T)$ と，$rot(T)$ の各行に Y を追加した $rot'(T')$ の様子．

るので，$rot(T)$ の各行で $ の直後に α を追加したとき，行の入れ替わりは発生しない．$rot(T)$ の各行で $ の直後に α を追加したものを $rot'(T')$ とする．$rot(T)[i] = \langle T_l \rangle$ のとき，$rot'(T')[i] = \langle T'_{l+1} \rangle$ なので，$l \neq 0$ のときは，$\overline{rot}(T)[i][n] = T_l[n] = T[l]$，$\overline{rot'}(T')[i][n] = T'_{l+1}[n] = T'[l+1]$ であり，$l = 0$ のときは，$\overline{rot}(T)[i][n] = T_0[n] =$ $，$\overline{rot'}(T')[i][n] = T'_1[n] = T'[1]$ である．具体例を，図 3.8 に示す．

また補題 4 より，$[\![T]\!]$ と $[\![T']\!]$ において，ある i $(1 \leq i)$ について，$T[i] \in \Sigma$ または，$T[i] \in \Pi$ かつ $i \neq \mathcal{R}(T, T[i])$ であるとき，$[\![T']\!][i+1] = [\![T]\!][i]$ である．よって，$rot(T)$ のある行 i について，$rot(T)[i] = \langle T_l \rangle$ $(l \neq 0)$ を満たし，$T_l[n] \in \Sigma$ または，$T_l[n] \in \Pi$ かつ $l \neq \mathcal{R}(T, T_l[n])$ であるとき，$[\![\overline{rot'}(T')[i]]\!][n] = [\![T']\!][l+1] = [\![T]\!][l] = [\![\overline{rot}(T)[i]]\!][n]$ である．また，$rot(T)[i] = \langle T_0 \rangle$ のときは，$\overline{rot}(T)[i][n] = T_0[n] =$ $，$\overline{rot'}(T')[i][n] = T'_1[n] = T'[1]$ であることから，$[\![\overline{rot'}(T')[i]]\!][n] = [\![T']\!][1]$ である．

文字列 L' を以下のように定義する．ただし，$rot(T)[i] = \langle T_l \rangle$ を満たすものとする．

$$L'[i] = \begin{cases} [\![T]\!][l] & \text{if } T_l[n] \in \Sigma \text{ or } (T_l[n] \in \Pi \text{ and } l \neq \mathcal{R}(T, T_l[n])) \\ [\![T']\!][l+1] & \text{if } T_l[n] \in \Pi \text{ and } l = \mathcal{R}(T, T_l[n]) \\ [\![T']\!][1] & \text{if } l = 0 \end{cases}$$

$pBWT(T)$ は，定義 7 より $rot(T)[i] = \langle T_l \rangle$ であるとき，$pBWT(T)[i] = [\![T_l]\!][n] = [\![T]\!][l]$ で表されることから，L' は $pBWT(T)$ を用いて次のように表すことができる．

$$L'[i] = \begin{cases} pBWT(T)[i] & \text{if } T_l[n] \in \Sigma \text{ or } (T_l[n] \in \Pi \text{ and } l \neq \mathcal{R}(T, T_l[n])) \\ [\![T']\!][l+1] & \text{if } T_l[n] \in \Pi \text{ and } l = \mathcal{R}(T, T_l[n]) \\ [\![T']\!][1] & \text{if } l = 0 \end{cases}$$

したがって，$pBWT(T)[i] = \langle T_0 \rangle[n] = \$$ を満たす位置 i と，$rot(T)[j] = T_l$ のとき $T_l[n] \in \Pi$ かつ $l = \mathcal{R}(T, T_l[n])$ を満たす位置 j について，$pBWT(T)$ の値を変更することで，L' を構築できる．また，$rot'(T')$ は，$rot(T)$ の各行で $\$$ の直後に α を追加したものであるため，$rot'(T')[i] = \langle T' \rangle$ を満たすような行 i が存在しない．よって，$pBWT(T')$ を構築する際には，$rot(T')[p] = \langle T' \rangle$ を満たす行を p としたとき，L' の p 文字目に $rot(T')[i][n] = \langle T' \rangle[n] = \$$ を挿入する必要がある．

よって，$pBWT(T')$ の構築は，次の 2 つの操作を行う．

1. 文字列 L' を $pBWT(T)$ と $[\![T']\!]$ から構築する．
2. 文字列 L' の p 文字目に $\$$ を挿入する．

また，$rot'(T')$ の各行の先頭文字をつなげた文字列を F' とすると，$pBWT(T')$ を構築する方法と同様の方法で，F' と $[\![T']\!]$ から $F(T')$ の構築が行える．

3.3.2 文字列 L' の構築

まず，文字列 L', F' の構築を考える．文字列 L' は，$pBWT(T)$ と $[\![T']\!]$ を用い，以下のように表記できる．

$$L'[i] = \begin{cases} pBWT(T)[i] & \text{if } T_l[n] \in \Sigma \text{ or } (T_l[n] \in \Pi \text{ and } l \neq \mathcal{R}(T, T_l[n])) \\ [\![T']\!][l+1] & \text{if } T_l[n] \in \Pi \text{ and } l = \mathcal{R}(T, T_l[n]) \\ [\![T']\!][1] & \text{if } l = 0 \end{cases}$$

よって，$pBWT(T)$ から L' を構築する際に $L'[i] \neq pBWT(T)[i]$ となる行 i は，$pBWT(T)[i] =$ \$ か，ある変数文字 β に対して $rot(T)[i] = \langle T_{\mathcal{R}(T,\beta)} \rangle$ を満たす．

$pBWT(T)$ から L' を構築する際に，ある変数文字 β に対して $rot(T)[i] = \langle T_{\mathcal{R}(T,\beta)} \rangle$ を満たす行 i を見つける必要性がある．条件を満たす行の発見を容易にするために，**配列** MR を導入する．配列 MR は，以下のように定義する．

$$\mathrm{MR}[i] = \begin{cases} \beta & \text{if } \beta \in \Pi \text{ and } rot(T)[i] = \langle T_{\mathcal{R}(T,\beta)} \rangle \\ 0 & \text{otherwise} \end{cases}$$

