LZD と LZMW 分解の部分文字列圧縮について

クップル・ドミニク

山梨大学 大学院　総合研究部 工学域電気電子情報工学系 コンピュータ理工学

文字列の分解

- 入力： n の長さを持つ文字列 T
- 出力： T の分解

分解の例

- LZ77
- LZ78
- Lyndon 分解

目的： $\mathcal{O}(n)$ 時間で出力を計算

今回の分解

LZ78 から生じる分解を研究する

- Lempel–Ziv Double (LZD) Goto'15
- Lempel–Ziv-Miller–Wegman (LZMW) Miller+'85 (割愛)

なぜ？

- LZ78 項の個数は $\Omega(\sqrt{n})$ との下界を持つ
- その一方で、LZD の下界は $\Omega(\lg n)$

LZ78 の下界

$$T = \mathtt{a\ a\ a\ a\ a\ a\ a\ a\ a\ a\ a\ a} \cdots$$

T	a	a	a	a	a	a	a	a	a	a	a	a ⋯
	1	2	3	4	5	6	7	8	9	10	11	12

符号化：

x 番目の項 F_x の長さ $|F_x|$ は x になるため、 $\sum_{x=1}^{z} |F_x| = n \Leftrightarrow z \in \Theta(\sqrt{n})$

LZ78 の下界

$T = \mathtt{a\ a\ a\ a\ a\ a\ a\ a\ a\ a\ a\ a} \cdots$

1 2 3 4 5 6 7 8 9 10 11 12

符号化：a

x 番目の項 F_x の長さ $|F_x|$ は x になるため、 $\sum_{x=1}^{z} |F_x| = n \Leftrightarrow z \in \Theta(\sqrt{n})$

LZ78 の下界

(1,a)

$T = $ a a a a a a a a a a a a $\cdots$

1 2 3 4 5 6 7 8 9 10 11 12

符号化：a(1,a)

x 番目の項 F_x の長さ $|F_x|$ は x になるため、 $\sum_{x=1}^{z} |F_x| = n \Leftrightarrow z \in \Theta(\sqrt{n})$

LZ78 の下界

(1,a)

(2,a)

$T = \mathtt{a}\ \mathtt{a}\ \mathtt{a}\ \mathtt{a}\ \mathtt{a}\ \mathtt{a}\ \mathtt{a}\ \mathtt{a}\ \mathtt{a}\ \mathtt{a}\ \mathtt{a}\ \mathtt{a} \cdots$

1 2 3 4 5 6 7 8 9 10 11 12

符号化：a(1,a)(2,a)

x 番目の項 F_x の長さ $|F_x|$ は x になるため、 $\sum_{x=1}^{z} |F_x| = n \Leftrightarrow z \in \Theta(\sqrt{n})$

LZ78 の下界

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

$T = \mathtt{a}\ \mathtt{a}\ \mathtt{a}\ \mathtt{a}\ \mathtt{a}\ \mathtt{a}\ \mathtt{a}\ \mathtt{a}\ \mathtt{a}\ \mathtt{a}\ \mathtt{a}\ \mathtt{a} \cdots$

1 2 3 4 5 6 7 8 9 10 11 12

符号化：a(1,a)(2,a)(3,a)

x 番目の項 F_x の長さ $|F_x|$ は x になるため、 $\sum_{x=1}^{z} |F_x| = n \Leftrightarrow z \in \Theta(\sqrt{n})$

LZ78 の下界

(1,a) (2,a) (3,a) (4,a)

$T =$ a a a a a a a a a a a a ···

1 2 3 4 5 6 7 8 9 10 11 12

符号化：a(1,a)(2,a)(3,a)(4,a)

x 番目の項 F_x の長さ $|F_x|$ は x になるため、 $\sum_{x=1}^{z} |F_x| = n \Leftrightarrow z \in \Theta(\sqrt{n})$

LZD の下界

$$T = \mathtt{a\ a\ a\ a\ a\ a\ a\ a\ a\ a\ a\ a} \cdots$$

1 2 3 4 5 6 7 8 9 10 11 12

符号化：

x 番目の項 F_x の長さ $|F_x|$ は 2^x になるため、 $\sum_{x=1}^{z} |F_x| = n \Leftrightarrow z \in \Theta(\lg n)$

