準リアルタイム接尾辞木構築に関する応用について

January 11, 2026

Dominik Köppl [1]　Gregory Kucherov [2]

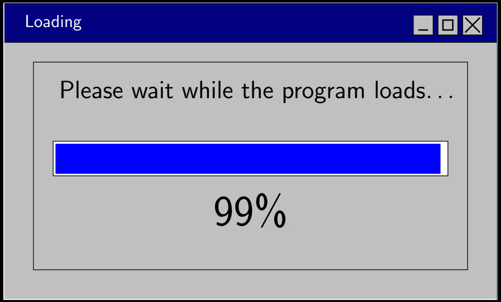
[1]: 山梨大学　コンピュータ理工学
[2]: ギュスターヴ・エッフェル大学

背景

- 古典的な問題：パターン照合，テキスト比較，データ圧縮など
- 大規模なテキスト集合に対して（例：DNA配列，リポジトリなど）

しかし：

- ほとんどのアルゴリズムはオフラインで動作：いかなる出力も生成される前に全体のテキストを必要とする
- 全入力が処理されるまで待つ必要がある！
- 実世界の現象：バーが99%に達したときにアルゴリズムが本格的に動き始め，ユーザーは待たされる . . .



オンラインアルゴリズム

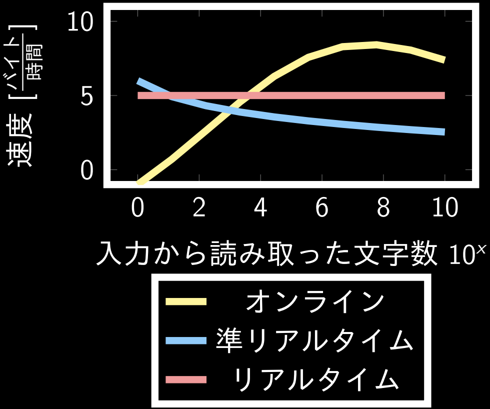
定義 (オンラインアルゴリズム)

新しい文字が到着するたびに，利用可能な解を漸進的に更新する

- ほとんどのオンラインアルゴリズムは平均化されている：1文字あたりの最悪時間は高くなる可能性がある！
- リアルタイムアルゴリズム：1文字あたりの最悪時間は定数

ここでは：

- 準リアルタイムアルゴリズム：1文字あたりの最悪時間は入力サイズに対して対数的



リアルタイムアルゴリズム

リアルタイムで何ができるか？

- スライディングウィンドウを用いたパターン照合 Galil'76
- 回文認識 Galil'76
- Lempel–Ziv 78分解 Hartman,Rodeh'85

準リアルタイムで何ができるか？

- 接尾辞木構築 Breslauer,Italiano'13
  Weinerのアルゴリズム Weiner'73に基づく

Weinerの接尾辞木構築

入力：

- 長さ n のテキスト $T[1..n]$ （整数アルファベット Σ 上）
- 文字は右から左へ到着する
- $T[n] = \$$ は一意の終端文字とする

出力： T の接尾辞木 $\mathsf{ST}(T)$

目標： $T[j]$ が到着したとき， $\mathsf{ST}(T[j+1..])$ から $\mathsf{ST}(T[j..])$ を構築する ($j = n, n-1, \ldots, 1$)

例 (T = aababaa\$)

Weinerの接尾辞木構築

入力：

- 長さ n のテキスト $T[1..n]$ （整数アルファベット Σ 上）
- 文字は右から左へ到着する
- $T[n] = \$$ は一意の終端文字とする

出力： T の接尾辞木 $\mathrm{ST}(T)$

目標： $T[j]$ が到着したとき， $\mathrm{ST}(T[j+1..])$ から $\mathrm{ST}(T[j..])$ を構築する
$(j = n, n-1, \ldots, 1)$

例 ($T = \mathtt{aababaa\$}$)

- $ から開始

ST($)

Weinerの接尾辞木構築

入力：

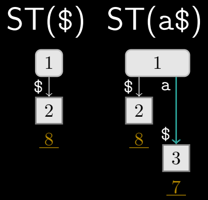
- 長さ n のテキスト $T[1..n]$ （整数アルファベット Σ 上）
- 文字は右から左へ到着する
- $T[n] = \$$ は一意の終端文字とする

出力： T の接尾辞木 $\mathrm{ST}(T)$

目標： $T[j]$ が到着したとき， $\mathrm{ST}(T[j+1..])$ から $\mathrm{ST}(T[j..])$ を構築する $(j = n, n-1, \ldots, 1)$

例 (T = aababaa\$)