配列 MR は，$pBWT(T)$ と同様に Navarro ら [8] が提案したデータ構造で保存するため，$rank$ 関数，$select$ 関数の計算を $O(\frac{\log n}{\log \log n})$ 時間で行える．したがって，$rot(T)[i] = \langle T_{\mathcal{R}(T,\beta)} \rangle$ を満たす行 i を，$select_{\mathrm{MR}}(\beta, 1)$ と演算することで $O(\frac{\log n}{\log \log n})$ 時間で発見できる．$pBWT(T)$ と $F(T)$ には，文字 α 同士の $pBWT(T)$ 上での出現順序と，$F(T)$ 上での出現順序が変化しない性質が成り立つので，$rot(T)[i] = \langle T_{\mathcal{R}(T,\beta)} \rangle$ を満たす行 i，$rot(T)[j] = \langle T_{\mathcal{R}(T,\beta)-1} \rangle$ を満たす行 j について，$rank_{pBWT(T)}([\![\overline{rot}(T)[i]]\!][n], i) = rank_{F(T)}([\![\overline{rot}(T)[i]]\!][n], j)$ が成立する．よって位置 j は，$select$ 関数を用い $select_{F'}(pBWT(T)[i], rank_{L'}(pBWT(T)[i], i))$ で導出できるため，$O(\frac{\log n}{\log \log n})$ 時間で発見できる．

定理 3. Algorithm 5 を用いることで，変数文字のアルファベットサイズを $|\Pi|$ とすると，文字列 L', F' の構築は $O(|\Pi|\frac{\log n}{\log \log n})$ 時間である．

証明. まず，$pBWT(T)[i] \neq L'[i]$ を満たす位置 i は，補題 4 よりある変数文字 β について，$rot(T)[i] = \langle T_{\mathcal{R}(T,\beta)} \rangle$ のときなので，最大で $|\Pi|$ 箇所存在する．

Algorithm 5: L', F' の構築

input : 文字列 $pBWT(T), F(T)$，文字列 T' の種類数符号化 $[\![T']\!]$，Algorithm 3 で更新された変数文字集合 P，配列 MR，配列 $R_{T'}$

output: 文字列 L', F'

```
L' = pBWT(T);
F' = F(T);
while P ≠ ∅ do
    Let α be an element of P;
    P = P \ {α};
    poj = select_MR(α, 1);
    cou = rank_{L'}(α, poj);
    poj_2 = select_{F'}(α, cou);
    L'[poj] = [[T']][|T| − R_{T'}[α]];
    F'[poj_2] = [[T']][|T| − R_{T'}[α]];
poj = select_{L'}($, 1);
L'[poj] = [[T']][1];
return L', F';
```

次に，$pBWT(T)[i] \neq L'[i]$ を満たす位置 i は，$i = select_{MR}(\beta, 1)$ によって得られる．F' 上での更新箇所 i' の検索は，$pBWT(T)$ と $F(T)$ での文字の出現位置の関係より，$j = rank_{pBWT(T)}(\beta, i)$，$i' = select_{F(T)}(\beta, j)$ の計算を行うことで可能である．結果，$rank$ 関数，$select$ 関数を用い更新箇所を検索できるため $O(\frac{\log n}{\log \log n})$ 時間で検索できる．

$pBWT(T)[i] \neq L'[i]$ を満たす位置を i としたとき，$L'[i]$ と $F'[i]$ の値の更新は，先に求めた変更箇所 i, i' を用い，$delete_{L'}(i)$，$delete_{F'}(i')$ の後に，$insert_{L'}([\![T']\!][\mathcal{R}(T', \beta)], i)$，$insert_{F'}([\![T']\!][\mathcal{R}(T', \beta)], i')$ を行うことで更新できるため，$O(\frac{\log n}{\log \log n})$ 時間で可能である．

最後に，$pBWT(T)[i] = \$$ を満たす位置を i とすると，$L'[i]$ の値を $[\![T']\!][1]$ へと更新する．まず，位置 i は，$rank_{pBWT(T)}(\$, 1)$ で見つけることができる．$L'[i]$ の値の更新は，先ほどと同様に $delete_{L'}(i)$ 後，$insert_{L'}([\![T']\!][1], i)$ を行うことで可能なので，$L'[i]$ の値を $[\![T']\!][1]$ へと更新するのにかかる時間は，$O(\frac{\log n}{\log \log n})$ 時間である．

したがって，文字列 L', F' の構築は，変数文字のサイズを $|\Pi|$ とすると $O(|\Pi|\frac{\log n}{\log \log n})$ 時間である． □

3.3.3 $rot(T')[p] = \langle T' \rangle$ を満たす位置 p の計算

次に， 3.3.2 で構築した L' と F' から，$pBWT(T')$ と $F(T')$ を構築する．具体的には，L', F' は，$rot'(T')[p] = \langle T' \rangle$ を満たす行 p が存在しない $rot'(T')$ から構築されている．したがって，$rot(T')[p] = \langle T' \rangle$ を満たす行を p としたとき，$pBWT(T') = L'[1, p-1] \circ [\![T']\!][n+1] \circ L'[p, n] = L'[1, p-1] \circ \$ \circ L'[p, n]$，$F(T') = F'[1, p-1] \circ [\![T']\!][1] \circ F'[p, n]$ を満たす．よって，$rot(T')[p] = \langle T' \rangle$ を満たす位置 p を L', F' から求め，$pBWT(T'), F(T')$ を構築する．

準備として，$LCP_\infty(T)$ 配列と，lcp_∞ 関数を以下のように定義する．

定義 9. $LCP_\infty(T)$ 配列は，$LCP_\infty(T)[i]$ に $rot(T)$ の i 行目と $i+1$ 行目の最長共通接頭辞内に出現した ∞ の数を格納する．$rot(T)[i], rot(T)[i+1]$ の最長共通接頭辞の長さを k としたとき，$rot(T)[i][1, k] = rot(T)[i+1][1, k]$ かつ $rot(T)[i][k+1] \neq rot(T)[i+1][k+1]$ である．このとき，$LCP_\infty(T)[i]$ は以下の式で表せる．

$$LCP_\infty(T)[i] = rank_{rot(T)[i]}(\infty, k)$$

また，lcp_∞ 関数は，2 つの自然数 i, j を引数として受け取り，$rot(T)$ の i 行目と j 行目の最長共通接頭辞内に出現した ∞ の数を返す関数とする．lcp_∞ 関数は，$LCP_\infty(T)$ 配列を用い，以下の式で導出できる．

$$lcp_\infty(i, j) = \min(LCP_\infty(T)[k] | i \leq k < j)$$

$LCP_\infty(T)$ 配列の具体例を，図 3.9 に示す．

位置 p を求める上で，Kim らの補題 [7] を記す．

補題 9. $pBWT(T)[i], pBWT(T)[j] \in \mathbb{N}\ (i < j)$ において，$pBWT(T)[i] > pBWT(T)[j]$ かつ $pBWT(T)[j] \leq lcp_\infty(i, j)$ を満たすとき $pLF[i] > pLF[j]$，満たさないとき $pLF[i] < pLF[j]$ である．