LZD の下界

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

1 2 3 4 5 6 7 8 9 10 11 12

符号化：(a,a)

x 番目の項 F_x の長さ $|F_x|$ は 2^x になるため、$\sum_{x=1}^{z} |F_x| = n \Leftrightarrow z \in \Theta(\lg n)$

LZD の下界

(1,1)

$T =$	a	a	a	a	a	a	a	a	a	a	a	a	$\cdots$
	1	2	3	4	5	6	7	8	9	10	11	12	

符号化：(a,a)(1,1)

x 番目の項 F_x の長さ $|F_x|$ は 2^x になるため、 $\sum_{x=1}^{z} |F_x| = n \Leftrightarrow z \in \Theta(\lg n)$

LZD の下界

(1,1)

(2,2)

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

1 2 3 4 5 6 7 8 9 10 11 12

符号化：(a,a)(1,1)(2,2)

x 番目の項 F_x の長さ $|F_x|$ は 2^x になるため、 $\sum_{x=1}^{z} |F_x| = n \Leftrightarrow z \in \Theta(\lg n)$

LZD の形式の定義

項は二組

- 前半と後半は文字または参照先を格納できる
- 前半の項のため、LZ78 のように最長の参照先を選んだあと、後半の計算を続ける

これから dst_x は項 F_x の開始位置を示す.

LZD の定義

$T[1..n] = F_1 \cdots F_z$ を $F_1, \ldots, F_z$ の項に分解する LZD は，以下の状況を満たす.

各 $x \in [1..z]$ に対して $F_x = G_1 \cdot G_2$，ただし

- $G_1, G_2 \in \{F_1, \ldots, F_{x-1}\} \cup \Sigma$ かつ
- G_1 と G_2 は，それぞれに $T[\mathrm{dst}_x..]$ と $T[\mathrm{dst}_x + |G_1|..]$ の最長接頭辞である.

LZD の例

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

1 2 3 4 5 6 7 8 9 10 11 12

符号化：

LZD の例

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

1 2 3 4 5 6 7 8 9 10 11 12

符号化：(a,b)

LZD の例

(1,b)

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

1 2 3 4 5 6 7 8 9 10 11 12

符号化：(a,b)(1,b)

LZD の例

(1,b)

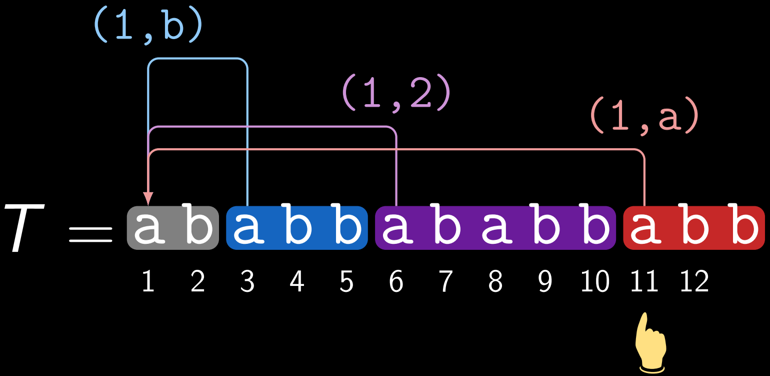
(1,2)

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

1 2 3 4 5 6 7 8 9 10 11 12

符号化：(a,b)(1,b)(1,2)

LZD の例



符号化：(a,b)(1,b)(1,2)(1,a)

LZD の計算

時間	領域	引用
$\mathcal{O}(n \lg \sigma)$	$\mathcal{O}(n)$	Goto+'15
$\Omega(n^{5/4})$	$\mathcal{O}(z)$	Goto+'15, Badkobeh+'17
$\mathcal{O}(n + z \lg^2 n)$ 期待	$\mathcal{O}(z)$	Badkobeh+'17
$\mathcal{O}(n)$	$\mathcal{O}(n)$	今回の発表