- \$ から開始
- a\$ を挿入



Weinerの接尾辞木構築

入力：
- 長さ n のテキスト $T[1..n]$ （整数アルファベット Σ 上）
- 文字は右から左へ到着する
- $T[n] = \$$ は一意の終端文字とする

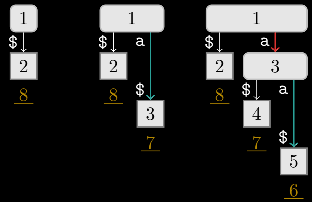
出力： T の接尾辞木 $\mathsf{ST}(T)$

目標： $T[j]$ が到着したとき， $\mathsf{ST}(T[j+1..])$ から $\mathsf{ST}(T[j..])$ を構築する
$(j = n, n-1, \ldots, 1)$

例 (T = aababaa\$)

- \$ から開始
- a\$ を挿入
- aa\$ を挿入

ST(\$) ST(a\$) ST(aa\$)



Weinerの接尾辞木構築

入力：
- 長さ n のテキスト $T[1..n]$ （整数アルファベット Σ 上）
- 文字は右から左へ到着する
- $T[n] = \$$ は一意の終端文字とする

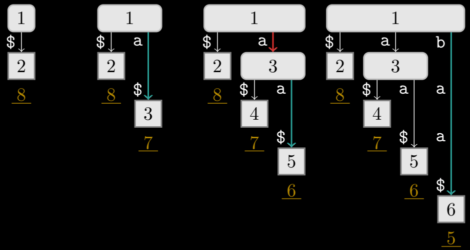
出力： T の接尾辞木 $\mathsf{ST}(T)$

目標： $T[j]$ が到着したとき， $\mathsf{ST}(T[j+1..])$ から $\mathsf{ST}(T[j..])$ を構築する
$(j = n, n-1, \ldots, 1)$

例 ($T = \texttt{aababaa\$}$)

- $ から開始
- a$ を挿入
- aa$ を挿入
- baa$ を挿入

ST($) ST(a$) ST(aa$) ST(baa$)

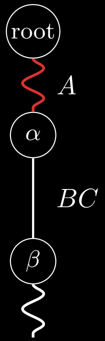


挿入点

$T[j..]$ を $\mathrm{ST}(T[j+1..])$ に挿入するとき， ST をどこで更新するか？

素朴な解法

1. $\mathrm{ST}(T[j+1..])$ で根から $T[j..]$ に一致する最長経路を検索する

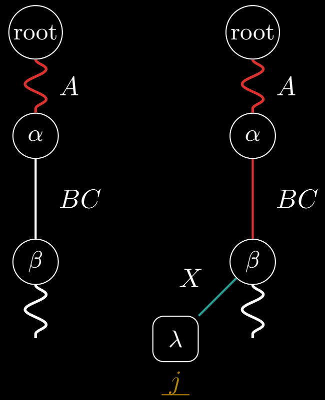


挿入点

$T[j..]$ を $\mathrm{ST}(T[j+1..])$ に挿入するとき，ST をどこで更新するか？

素朴な解法

1. $\mathrm{ST}(T[j+1..])$ で根から $T[j..]$ に一致する最長経路を検索する
2. $T[j..] = ABCX$ の場合，頂点 β で分岐する



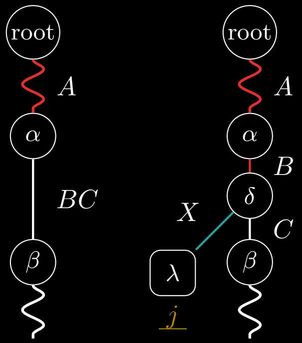
挿入点

$T[j..]$ を $\mathrm{ST}(T[j+1..])$ に挿入するとき， ST をどこで更新するか？

素朴な解法

1. $\mathrm{ST}(T[j+1..])$ で根から $T[j..]$ に一致する最長経路を検索する
2. $T[j..] = ABCX$ の場合，頂点 β で分岐する
3. $T[j..] = ABX$ の場合，
   3.1 辺 (α, β) を分割して頂点 δ を作成する
   3.2 頂点 δ で分岐する

いずれの場合も，分岐点を**挿入点**と呼ぶ．



挿入点問題

重要な観察

- 接尾辞 $T[j..]$ は $\mathrm{ST}(T[j..])$ において葉 λ で表される
- $\mathrm{ST}(T[j+1..])$ を $\mathrm{ST}(T[j..])$ に更新するには， λ を挿入する必要がある
- これには $\mathrm{ST}(T[j+1..])$ における $T[j..]$ の挿入点を見つける必要がある

問題 (InsertionPoint)

入力: 接尾辞木 $\mathrm{ST}(T[j+1..])$ と文字 $T[j]$.
出力: $T[j+1..]$ に出現する $T[j..]$ の最長接頭辞の位置.

t_{SU} をInsertionPointを解く時間計算量とする.