これは，$pBWT(T)[i], pBWT(T)[j] \in \mathbb{N}\ (i < j)$ の位置関係が，条件を満たすとき $F(T)$ 上で逆転し，条件を満たさないとき $pBWT(T)$ 上と $F(T)$ 上での位置関係が変化しない

	$LCP_\infty(T)$	rot(T)
1	0	$∞a∞∞1a252a
2	0	a$∞a∞∞1a252
3	2	a∞∞1a252a$∞
4	0	a∞∞2a$∞a661
5	1	∞a$∞a∞61a25
6	1	∞a2∞2a$∞a66
7	1	∞a∞∞1a252a$
8	1	∞1a2∞2a$∞a6
9	2	∞∞a$∞a661a2
10	2	∞∞1a252a$∞a
11	-	∞∞2a$∞a661a

図 3.9: 文字列 $T =$ `XaYZZaZYZa$` の $LCP_\infty(T)$ 配列と $rot(T)$. $rot(T)$ の 3 行目と 4 行目に着目してみると，最長共通接頭辞は 3 文字目までであり，$rank_{rot(T)[3]}(\infty, 3) = 2 = LCP_\infty(T)[3]$ となっていることがわかる．

ことを表している．

この補題 9 を用いることで L' 上での $[\![T']\!][1]$ と他の文字との位置関係から，$F(T')$ 上での位置関係を求めることができ，位置 p を求めることができる．

具体的には，$pBWT(T)$ 上での `$` の位置を t とすると，$L'[t] = [\![T']\!][1]$ と $L'[i]$ $(1 \leq i \leq n$ かつ $i \neq t)$ をそれぞれ補題 9 の条件と照らし合わせ，$F(T')$ 上で $L'[t]$ よりも $L'[i]$ が前方に出現するか，後方に出現するかを確認することで $rot(T')[p] = \langle T' \rangle$ を満たす位置 p を求めることができる．

しかし，$L'[t]$ と $L'[i]$ を 1 文字ずつ確認すると，文字列長に線形な時間を必要とする．そのため，L' 上に出現する文字に着目することで位置 p を求める．

定義 10. $rot(T')[p] = \langle T' \rangle$ を満たす位置 p は，以下のように求められる．また，l_k は自然数 k ごとに定められる L' 上の位置とし，x を T 中に出現する定数文字のうち辞書式順序が一番大きいものとする．

1. $[\![T']\!][1] \in \Sigma$ のとき：$p = C_T[T'[1]] + rank_{L'}(T'[1], t-1)$

2. $[\![T']\!][1] \in \mathbb{N}$ のとき：$p = C_T[x] + rank_{L'}(x, n) + \sum_{k=1}^{|\Pi|} rank_{L'}(k, l_k)$

上記定義に出現している l_k の求め方を次に記す．

補題 10. l_k は，以下のように求められる．

1. $k < [\![T']\!][1]$ のとき：$t \leq l_k$ かつ $LCP_\infty(T)[l_k] < k$ を満たす最小の位置

2. $k = [\![T']\!][1]$ のとき：$l_k = t - 1$

3. $k > [\![T']\!][1]$ のとき：$t > l_k$ かつ $LCP_\infty(T)[l_k] < L'[t]$ を満たす最大の位置

証明. まず，$k < [\![T']\!][1]$ のときを考える．k の L' 上の出現位置を p_k とする．つまり，$L'[p_k] = k$ である．$p_k < t$ のときは，補題 9 の条件である $pBWT(T)[i] > pBWT(T)[j]$ を満たさず，$L'[p_k] = k < [\![T']\!][1] = L'[t]$ となるため，$p_k < t$ を満たす k に関しては，$F(T')$ 上で $L'[t]$ よりも前方に出現する．$p_k \geq t$ のときは，補題 9 の条件である $pBWT(T)[i] > pBWT(T)[j]$ を満たすため，p_k が 2 つ目の条件である $pBWT(T)[j] \leq lcp_\infty(i, j)$ を満たす場合，位置関係が $F(T')$ 上で入れ替わり $L'[t]$ よりも前方に出現することとなる．このとき，$LCP_\infty(T)[l'] < k$ を満たす最小の位置を l' とすると $lcp_\infty(t, l'+1)$ の値は $LCP_\infty(T)[l']$ 以下となるので，$lcp_\infty(t, l'+1) \leq LCP_\infty(T)[l'] < k$ となり，k よりも小さくなることがわかる．その結果，$lcp_\infty(t, l'+a)$ $(a \geq 1)$ の値は，k よりも小さくなるため補題 9 の 2 つ目の条件を満たさなくなる．よって，$t \leq l$ かつ $LCP_\infty(T)[l] < k$ を満たす最小の位置を l とすると，$L'[1, l]$ 中に出現する k の個数が $F(T')$ 上で $L'[t]$ より前方に出現する k の個数となる．

次に，$k = [\![T']\!][1]$ のときを考える．$p_k < t$ のときは，補題 9 の条件である $pBWT(T)[i] > pBWT(T)[j]$ を満たさないため，$p_k < t$ を満たす k に関しては，$F(T')$ 上で $L'[t]$ よりも前方に出現する．$p_k \geq t$ のときは，補題 9 の条件である $pBWT(T)[i] > pBWT(T)[j]$ を満たさないため，$p_k \geq t$ を満たす k に関しては，$F(T')$ 上で $L'[t]$ よりも後方に出現するよって，$l = t - 1$ とし，$L'[1, l]$ 中に出現する k の個数が $F(T')$ 上で $L'[t]$ よりも前方に出現する k の個数となる．

最後に，$k > [\![T']\!][1]$ のときを考える．$p_k \geq t$ のときは，補題 9 の条件の $pBWT(T)[i] > pBWT(T)[j]$ を満たさないため，$p_k \geq t$ を満たす k に関しては，$F(T')$ 上で $L'[t]$ よりも後方に出現する．$p_k < t$ のときは，補題 9 の条件である $pBWT(T)[i] > pBWT(T)[j]$ を満たすため，2 つ目の条件である $pBWT(T)[j] \leq lcp_\infty(i, j)$ を満たす必要がある．このとき，$LCP_\infty(T)[l'] < L'(T)[t]$ を満たす最大の位置を l' とすると $lcp_\infty(l', t)$ の値は $LCP_\infty(T)[l']$ となるので，$lcp_\infty(l', t) = LCP_\infty(T)[l'] < L'[t]$ となり，$L'[t]$ よりも小さくなることがわかる．また，$l' + a$ $(1 \leq a < t - l')$ の位置では，$LCP_\infty(T)[l' + a] \geq L'[t]$ となるため，補題 9 の 2 つ目の条件を満たす．よって $p_k < t$ のとき，$t > l$ かつ $LCP_\infty(T)[l] < L'[t]$ を満