ただし

- Goto+'15 のアルゴリズムは LZD のみ計算する
- $\sigma = n^{\mathcal{O}(1)}$ は整数アルファベットのサイズを示す

今回の発表

研究の寄与は

- 全体のテキストに対して、LZD を決定的線形時間で計算

ただし

- 整数アルファベットに対応できる

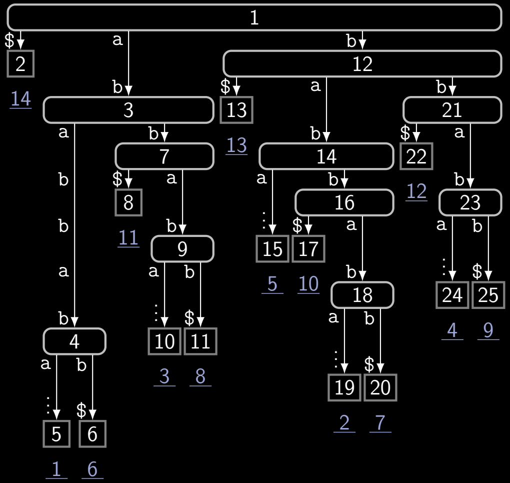
道具

計算のため、以下のデータ構造を利用

- 接尾辞木 ST Weiner'73
  - 線形時間で構築可能 Farach-Colton'00
- 重み付き祖先 (weighted ancestor) データ構造 Gawrychowski'14
  - 任意の ST 葉ノードの任意の文字列深さ d を持つ先祖を $\mathcal{O}(1)$ 時間で検索
  - 線形時間で構築可能 Belazzougui'21
- 最深マークされた先祖 (lowest marked ancestor) データ構造 Cole+'05
  - 任意の ST ノードを $\mathcal{O}(1)$ 時間でマーク
  - 任意の ST 葉ノードの最深マークされた先祖を $\mathcal{O}(1)$ 時間で検索