時間計算量 t_{SU}

素朴な手法：

- $\mathsf{ST}(T[j+1..])$ を根から $T[j..]$ に一致するまで走査する
- $O(n)$ 時間

より良い方法はあるか？

t_{SU}	参考文献
$O(\lg \sigma)$ 平均	Weiner'73
$O(\lg n)$	Amir,Kopelowitz,+'05
$O(\sigma \log\log n)$	Breslauer,Italiano'13
$O(\log\log n + \log\log \sigma)$ 期待	Kopelowitz'12
$O(\log\log n + \frac{\log^2 \log \sigma}{\log\log\log \sigma})$	Fischer,Gawrychowski'15
$O(\log\log n)$ ただし $\sigma = O(\log^{1/4} n)$	Kucherov,Nekrich'17

挿入点について

InsertionPointはどんな情報を保持するか？

- $T[j..n]$ と $\mathrm{ST}(T[j+1..n])$ 内の $T[j..n]$ に辞書順で最も近い接尾辞との最長共通接頭辞

最長繰り返し接頭辞問題を解くことができる！

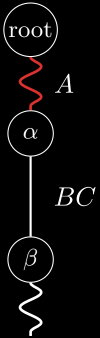
問題 (LongestRepeatingPrefix)

入力: 接尾辞木 $\mathrm{ST}(T[j+1..])$ と文字 $T[j]$.
出力: $T[j..]$ の最長繰り返し接頭辞の長さ.

InsertionPointを用いてLongestRepeatingPrefixを解く

アルゴリズム

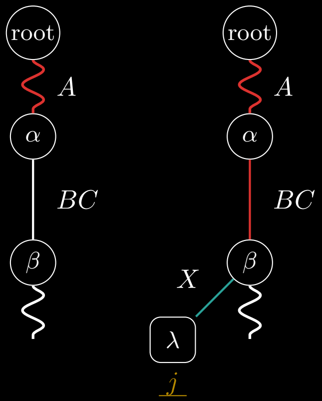
1. ST($T[j+1..]$) において $T[j..]$ のInsertionPointを計算する



InsertionPointを用いてLongestRepeatingPrefixを解く

アルゴリズム

1. ST($T[j+1..]$) において $T[j..]$ のInsertionPointを計算する
2. InsertionPointが β ならば $|ABC|$ を返す

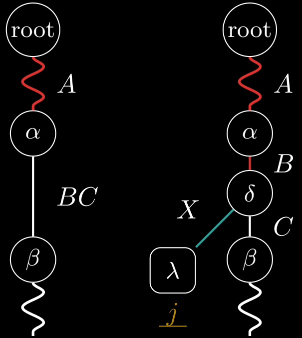


InsertionPointを用いてLongestRepeatingPrefixを解く

アルゴリズム

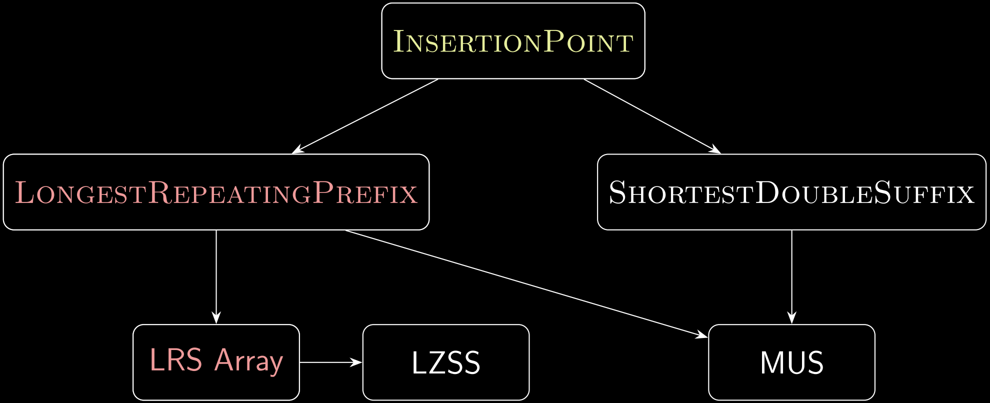
1. ST($T[j+1..]$) において $T[j..]$ のInsertionPointを計算する
2. InsertionPointが β ならば $|ABC|$ を返す
3. InsertionPointが δ ならば $|AB|$ を返す

いずれの場合も：InsertionPointの文字深さを出力する



他に何ができるか？

最長繰り返し接尾辞配列



最長繰り返し接尾辞配列

LongestRepeatingPrefixを使って計算できる：

定義 (最長繰り返し接尾辞配列 (LRS))

文字列 $T[1..n]$ の最長繰り返し接尾辞配列（LRS）は配列 $\mathrm{LRS}[1..n]$ であり，各位置 $j \in [1..n]$ に対して， $\mathrm{LRS}[j]$ は $T[1..j]$ の中で少なくとも2回出現する最長の接尾辞の長さである.