たす最大の位置を l とすると，$L'[1,l]$ 中に出現する k の個数が $F(T')$ 上で $L'[t]$ より前方に出現する k の個数となる．□

$rot(T')[p] = \langle T' \rangle$ を満たす位置 p を求める計算アルゴリズムを Algorithm 6 に記す．

Algorithm 6: $rot(T')[p] = \langle T' \rangle$ を満たす位置 p を求めるアルゴリズム

input : 文字列 L', F'，文字 $[\![T']\!][1]$，配列 $LCP_\infty(T)$，配列 C_T，$pBWT(T)[t] = \$$ を満たす位置 t，T 中に出現する定数文字のうち辞書式順序最大の文字 x

output: 位置 p

```
1  p = 0;
2  if [[T']][1] ∈ Σ then
3  |  p = rank_{L'}(T'[1], t − 1);
4  |  p = p + C_T[T'[1]];
5  else
6  |  p = C_T[x];
7  |  p = p + rank_{L'}(x, n);
8  |  l = |T|;
9  |  for i = 1 to [[T']][1] do
10 |  |  if i < [[T']][1] then
11 |  |  |  cou = rank_{LCP_∞(T)}(i − 1, l) − rank_{LCP_∞(T)}(i − 1, t − 1);
12 |  |  |  if cou ≠ 0 then
13 |  |  |  |  l = select_{LCP_∞(T)}(i − 1, rank_{LCP_∞(T)}(i − 1, t − 1) + 1)
14 |  |  |  p = p + rank_{L'}(i, l)
15 |  |  else
16 |  |  |  p = p + rank_{L'}(i, t − 1);
17 |  l = 0;
18 |  for i = 0 to [[T']][1] − 1 do
19 |  |  cou = rank_{LCP_∞(T)}(i, t);
20 |  |  if cou ≠ 0 then
21 |  |  |  l_p = select_{LCP_∞(T)}(i, cou);
22 |  |  |  if l < l_p then
23 |  |  |  |  l = l_p;
24 |  for i = [[T']][1] + 1 to |Π| do
25 |  |  p = p + rank_{L'}(i, l);
26 return p;
```

定理 4. Algorithm 6 を用いることで，変数文字のアルファベットサイズを $|\Pi|$ とすると，$rot(T')[p] = \langle T' \rangle$ を満たす位置 p は，$O(|\Pi|\frac{\log n}{\log\log n})$ 時間で求められる．

証明. まず，$T'[1] \in \Sigma$ の場合は，配列 C_T の値と L' に対する $rank$ 関数で $F(T')$ 上で $T'[1]$ よりも前方に出現する文字の出現回数を求められるため，$O(\frac{\log n}{\log\log n})$ 時間で求められる．

次に，$T'[1] \in \Pi$ の場合は，仕切り位置 l_k と $L'[1, l_k]$ に出現する対象となる文字の出現個数を交互に更新することで挿入位置を求める．$L'[1, l_k]$ に出現する対象となる文字の出現個数を数える部分では，$rank$ 関数を用い数えることができるので $O(\frac{\log n}{\log\log n})$ 時間で可能である．

仕切り位置 l_k の導出にかかる時間は，補題 10 より L' 上に出現する文字 $k \in \mathbb{N}$ と $[\![T']\!][1]$ の大小関係で考える．また，このとき $rot(T)[t][n] = \$$ を満たす位置を t とする．つまり，$L'[t] = [\![T']\!][1]$ を満たす位置である．まず，$k < [\![T']\!][1]$ のときを考える．自然数 $k-1$ に関する仕切り位置を l_{k-1} としたとき，l_k は $LCP_\infty(T)[t, l_{k-1}]$ 中に $k-1$ が出現している場合は，$l_k = \min\{j | LCP_\infty(T)[j] = k-1 \text{ and } t \leq j \leq l_{k-1}\}$ となり，出現していない場合は，$l_k = l_{k-1}$ となる．なぜならば，$l_{k-1} < j \leq n$ 満たす位置 j と t に関して，$lcp_\infty(t, j) < k-1$ となり，位置 j に関しては，既に $lcp_\infty(t, j) < k$ を満たす．したがって，l_k の更新は $LCP_\infty(T)[t, l_{k-1}]$ 中での $k-1$ の出現確認と，出現する場合にはその中で最小の位置を探す．よって，$rank$ 関数と $select$ 関数を用いて l_k の更新を行えるため，$O(([\![T']\!][1]-1)\frac{\log n}{\log\log n})$ 時間で求められる．

次に，$k = [\![T']\!][1]$ のときは，補題 10 の 2 つ目より $l_k = t-1$ と定まるので $O(1)$ 時間で l_k を求めることが出来る．

最後に，$k > [\![T']\!][1]$ のときは，補題 10 の 3 つ目の条件のうち $LCP_\infty(T)[l_k] < L'[t]$ に着目すると，$LCP_\infty(T)[l_k]$ と $L'[t]$ との比較のため k に依存しない条件となる．よって，l_k を逐次更新する必要がないため一度の導出で良い．l_k を求める流れでは，まず $LCP_\infty(T)[l_k] < L'[t]$ の条件より $LCP_\infty(T)[1, t-1]$ において $[\![T']\!][1]$ よりも小さい値の仕切り位置 l' を求める．l' は $rank$ 関数を用いた後に $select$ 関数を用いることで導出できる．l' の中で一番最大の位置が l_k となる．よって，l_k は $O(|\Pi|\frac{\log n}{\log\log n})$ 時間で求められる．

したがって，$T'[1] \in \mathbb{N}$ の場合は，$O(|\Pi|\frac{\log n}{\log\log n})$ 時間で位置 p を求めることができる．

よって，$rot(T')[p] = \langle T' \rangle$ を満たす位置 p の計算は，変数文字のサイズを $|\Pi|$ とすると $O(|\Pi|\frac{\log n}{\log\log n})$ 時間で行える． □