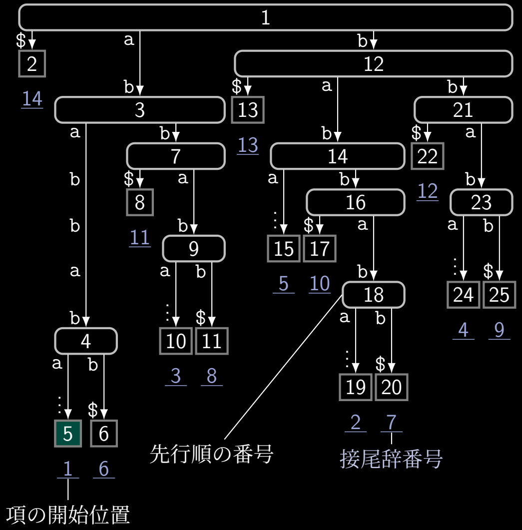
すべてのデータ構造の領域は $\mathcal{O}(n)$ となる

$T\$$ = ababbababbabb の接尾辞木

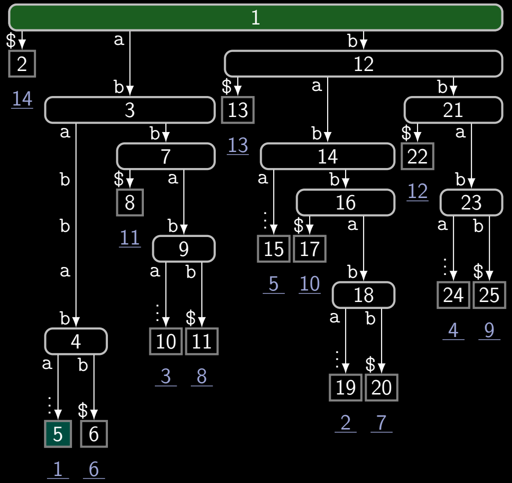


T = ababbababbabb

$T\$$ = ababbababbabb の接尾辞木



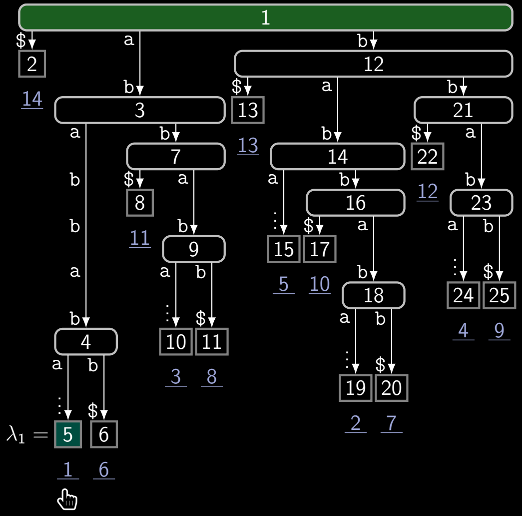
T = ababbababbabb



$T\$ = $ ababbababbabb の接尾辞木

- ST の根は空文字の参照先を表現する

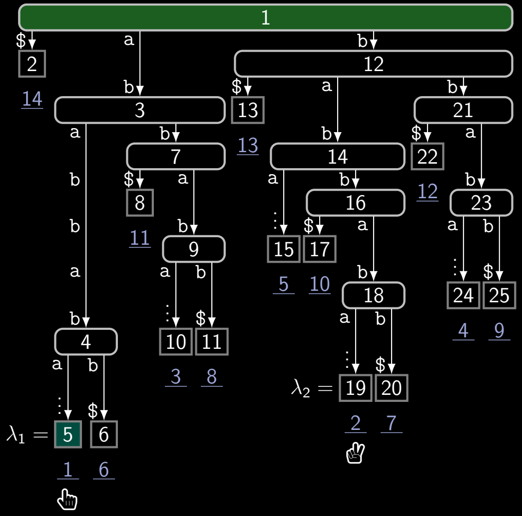
$T = $ ababbababbabb



$T = $ ababbababbabb

$T\$ = $ ababbababbabb の接尾辞木

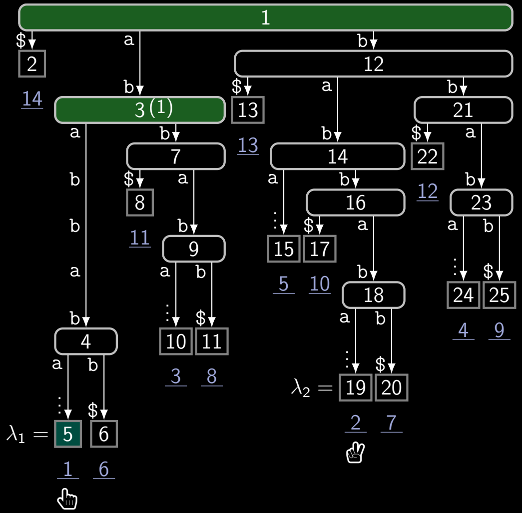
- ST の根は空文字の参照先を表現する
- 最初の項 $F_1 = (e_\mathrm{L}, e_\mathrm{R})$ の 2 組の計算
- λ_1 は項の開始位置を接尾辞番号として持つ葉ノードを示す
- λ_1 の最深マークされた先祖は根だから、$e_\mathrm{L} = T[1] = \mathrm{a}$



$T = \mathtt{ababbababbabb}$

$T\$ = \mathtt{ababbababbabb}$ の接尾辞木

- ST の根は空文字の参照先を表現する
- 最初の項 $F_1 = (e_\mathrm{L}, e_\mathrm{R})$ の 2 組の計算
- λ_1 は項の開始位置を接尾辞番号として持つ葉ノードを示す
- λ_1 の最深マークされた先祖は根だから、$e_\mathrm{L} = T[1] = \mathtt{a}$
- λ_2 は接尾辞番号 2 を持つ葉ノードを示す