LRS をオンラインで計算する既知の解法，1文字あたりの時間

時間	参考文献
$O(\log^3 n)$	Okanohara,Sadakane'08
$O(\log^2 n)$ 平均	Prezza,Rosone'20
$O(t_{\mathrm{SU}})$	この研究

準リアルタイムLRS計算

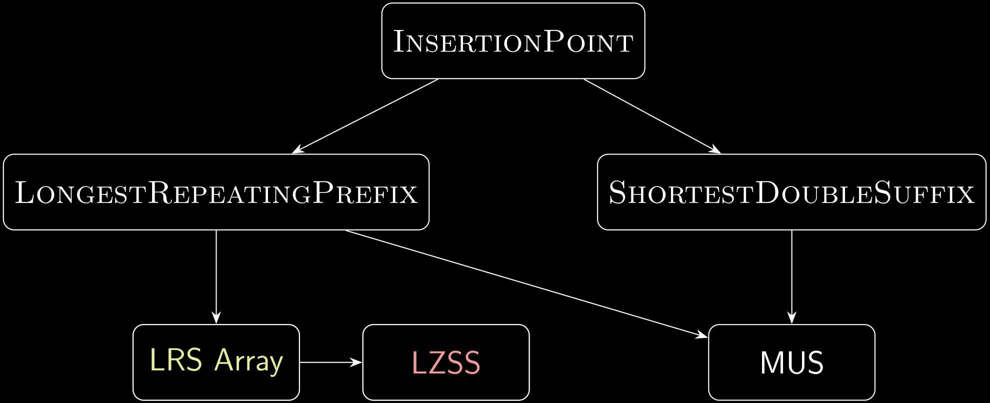
Amir,Landau,+'02の考え方に従う：

- 反転テキスト $\overleftarrow{T} = T[n]T[n-1]\cdots T[1]$ 上で ST をオンラインで計算する
- LongestRepeatingPrefixクエリを使って LRS を計算する

定理

LRS を1文字あたり $O(t_{\mathrm{SU}})$ 最悪時間でオンライン計算できる.

LZSS計算


INSERTIONPOINT
LONGESTREPEATINGPREFIX
SHORTESTDOUBLESUFFIX
LRS Array
LZSS
MUS

Lempel–Ziv–Storer–Szymanski (LZSS)

$$T = \texttt{a a b a b a a \$}$$

1 2 3 4 5 6 7 8

符号化：

- 最長の読み取り可能な部分文字列を後方参照で置き換える

Lempel–Ziv–Storer–Szymanski (LZSS)

$T = $ a a b a b a a \$

a	a	b	a	b	a	a	\$
1	2	3	4	5	6	7	8

符号化： a

- 最長の読み取り可能な部分文字列を後方参照で置き換える
- 単一の文字から開始する（置き換えなし）

Lempel–Ziv–Storer–Szymanski (LZSS)

$$T = \texttt{a}\,\texttt{a}\,\texttt{b}\,\texttt{a}\,\texttt{b}\,\texttt{a}\,\texttt{a}\,\$$$

a	a	b	a	b	a	a	$
1	2	3	4	5	6	7	8

(1,1)

符号化： a(1,1)

- 最長の読み取り可能な部分文字列を後方参照で置き換える
- 単一の文字から開始する（置き換えなし）
- 最長の以前の出現で置き換える（重複は許可される）

Lempel–Ziv–Storer–Szymanski (LZSS)

$(1,1)$

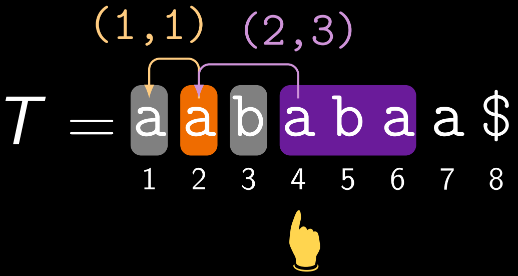
$T = \mathtt{a\,a\,b\,a\,b\,a\,a\,\$}$

1 2 3 4 5 6 7 8

符号化： $\mathtt{a(1,1)b}$

- 最長の読み取り可能な部分文字列を後方参照で置き換える
- 単一の文字から開始する（置き換えなし）
- 最長の以前の出現で置き換える（重複は許可される）

Lempel–Ziv–Storer–Szymanski (LZSS)