3.4 $LCP_\infty(T)$ 配列の更新

最後に，$LCP_\infty(T)$ 配列の更新を行う．

$LCP_\infty(T)$ 配列は，$rot(T)$ で隣接する行の最長共通接頭辞内に含まれる ∞ の個数を格納した配列である．補題 7 と，T の巡回文字列 T_i では，\$ の後ろに α が追加される性質より，$rot'(T')$ で隣接する行の最長共通接頭辞内に含まれる ∞ の個数を格納した配列を LCP'_∞ 配列とすると，$LCP'_\infty[i] = LCP_\infty[i]$ である．これは，$rot(T)$ と $rot'(T')$ のある行 i は，$rot(T)[i][k] =$ \$ とすると，$rot'(T')[i][k] =$ \$ を満たし，かつ T 中に \$ が一度しか出現しないため，$rot(T)$ の隣接行の共通接頭辞中に \$ が含まれないことから，隣接する行の最長共通接頭辞は変化しない．そのため，最長共通接頭辞内に含まれる ∞ の個数も変化しないので，$LCP'_\infty[i] = LCP_\infty[i]$ が成立する．上記点を踏まえ，$LCP_\infty(T')$ 配列について以下の補題が成立する．

補題 11. $LCP_\infty(T')$ 配列は，$LCP_\infty(T)$ 配列の要素を用いて以下のように表記できる．また，位置 t は $rot(T')[t] = T'$ を満たす位置とし，l を $rot(T')[t-1]$ と $rot(T')[t]$ の最長共通接頭辞の長さ，l' を $rot(T')[t]$ と $rot(T')[t+1]$ の最長共通接頭辞の長さとする．

$$LCP_\infty(T')[i] = \begin{cases} LCP_\infty(T)[i] & \text{if } i < t-1 \\ rank_{rot(T')[i]}(\infty, l) & \text{else if } i = t-1 \\ rank_{rot(T')[i+1]}(\infty, l') & \text{else if } i = t \\ LCP_\infty(T)[i-1] & \text{otherwise} \end{cases}$$

補題 11 より，$LCP_\infty(T')$ の構築は，$rot(T')[k] = T'$ としたとき，$rot(T')[k-1], rot(T')[k]$ の最長共通接頭辞内に含まれる ∞ の個数と，$rot(T')[k], rot(T')[k+1]$ の最長共通接頭辞内に含まれる ∞ の個数を求めることが必要となる．

$LCP_\infty(T')$ の構築の具体的な手法の前に，$pBWT(T)$ と lcp_∞ 関数に関する定理を記す．

定理 5. $pBWT(T)[i], pBWT(T)[j], lcp_\infty(i,j)$ から，$lcp_\infty(i',j')$ は次のパターンに分けられる．また，i', j' は，$rot(T)[i] = \langle T_l \rangle$，$rot(T)[j] = \langle T_m \rangle$ のとき，$rot(T)[i'] = \langle T_{l-1} \rangle$，$rot(T)[j'] = \langle T_{m-1} \rangle$ を満たす行とする．

1. $pBWT(T)[i], pBWT(T)[j] \in \Sigma$ のとき

   (a) $pBWT(T)[i] = pBWT(T)[j]$ のとき：$lcp_\infty(i', j') = lcp_\infty(i, j)$

   (b) $pBWT(T)[i] \neq pBWT(T)[j]$ のとき：$lcp_\infty(i', j') = 0$

2. $(pBWT(T)[i] \in \Sigma \wedge pBWT(T)[j] \in \mathbb{N}) \vee (pBWT(T)[i] \in \mathbb{N} \wedge pBWT(T)[j] \in \Sigma)$ のとき：$lcp_\infty(i', j') = 0$

3. $pBWT(T)[i], pBWT(T)[j] \in \mathbb{N}$ のとき

   (a) $lcp_\infty(i, j) < \min(pBWT(T)[i], pBWT(T)[j])$ のとき：
   $lcp_\infty(i', j') = lcp_\infty(i, j) + 1$

   (b) $(lcp_\infty(i, j) \geq pBWT(T)[i])$ かつ $(pBWT(T)[j] > pBWT(T)[i])$ のとき：
   $lcp_\infty(i', j') = pBWT(T)[i]$

   (c) $(lcp_\infty(i, j) \geq pBWT(T)[j])$ かつ $(pBWT(T)[i] > pBWT(T)[j])$ のとき：
   $lcp_\infty(i', j') = pBWT(T)[j]$

   (d) $(lcp_\infty(i, j) \geq pBWT(T)[i])$ かつ $(pBWT(T)[i] = pBWT(T)[j])$ のとき：
   $lcp_\infty(i', j') = lcp_\infty(i, j)$

証明. $rot(T)[i], rot(T)[j]$ の最長共通接頭辞の長さを p とする．つまり，$rot(T)[i][1, p] = rot(T)[j][1, p]$ かつ $rot(T)[i][p+1] \neq rot(T)[j][p+1]$ である．$rot(T)[i] = \langle T_l \rangle, rot(T)[j] = \langle T_m \rangle$ とすると，最長共通接頭辞の長さは p なので，$\langle T_l[1, p] \rangle = \langle T[l + 1, l + p + 1] \rangle = \langle T_m[1, p] \rangle = \langle T[m + 1, m + p + 1] \rangle$ である．また \$ が，T の文末に一度しか出現しないため，T の異なる巡回文字列 T_l, T_m $(l \neq m)$ での \$ の出現位置が異なる．したがって，T の巡回文字列を直前符号化した $rot(T)[i]$ と $rot(T)[j]$ の最長共通接頭辞中に \$ は含まれない．よって，$l + p + 1 < n, m + p + 1 < n$ である．

まず，$pBWT(T)[i], pBWT(T)[j] \in \Sigma$ かつ $pBWT(T)[i] = pBWT(T)[j] = \alpha$ のときを考える．このとき，それぞれの末尾の文字を先頭に巡回したときの巡回文字列 T_{l-1}, T_{m-1} は，巡回する前の文字列を用いて $T_{l-1}[1, p+1] = \alpha \circ T_l[1, p], T_{m-1}[1, p+1] = \alpha \circ T_m[1, p]$ と表せる．$\langle T_l[1, p] \rangle = \langle T_m[1, p] \rangle$ から，$\langle T_{l-1}[1, p + 1] \rangle = \langle T_{m-1}[1, p + 1] \rangle$ である．また，$\langle T_l[1, p] \rangle$ 中に出現する ∞ の個数と，$\langle T_{l-1}[1, p + 1] \rangle$ 中に出現する ∞ の個数は，先頭に追

加した α が定数文字であることから変化しない．したがって，最長共通接頭辞内の ∞ の個数は変化しないため，$lcp_\infty(i', j') = lcp_\infty(i, j)$ となる．