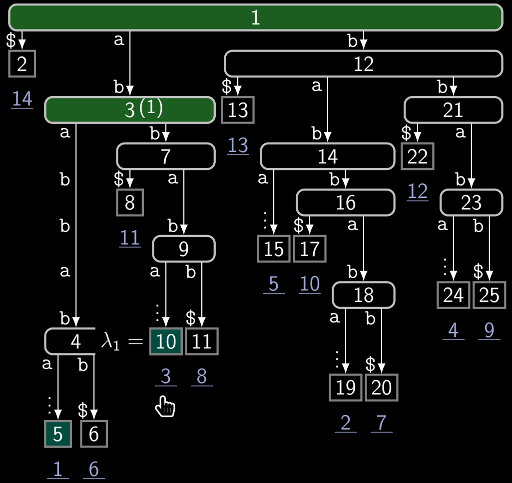


$T = \mathtt{ab}|\mathtt{abbababbabb}$

$T\$ = \mathtt{ababbababbabb}$ の接尾辞木

- ST の根は空文字の参照先を表現する
- 最初の項 $F_1 = (e_\mathrm{L}, e_\mathrm{R})$ の 2 組の計算
- λ_1 は項の開始位置を接尾辞番号として持つ葉ノードを示す
- λ_1 の最深マークされた先祖は根だから、$e_\mathrm{L} = T[1] = \mathtt{a}$
- λ_2 は接尾辞番号 2 を持つ葉ノードを示す
- λ_2 の最深マークされた先祖は根だから、$e_\mathrm{R} = T[2] = \mathtt{b}$
- 文字列長さ $|F_1| = 2$ を持つ λ_1 の先祖を 1 でマークする

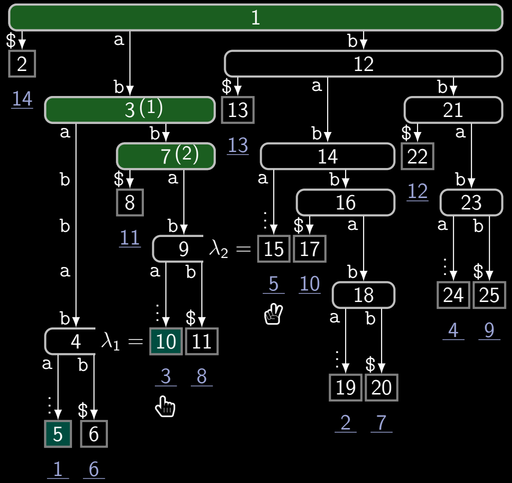
$T\$ = $ ababbababbabb の接尾辞木



F_2 の計算

- λ_1 は F_2 の開始位置を接尾辞番号として持つ葉を示す
- λ_1 の最深マークされた先祖は 3 ので、$e_L = 1$ (3 のマーク)

$T = $ ab|abbababbabb



$T\$ = $ ababbababbabb の接尾辞木

F_2 の計算

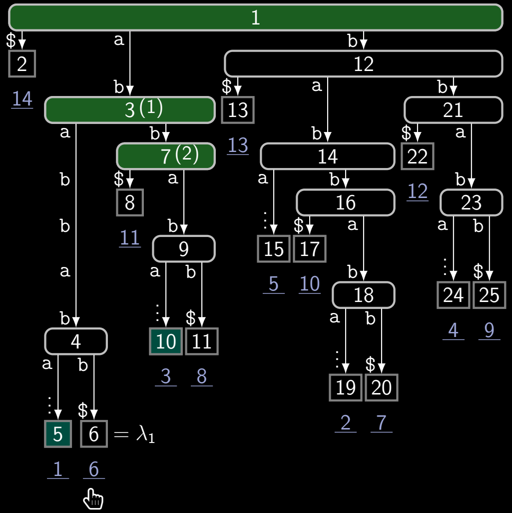
- λ_1 は F_2 の開始位置を接尾辞番号として持つ葉を示す
- λ_1 の最深マークされた先祖は 3 ので、$e_L = 1$ (3 のマーク)
- 前述のように、$e_R = T[2] = $ b
- 文字列長さ $|F_2| = 3$ を持つ λ_1 の先祖を 2 でマークする