符号化： a(1,1)b(2,3)

- 最長の読み取り可能な部分文字列を後方参照で置き換える
- 単一の文字から開始する（置き換えなし）
- 最長の以前の出現で置き換える（重複は許可される）

Lempel–Ziv–Storer–Szymanski (LZSS)

(1,1) (2,3) (1,1)

$T =$ a a b a b a a \$

1 2 3 4 5 6 7 8

符号化： a(1,1)b(2,3)(1,1)

- 最長の読み取り可能な部分文字列を後方参照で置き換える
- 単一の文字から開始する（置き換えなし）
- 最長の以前の出現で置き換える（重複は許可される）

Lempel–Ziv–Storer–Szymanski (LZSS)

(1,1) (2,3) (1,1)

$T =$ a a b a b a a $

1 2 3 4 5 6 7 8

符号化： a(1,1)b(2,3)(1,1)$

- 最長の読み取り可能な部分文字列を後方参照で置き換える
- 単一の文字から開始する（置き換えなし）
- 最長の以前の出現で置き換える（重複は許可される）

LZSS計算

LZSSをオンラインで計算する既知の解法，1文字あたりの時間：

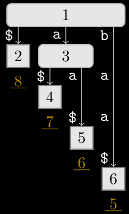
時間	参考文献
$O(\log \sigma)$ 平均	Gusfield'97
$O(t_{\mathrm{SU}})$	この研究
$O(\log^3 n)$	Okanohara,Sadakane'08
$O(\log^2 n)$ 平均	Starikovskaya'12
$O(\log n)$ 平均	Yamamoto+'14

※ 後者の解法はコンパクトな空間を使用する

準リアルタイムLZSS計算

- 仮定：$j = 3$，すなわち $T = \mathrm{aababaa}$ の $T[1..3] = \mathrm{aab}$ を処理済み
- $\mathrm{ST}(\overleftarrow{T[1..3]}) = \mathrm{ST}(\mathrm{baa})$ を持つ
- 次の要素 F が $T[4..] = \mathrm{abaa}$ から始まることを知っている

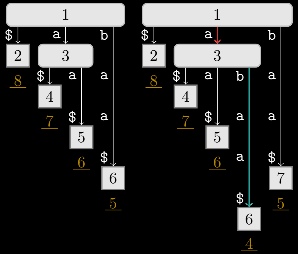
目標：$p = 4$ から始まる F の長さを計算する



準リアルタイムLZSS計算

目標： $p = 4$ から始まる F の長さを計算する

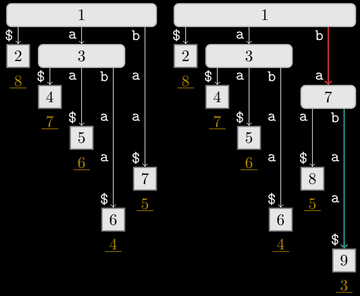
- $j \leftarrow 4$ を $\mathrm{ST}(\overleftarrow{T}[1..j]) = \mathrm{ST}(\mathtt{abaa})$ に更新
- 挿入点の位置は $L = \mathtt{a}$
- $|L| \geq p + 1 - j = 1$ のため：続行



準リアルタイムLZSS計算

目標： $p = 4$ から始まる F の長さを計算する

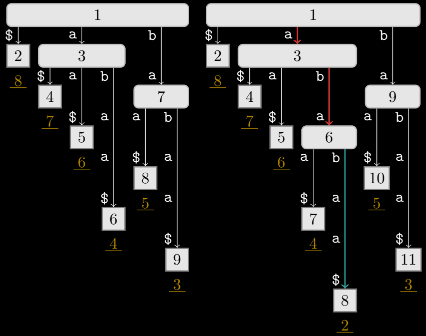
- $j \leftarrow 5$ を $\mathrm{ST}(\overleftarrow{T[1..j]}) = \mathrm{ST}(\mathtt{babaa})$ に更新
- 挿入点の位置は $L = \mathtt{ba}$
- $|L| \geq p + 1 - j = 2$ のため：続行



準リアルタイムLZSS計算

目標： $p = 4$ から始まる F の長さを計算する

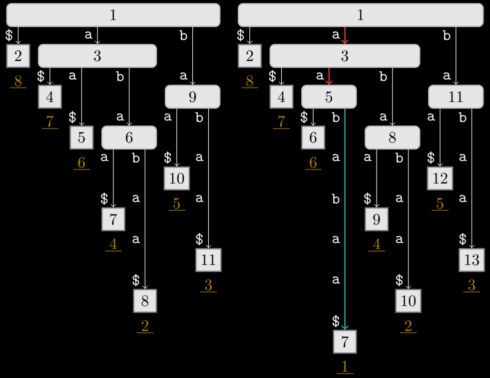
- $j \leftarrow 6$ を $\overleftarrow{\mathrm{ST}(T[1..j])} = \mathrm{ST}(\mathtt{ababaa})$ に更新
- 挿入点の位置は $L = \mathtt{aba}$
- $|L| \geq p + 1 - j = 3$ のため：続行