また，$pBWT(T)[i], pBWT(T)[j] \in \Sigma$ かつ $pBWT(T)[i] \neq pBWT(T)[j]$ のときを考える．それぞれの末尾の文字を先頭に巡回したときの巡回文字列 T_{l-1}, T_{m-1} は，巡回する前の文字列を用いて $T_{l-1}[1, p+1] = pBWT(T)[i] \circ T_l[1, p], T_{m-1}[1, p+1] = pBWT(T)[j] \circ T_m[1, p]$ と表せる．$pBWT(T)[i] \neq pBWT(T)[j]$ なので，$\langle T_{l-1}[1] \rangle \neq \langle T_{m-1}[1] \rangle$ となる．したがって，最長共通接頭辞の長さが 0 となるので，$lcp_\infty(i', j') = 0$ となる．

次に，$(pBWT(T)[i] \in \Sigma \wedge pBWT(T)[j] \in \mathbb{N}) \vee (pBWT(T)[i] \in \mathbb{N} \wedge pBWT(T)[j] \in \Sigma)$ のときは，$pBWT(T)[i], pBWT(T)[j] \in \Sigma$ かつ $pBWT(T)[i] \neq pBWT(T)[j]$ のときと同様に末尾の文字を先頭に巡回したときの巡回文字列 T_{l-1}, T_{m-1} の先頭の文字が異なるので，最長共通接頭辞の長さが 0 となる．したがって，$lcp_\infty(i', j') = 0$ となる．

最後に，$pBWT(T)[i], pBWT(T)[j] \in \mathbb{N}$ のときを考える．$\overline{rot}(T)[i]$ での $\overline{rot}(T)[i][n]$ の最左出現位置を k_i，$\overline{rot}(T)[j]$ での $\overline{rot}(T)[j][n]$ の最左出現位置を k_j とする．つまり，$k_i = \mathcal{L}(\overline{rot}(T)[i], \overline{rot}(T)[i][n])$，$k_j = \mathcal{L}(\overline{rot}(T)[j], \overline{rot}(T)[j][n])$ である．$pBWT(T)[i] \in \mathbb{N}$ のとき，$pBWT(T)[i] = \pi(\overline{rot}(T)[i][1, k_i])$ であり，$rot(T)$ で考えると，$pBWT(T)[i] = rank_{rot(T)[i]}(\infty, k_i)$ であるため，$rot(T)[i'], rot(T)[j']$ は以下のように表記できる．

$$rot(T)[i'] = \infty \circ rot(T)[i][1, k_i - 1] \circ k_i \circ rot(T)[i][k_i + 1, n - 1]$$
$$rot(T)[j'] = \infty \circ rot(T)[j][1, k_j - 1] \circ k_j \circ rot(T)[j][k_j + 1, n - 1]$$

また，$pBWT(T)[i] = rank_{rot(T)[i]}(\infty, k_i)$ であることから，位置 k_i は $rot(T)[i]$ において ∞ が $pBWT(T)[i]$ 個目に出現した位置である．

上記のことを踏まえて，まず，$lcp_\infty(i, j) < \min(pBWT(T)[i], pBWT(T)[j])$ のときを考える． このとき，$\min(pBWT(T)[i], pBWT(T)[j]) = pBWT(T)[i]$ としておく．$rot(T)[i]$ と $rot(T)[j]$ の最長共通接頭辞の長さを p としたとき，$rot(T)[i][1, p] = rot(T)[j][i, p]$ かつ $rot(T)[i][p+1] \neq rot(T)[j][p+1]$ であり，$lcp_\infty(i, j) = rank_{rot(T)[i]}(\infty, p)$ である．位置 k_i は，$rot(T)[i]$ 中で ∞ が $pBWT[i]$ 個目に出現した位置であるが，$lcp_\infty(i, j) < \min(pBWT(T)[i], pBWT(T)[j])$ より，$p < k_i$ であり，位置 k_j についても同様のことが言えるため，$p < k_j$ である．よって，$rot(T)[i'], rot(T)[j']$ は p, k_i, k_j を用いて以下のように

表記できる．

$$rot(T)[i'] = \infty \circ rot(T)[i][1,p] \circ rot(T)[i][p+1,k_i-1] \circ k_i \circ rot(T)[i][k_i+1,n-1]$$
$$rot(T)[j'] = \infty \circ rot(T)[j][1,p] \circ rot(T)[j][p+1,k_j-1] \circ k_j \circ rot(T)[j][k_j+1,n-1]$$

したがって，$rot(T)[i']$ と $rot(T)[j']$ の最長共通接頭辞の長さは $p+1$ であり，$rot(T)[i'][1,p+1] = \infty \circ rot(T)[i][1,p]$ なので，$lcp_\infty(i',j') = rank_{rot(T)[i']}(\infty, p+1) = rank_{rot(T)[i]}(\infty,p)+1$ である．よって，$lcp_\infty(i,j) < \min(pBWT(T)[i], pBWT(T)[j])$ のときは，$lcp_\infty(i',j') = lcp_\infty(i,j)+1$ となる．

次に，$(lcp_\infty(i,j) \geq pBWT(T)[i])$ かつ $(pBWT(T)[j] > pBWT(T)[i])$ のときを考える．$lcp_\infty(i,j) \geq pBWT(T)[i]$ のため $p > k_i$ である．よって，$rot(T)[i']$ は p, k_i を用いて以下のように表記できる．

$$rot(T)[i'] = \infty \circ rot(T)[i][1,k_i-1] \circ k_i \circ rot(T)[i][k_i+1,p] \circ rot(T)[i][p+1,n-1]$$

したがって，$rot(T)[i']$ と $rot(T)[j']$ の最長共通接頭辞は，$rot(T)[i'][1,k_i] = \infty \circ rot(T)[i][1,k_i-1]$，$rot(T)[j'][1,k_i] = \infty \circ rot(T)[j][1,k_i-1]$ となるので，最長共通接頭辞の長さは k_i である．位置 k_i は，$rot(T)[i]$ 中で ∞ が $pBWT[i]$ 個目に出現した位置なので，$rank_{rot(T)[i]}(\infty,k_i) = pBWT(T)[i]$ であり，$rank_{rot(T)[i]}(\infty,k_i-1) = pBWT(T)[i]-1$ である．$rot(T)[i'][1,k_i] = \infty \circ rot(T)[i][1,k_i-1]$ であることと，$rank_{rot(T)[i]}(\infty,k_i-1) = pBWT(T)[i]-1$ であることから，$lcp_\infty(i',j') = rank_{rot(T)[i']}(\infty,k_i) = rank_{rot(T)[i]}(\infty,k_i-1)+1 = pBWT(T)[i]-1+1 = pBWT(T)[i]$ である．よって，$(lcp_\infty(i,j) \geq pBWT(T)[i])$ かつ $(pBWT(T)[j] > pBWT(T)[i])$ のときは，$lcp_\infty(i',j') = pBWT(T)[i]$ となる．