$T = $ ab|abb|ababbabb

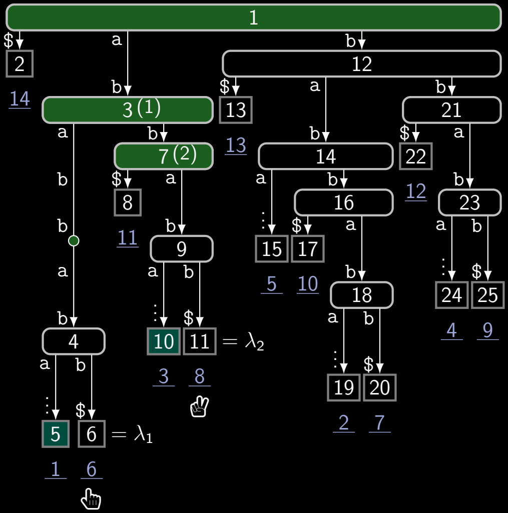
$T\$ = \texttt{ababbababbabb}$ の接尾辞木

F_3 の計算

- λ_1 は F_3 の開始位置を接尾辞番号として持つ葉を示す
- λ_1 の最深マークされた先祖は 3 ので、$e_L = 1$ (3 のマーク)



$T = \texttt{ab}|\texttt{abb}|\texttt{ababbabb}$

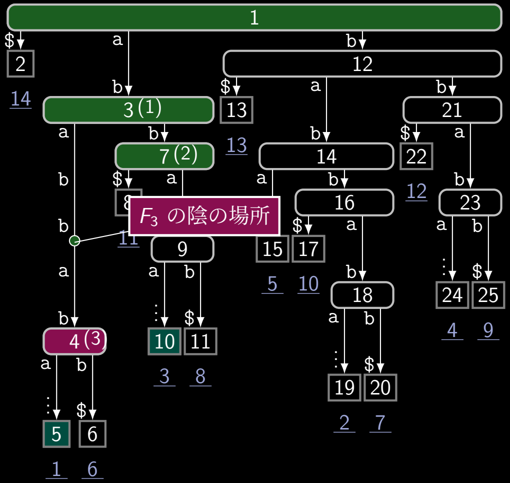


$T = \mathrm{ab}|\mathrm{abb}|\mathrm{ababb}|\mathrm{abb}$

$T\$ = \mathrm{ababbababbabb}$ の接尾辞木

F_3 の計算

- λ_1 は F_3 の開始位置を接尾辞番号として持つ葉を示す
- λ_1 の最深マークされた先祖は 3 ので、$e_L = 1$ (3 のマーク)
- λ_1 の最深マークされた先祖は 7 ので、$e_L = 2$ (2 のマーク)
- 文字列長さ $|F_3| = 5$ を持つ λ_1 の先祖がない！



$T\$ = \mathtt{ababbababbabb}$ の接尾辞木

F_3 を参照先として格納

- F_3 の場所はノード 4 で目撃される
- ノード 4 で F_3 の長さを格納し、マークする

$T = \mathtt{ab}|\mathtt{abb}|\mathtt{ababb}|\mathtt{abb}$

計算量

各項 F_x の計算のため

- F_x の開始位置 dst_x を接尾辞番号として持つ葉ノード λ_1 を取り
- λ_1 の最深マークされた先祖 v_1 を計算し
- v_1 の文字列深さは ℓ_1 とすると、接尾辞番号 $\mathsf{dst}_x + \ell_1$ を接尾辞番号として持つ葉ノード λ_2 を取り
- λ_2 の最深マークされた先祖 v_2 を計算し
- F_x の長さは $\ell_1 + \ell_2$ になる
- もし v_1 (または v_2) は陰の項を目撃すれば、ℓ_1 の変わりに格納された長さを利用する

各演算は定数時間で行い、全ての演算は $\mathcal{O}(z)$ 時間で走らせる、ただし z は項を個数をしめす

まとめ

- LZD を $\mathcal{O}(n)$ 時間で計算できる

ただし

- n は入力文字列の長さ
- 計算モデル
  - 整数アルファベット
  - ワード RAM

割愛した結果：部分文字列圧縮問題

- $\mathcal{O}(n)$ 時間の前処理で、LZD・LZMW の部分文字列圧縮問題を $\mathcal{O}(z)$ 時間で解ける

ご清聴ありがとうございました