準リアルタイムLZSS計算

目標： $p = 4$ から始まる F の長さを計算する

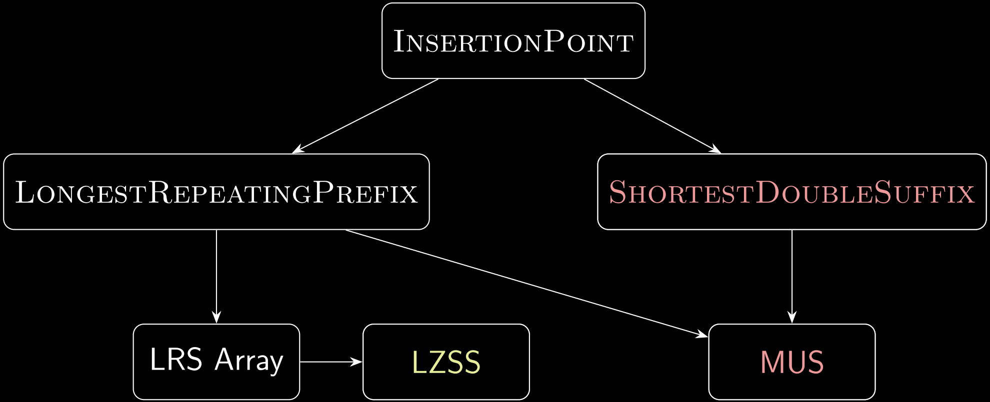
- $j \leftarrow 7$ を $\mathrm{ST}(\overleftarrow{T[1..j]}) = \mathrm{ST}(\mathrm{aababaa})$ に更新
- 挿入点の位置は $L = \mathrm{aa}$
- $|L| < p + 1 - j = 4$ のため：停止
- これで分かる： $|F| = p - j = 3$



LZSS計算（続き）

定理

T の LZSS 分解を1文字あたり $O(t_{\mathrm{SU}})$ 最悪時間でオンライン計算できる.



最小一意部分文字列 (MUS)

LongestRepeatingPrefixを使って計算できる：

定義 (最小一意部分文字列 (MUS))

$T[1..n]$ の最小一意部分文字列は部分文字列 $T[\ell..r]$ であり，次を満たす

- $T[\ell..r]$ は T において正確に1回出現
- $T[\ell+1..r]$ と $T[\ell..r-1]$ は T で少なくとも2回出現（$\ell < r$ の場合）

MUS($T[1..j]$) を $T[1..j]$ のすべての MUS の集合とする.

例 ($T[1..4] = \mathtt{aaba}$は MUS)

- $T[1..4] = \mathtt{aaba}$ は1回出現
- $T[1..3] = \mathtt{aab}$ は2回出現
- $T[2..4] = \mathtt{aba}$ は2回出現

$$\begin{array}{cccccccccc} & & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ T & = & \mathtt{a} & \mathtt{a} & \mathtt{b} & \mathtt{a} & \mathtt{b} & \mathtt{a} & \mathtt{a} & \mathtt{b} \end{array}$$

$\mathrm{MUS}(T) = \{\mathtt{abaa}, \mathtt{bab}, \mathtt{baa}\}$

既存の研究

- MUS をオンラインで計算する既知の解法，1文字あたりの時間

時間	参考文献
$O(\lg \sigma)$ 平均	Mieno+'20
$O(t_{SU})$	この研究

拡張可能配列

n 個の要素を持つ整数配列 $A[1..n]$ であり，以下をサポートする

- $A.\mathrm{grow}()$ ： A のサイズを増加させる
- $A[j]$ ：エントリ $A[j]$ へのランダム読み書きアクセス

拡張可能配列：

時間	参考文献
$O(1)$ 平均	配列倍増
$O(1)$	Brodnik+'99

Mieno+'20の手法

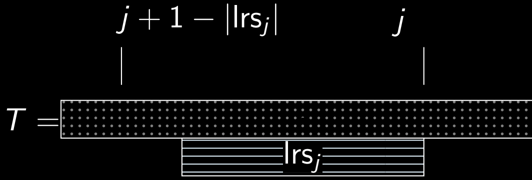
MUS($T[1..j]$) は MUS($T[1..j-1]$)， lrs_j， sds_j から計算できることを示した

定義 (最短2重接尾辞 sds_j)