また，$(lcp_\infty(i,j) \geq pBWT(T)[j])$ かつ $(pBWT(T)[i] > pBWT(T)[j])$ のときは，$(lcp_\infty(i,j) \geq pBWT(T)[i])$ かつ $(pBWT(T)[j] > pBWT(T)[i])$ のときと同様のため，$lcp_\infty(i',j') = pBWT(T)[j]$ となる．

最後に，$(lcp_\infty(i,j) \geq pBWT(T)[i])$ かつ $(pBWT(T)[i] = pBWT(T)[j])$ のときを考える．$lcp_\infty(i,j) \geq pBWT(T)[i]$ のため $p > k_i$ であり，$pBWT(T)[i] = pBWT(T)[j]$ なので $k_i = k_j$ である．よって，$rot(T)[i'], rot(T)[j']$ は p, k_i, k_j を用いて以下のように表記で

きる．

$$rot(T)[i'] = \infty \circ rot(T)[i][1, k_i - 1] \circ k_i \circ rot(T)[i][k_i + 1, p] \circ rot(T)[i][p + 1, n - 1]$$
$$rot(T)[j'] = \infty \circ rot(T)[j][1, k_j - 1] \circ k_j \circ rot(T)[j][k_j + 1, p] \circ rot(T)[j][p + 1, n - 1]$$

したがって，$rot(T)[i']$ と $rot(T)[j']$ の最長共通接頭辞は $k_i = k_j$ より，$rot(T)[i'][1, p+1] = \infty \circ rot(T)[i][1, k_i - 1] \circ k_i \circ rot(T)[i][k_i + 1, p]$，$rot(T)[j'][1, p+1] = \infty \circ rot(T)[j][1, k_j - 1] \circ k_j \circ rot(T)[j][k_j + 1, p]$ となるので，最長共通接頭辞の長さは $p+1$ である．$rot(T)[i][k_i] = \infty$ かつ，$rot(T)[i'][k_i + 1] = k_i$ なので，$rot(T)[i'][2, p + 1]$ 中に出現する ∞ の個数は，$rot(T)[i][1, p]$ 中に出現する ∞ の個数よりも 1 小さくなる．したがって，$rank_{rot(T)[i']}(\infty, p+1) = (rank_{rot(T)[i]}(\infty, p) - 1) + 1 = lcp_\infty(i, j)$ である．よって，$(lcp_\infty(i, j) \geq pBWT(T)[i])$ かつ $(pBWT(T)[i] = pBWT(T)[j])$ のときは，$lcp_\infty(i', j') = lcp_\infty(i, j)$ である．

□

この定理を用い，$LCP_\infty(T)[t-1]$ の値を更新し，$LCP_\infty(T)[t]$ に新たな値を挿入する．

まず，$LCP_\infty(T)[t-1]$ の値を更新を考える．$LCP_\infty(T')[t-1]$ の値は $rot(T')[t-1], rot(T')[t]$ の最長共通接頭辞内の ∞ の個数である．これは，$rot(T')[t-1], rot(T')[t]$ の先頭文字を文末に巡回した行を $rot(T')[(t-1)^{-1}], rot(T')[t^{-1}]$ としたときに，行番号 $(t-1)^{-1}, t^{-1}$ を求め，$pBWT(T')[(t-1)^{-1}], pBWT(T')[t^{-1}], lcp_\infty((t-1)^{-1}, t^{-1})$ から，$lcp_\infty(t-1, t)$ を定理 5 を用いることで求められる．

以上の $LCP_\infty(T)$ の更新アルゴリズムを Algorithm 7 に記す．

定理 6. Algorithm 7 を用いることで，変数文字のアルファベットサイズを $|\Pi|$ とすると，配列 $LCP_\infty(T)$ の更新は $O(|\Pi| \frac{\log n}{\log \log n})$ 時間である．

証明. まず，配列 $LCP_\infty(T)$ 上で更新が生じる箇所は，Algorithm 6 で求めた $rot(T')[p] = \langle T' \rangle$ を満たす位置を p とすると，$LCP_\infty(T)[p-1, p]$ の 2 箇所である．

まず，$rot(T')[p-1] = \langle T'_l \rangle$，$rot(T')[p] = \langle T'_m \rangle$ としたときに，$rot(T')[p'_1] = \langle T'_{l-1} \rangle$，$rot(T')[p'_2] = \langle T'_{m-1} \rangle$ を満たす行 p'_1, p'_2 を見つける必要がある．行 p'_1, p'_2 は $rank$ 関数,$select$ 関数を用い，$p'_1 = select_{pBWT(T')}(F(T')[p-1], rank_{F(T')}(F(T')[p-1], p-1))$，$p'_2 =$

Algorithm 7: $LCP_\infty(T)$ の更新

input : 配列 $LCP_\infty(T)$，文字列 $pBWT(T'), F(T')$，Algorithm 6 での挿入位置 p
output: 配列 $LCP_\infty(T')$

```
for i = 0 to 1 do
    cou = rank_{F(T')}(F(T')[p − 1 + i], p − 1 + i);
    poj = select_{pBWT(T')}(F[p − 1 + i], cou);
    cou_2 = rank_F(F[p + i], p + i);
    poj_2 = select_{pBWT(T')}(F[p + i], cou_2);
    value = lcp_∞(poj, poj_2);
    r = 0;
    if pBWT(T')[poj] ∈ Σ or pBWT(T')[poj_2] ∈ Σ then
        if pBWT(T')[poj] = pBWT(T')[poj_2] then
            r = value;
    else
        if value < pBWT(T')[poj] and value < pBWT(T')[poj_2] then
            r = value + 1;
        else if value ≥ pBWT(T')[poj] and pBWT(T')[poj] < pBWT(T')[poj_2]
         then
            r = pBWT(T')[poj];
        else if value ≥ pBWT(T')[poj_2] and pBWT(T')[poj_2] < pBWT(T')[poj]
         then
            r = pBWT(T')[poj_2];
        else
            r = value;
    if i = 0 then
        LCP_∞(T)[p − 1 + i] = r;
    else
        insert_{LCP_∞(T)}(r, p);
```

$select_{pBWT(T')}(F(T')[p], rank_{F(T')}(F(T')[p], p))$ によって検索できるため，$O(\frac{\log n}{\log\log n})$ 時間で見つけられる．

その後，$lcp_\infty(p'_1, p'_2)$ を計算する必要があり，これは，$LCP_\infty(T)[p'_1, p'_2 - 1]$ の最小値である．$LCP_\infty(T)$ の値は，0 以上 $|\Pi|$ 以下であることから，0 から $|\Pi|$ まで順に確認することで，$LCP_\infty(T)[p'_1, p'_2 - 1]$ の最小値を求めることができる．自然数 k の $LCP_\infty(T)[p'_1, p'_2 - 1]$ 中の出現個数は，$rank$ 関数を用い以下式で求められる．

$$rank_{LCP_\infty(T)}(k, p'_2 - 1) - rank_{LCP_\infty(T)}(k, p'_1 - 1)$$

上記式の値が 1 以上となるとき，$LCP_\infty(T)[p'_1, p'_2 - 1]$ 中に自然数 k が出現していることとなる．したがって，$rank$ 関数を最大 $|\Pi|$ 回用いることで $LCP_\infty(T)[p'_1, p'_2 - 1]$ 中の最小値を求めることができるため，$O(|\Pi|\frac{\log n}{\log\log n})$ 時間で $lcp_\infty(p'_1, p'_2)$ を計算できる．

最後に，条件より当てはまる値を $LCP_\infty(T)[p-1]$ では置き換えを，$LCP_\infty(T)[p]$ では挿入を行うことで配列 $LCP_\infty(T)$ の更新ができる．具体的には，$delete_{LCP_\infty(T)}(p-1)$ 後，$insert_{LCP_\infty(T)}(lcp_\infty(p-1, p), p-1)$ の操作をすることで $LCP_\infty(T)[p-1]$ の置き換えができ，$insert_{LCP_\infty(T)}(lcp_\infty(p, p+1), p)$ の操作を行うことで，$LCP_\infty(T)[p]$ の挿入ができる．したがって，$insert$ と $delete$ の操作を定数回行うことで， $LCP_\infty(T)$ の更新ができるため，$O(\frac{\log n}{\log\log n})$ 時間かかる．

よって，配列 $LCP_\infty(T)$ の更新は，変数文字のサイズを $|\Pi|$ とすると $O(|\Pi|\frac{\log n}{\log\log n})$ 時間で行える． □

3.5 まとめ

文字を読み込む毎に Algorithm 3 〜 7 を行うことで，文字列 T の $F(T), pBWT(T)$ の構築を行うことができる．

定理 7. Algorithm 3 〜 7 を用いることで，変数文字のアルファベットサイズを $|\Pi|$ とすると，長さ n の文字列 T の $F(T), pBWT(T)$ のオンライン構築は，$O(n|\Pi|\frac{\log n}{\log\log n})$ 時間で行える．

第4章

まとめ

本論文では，パラメタ化 BW 変換のオンライン構築を $O(n|\Pi|\frac{\log n}{\log\log n})$ 時間で行う初めての手法について提案した．現状，文字列に対する操作に $O(\frac{\log n}{\log\log n})$ 時間がかかってしまっているため，手法全体の計算時間が $O(n|\Pi|\frac{\log n}{\log\log n})$ 時間となっているので，$rank$ 関数等の計算や $insert$ 等の操作が $O(1)$ 時間でできるようなデータ構造が構築されるとより効率的にオンライン構築が行えると考えられる．

また，今回のオンライン構築は 1 文字ずつの入力を想定して行なっているが，単語や特定の区切りでの入力ということも考えられ，その際により効率的な計算時間での構築というものも今後の課題として挙げられる．

BW 変換では，省スペースでの構築や，入力された文字列の領域を出力に変更しながら構築する in-place な手法などの研究 [3] も行われているので，パラメタ化 BW 変換の省スペースでの構築なども今後の課題として挙げられる．

参考文献

[1] Brenda S Baker. A theory of parameterized pattern matching: algorithms and applications. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pp. 71–80, 1993.

[2] Michael Burrows and David Wheeler. A Block-Sorting Lossless Data Compression Algorithm. Technical report, DIGITAL SRC RESEARCH REPORT, 1994.

[3] Maxime Crochemore, Roberto Grossi, JuhaKärkkäinen, Gad M. Landau. Computing the Burrows–Wheeler transform in place and in small space. *Journal of Discrete Algorithms*, Vol. 32, pp. 44–52, 2015. StringMasters 2012 & 2013 Special Issue (Volume 2).

[4] Paolo Ferragina and Giovanni Manzini. Opportunistic data structures with applications. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pp. 390–398. IEEE, 2000.

[5] Arnab Ganguly, Rahul Shah, and Sharma V Thankachan. pBWT: Achieving succinct data structures for parameterized pattern matching and related problems. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 397–407. SIAM, 2017.

[6] Wing-Kai Hon, K. Sadakane, and Wing-Kin Sung. Breaking a time-and-space barrier in constructing full-text indices. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pp. 251–260, 2003.

[7] Sung-Hwan Kim and Hwan-Gue Cho. Simpler FM-index for parameterized string matching. *Information Processing Letters*, Vol. 165, p. 106026, 2021.

[8] Gonzalo Navarro and Yakov Nekrich. Optimal dynamic sequence representations. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 865–876. SIAM, 2013.

[9] Tatsuya Ohno, Kensuke Sakai, Yoshimasa Takabatake, Tomohiro I, and Hiroshi Sakamoto. A faster implementation of online RLBWT and its application to LZ77 parsing. *Journal of Discrete Algorithms*, Vol. 52-53, pp. 18–28, 2018. Combinatorial Algorithms – Special Issue Devoted to Life and Work of Mirka Miller.

[10] Alberto Policriti and Nicola Prezza. Fast Online Lempel-Ziv Factorization in Compressed Space. In *Proceedings of the 22nd International Symposium on String Processing and Information Retrieval - Volume 9309*, SPIRE 2015, pp. 13–20, 2015.

謝辞

本論文を執筆するにあたり，多くのご指導を賜りました，東北大学大学院情報科学研究科 篠原 歩教授，吉仲 亮准教授，ならびに Diptarama Hendrian 助教に心より感謝申し上げます．先生方には，研究内容や論文執筆，発表内容など数多くの場面でご助言を賜りました．

また，本論文の副審査委員を務めていただきました，東北大学大学院情報科学研究科 張山 昌論教授，ならびに東北大学大学院情報科学研究科 伊藤 健洋教授には，貴重なご意見を賜わりましたことを心より感謝申し上げます．

研究室の同期や後輩の皆様には，発表内容などに関して様々な視点からの意見を頂き，発表をよりよくすることができました．感謝申し上げます．

最後に，大学生活を支えてくれた家族に心から感謝申し上げます．