$T[1..j]$ において正確に2回出現する最短の接尾辞であり，存在しない場合もある．

オンライン MUS アルゴリズム

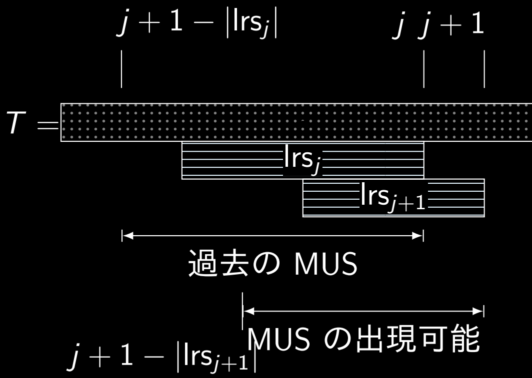
1. 新しい文字 $T[j+1]$ を読み込む



オンライン MUS アルゴリズム

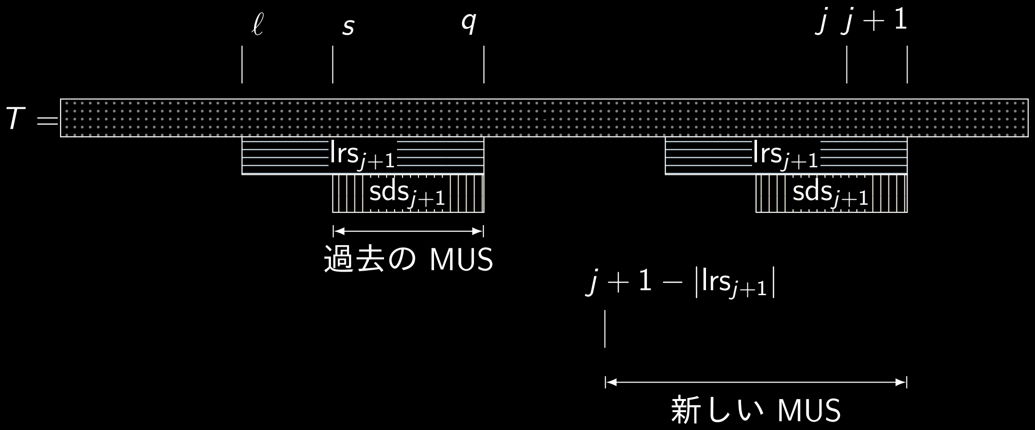
1. 新しい文字 $T[j+1]$ を読み込む
2. $|\mathsf{lrs}_{j+1}| \le \mathsf{lrs}_j|$ の場合，新しい MUS $[j+1-|\mathsf{lrs}_{j+1}|..j+1]$ を追加

- sds_{j+1} が存在しない場合，完了
- ここから： sds_{j+1} が存在すると仮定する



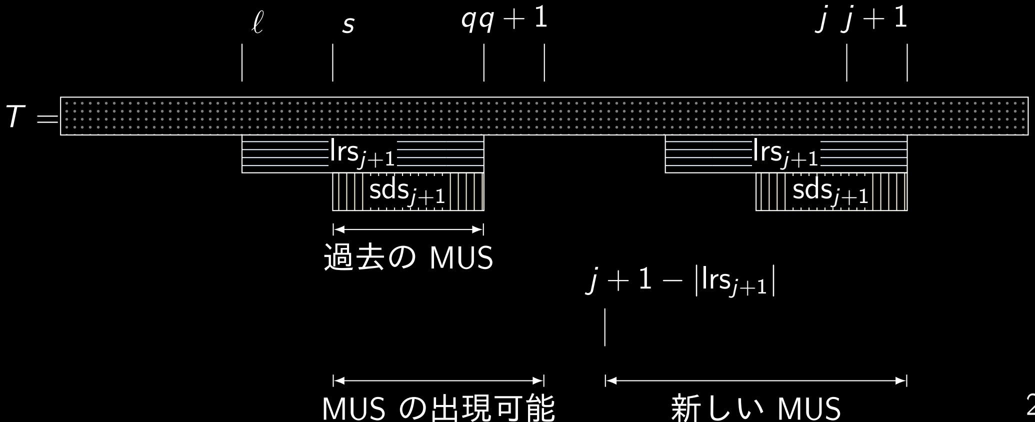
場合：sds_{j+1} が存在する

- sds_{j+1} の非接尾辞出現 $[s..q]$ を見つける：
  - s ： $\mathsf{ST}(\overleftarrow{T[1..j]})$ で既に存在していた sds_{j+1} の位置の葉のラベル
  - $q = s + |\mathsf{sds}_{j+1}| - 1$



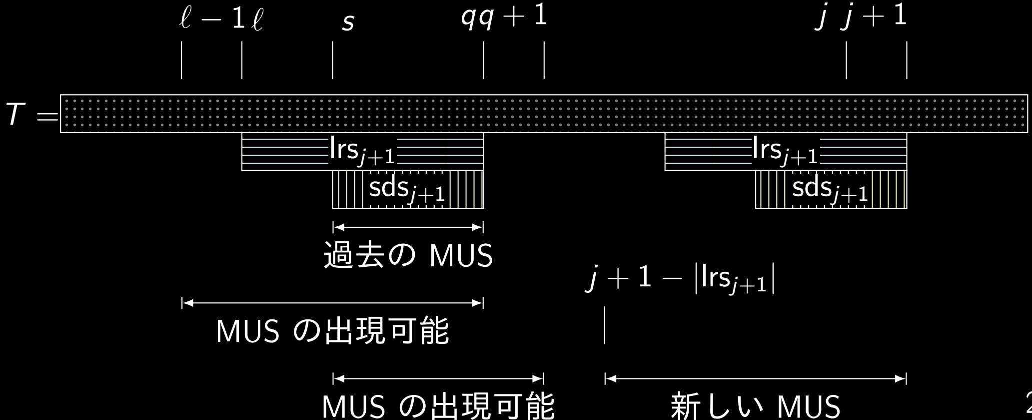
場合：sds_{j+1} が存在する

- 以前の MUS $[s..q]$ を削除
- 位置 $q+1$ で終わる MUS が存在しない場合：MUS $[s..q+1]$ を追加



場合：sds_{j+1} が存在する

- 以前の MUS $[s..q]$ を削除
- 位置 $q+1$ で終わる MUS が存在しない場合：MUS $[s..q+1]$ を追加
- $q - |\mathrm{lrs}_{j+1}| \geq 1$ かつ位置 $q - |\mathrm{lrs}_{j+1}|$ で始まる MUS がない場合：MUS $[q - |\mathrm{lrs}_{j+1}|..q]$ を追加



InsertionPoint からの帰着

問題 (ShortestDoubleSuffix sds_j)

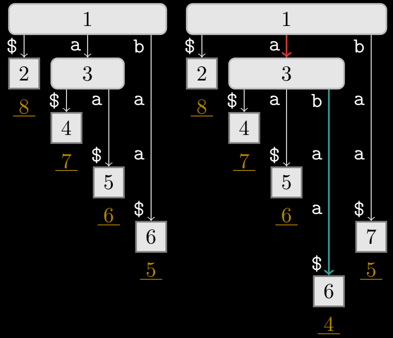
入力: 接尾辞木 $\mathrm{ST}(\overleftarrow{T[..j]})$ と文字 $T[j+1]$.
出力: $\overleftarrow{T[1..j]}$ において正確に1回の出現を持つ $\overleftarrow{T[..j+1]}$ の最短接頭辞（存在する場合).

ShortestDoubleSuffix は InsertionPoint を用いて解くことができる！

InsertionPoint からの帰着

アルゴリズム

もし挿入点が ST($T[j+1..]$) で既に存在する頂点である場合：
LongestRepeatingPrefix は $T[j..]$ において少なくとも3回出現し， sds_j は存在しない



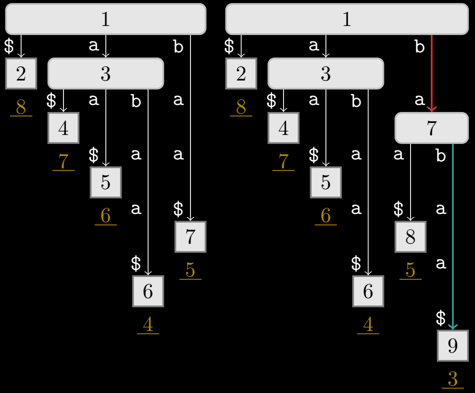
- T = aababaa
- ST(baa) から ST(abaa) への更新

InsertionPoint からの帰着

アルゴリズム

そうでない場合：

- LongestRepeatingPrefixは ST($T[j..]$) において正確に2つの葉を持つ頂点になる
- sds_j が存在し，その位置はLongestRepeatingPrefixの位置への親辺上にある



- T = aababaa
- ST(abaa) から ST(babaa) への更新

結論

定理

$\mathrm{MUS}(T[1..j])$ を1文字あたり $O(t_{\mathrm{SU}})$ 最悪時間でオンライン計算できる.

まとめ

- リアルタイム ST 構築は未解決だが，準リアルタイムで可能
- 多くの応用（LRS，LZSS，MUS，...）はInsertionPointを用いて準リアルタイムで解ける
- 示していない：反転LZ分解の準リアルタイム計算

ご清聴ありがとうございました