接尾辞木に基づくLZ77とLPF配列の変種の計算

クップル　ドミニク

概 要

重複なし Lempel–Ziv 77 (LZ77) 分解と重複なし逆 Lempel–Ziv 分解 [Kolpakov and Kucherov, TCS ’09] は Lempel–Ziv 77 (LZ77) 分解の変種の一つである．longest previous factor (LPF) 表から LZ77 分解を導出できる．同様に，上記２つの分解も longest previous non-overlapping factor 表 (LPnF) と longest previous non-overlapping reverse factor 表 (LPnrF) [Crochemore et al., JDA ’12] という LPF 表の変種を用いて，それぞれ線形時間で計算可能であることが知られている．本稿で，２つの接尾辞木表現に基づき，その表現の漸近的な領域の範囲に上記２つの表，または，直接に上記２つの分解の計算手法を提案する．

Keywords: 可逆圧縮，Lempel–Ziv 77 の変種，少領域のアルゴリズム，接尾辞木の応用

1　序論

可逆圧縮は，圧縮のうち，復元前の文字列を完全な形で取り出すことが可能な圧縮方法である．文字列の可逆圧縮における一つの専門的分野として，Lempel-Ziv-77 (LZ77) 分解 (Lempel–Ziv 77 分解) [8] がよく知られている．LZ77 は任意文字列 T に対し，T を項に分解する．このとき，LZ77 分解は任意の項がその項の開始位置より前に出現を持つように分解する．

現在，LZ77 といえば，Lempel-Ziv-Storer-Szymanski (LZSS) 分解は LZ77 分解の基準と見なされている．LZSS [7] といった LZ77 の変種も考えられ，LZSS は LZ77 分解の基準とされている．区別のため，本論では従来の LZ77 分解を古典的な LZ77 と呼ぶ．２つの分解は下記のように定義されている．

任意テキスト T に対して，T の部分文字列 $F_1, \ldots, F_z$ の連結 $F_1 \cdots F_z = T$ が T と一致したら，$F_1, \ldots, F_z$ を T の分解と呼び，任意 F_j を項と呼ぶ (任意 $j \in [1..z]$). 一つの分解 $F_1, \ldots, F_z$ を固定すると，各 F_x ($x \in [1..z]$) は (a) T の中に最左の出現の文字 (よって，F_x の長さは１)，もしくは，(b) $F_1 \cdots F_x$ に少なくとも２つ出現がある $F_x \cdots F_z$ の最長接頭辞であるとき，分解 $F_1, \ldots, F_z$ は LZSS 分解と呼ばれる．

その一方で，古典的な LZ77 分解というのは，各 F_x ($x \in [1..z-1]$) は $F_1 \cdots F_x$ の中に一つしかない出現を持つ (その出現は $F_1 \cdots F_x$ の接尾辞) $F_x \cdots F_z$ の最

小接頭辞である．最後の項 F_z は $T[1+|F_1 \cdots F_{z-1}|\,..]$ とする．(F_z は $F_1 \cdots F_z$ の中に複数の出現を持つ場合もある．)

両方の分解の共通性質は，項 F_x は最左の文字の出現ではないとき，F_x は F_x より前に始まる出現がある．その出現を F_x の参照先と呼ぶ．

重複なし変種は参照先に制限を設ける．F_x を特定するとき，F_x の参照先の終了位置を F_x の開始位置より小さくする．この制限の結果は，LZSS 分解の重複なし変種で，各項 F_x は $F_1 \cdots F_{x-1}$ の中に少なくとも一つの出現を持つ．

他の変種，重複なし逆 Lempel-Ziv (LZ) 分解は *gapped* **分解**である [4]. gapped 分解とは，$S^{\mathrm{R}}GS$ で表されるテキスト T の部分文字列，ただし S と G は文字列，$S^{\mathrm{R}} = S[|S|]S[|S|-1]\cdots S[1]$ は $S = S[1]S[2]\cdots S[|S|]$ の逆を示す．T の分解 $T = F_1 \cdots F_z$ は下記の状況を満たすとき，重複なし逆 LZ 分解と呼ぶ．各項 F_x は ($x \in [1\,..\,z]$) 文字の最左の出現，もしくは $F_1 \cdots F_{x-1}$ の中に反対の出現を持つ　$F_1 \cdots F_{x-1}$　の最長接頭辞である．各項に対して，可能な限り最長接頭辞を選択している分解なので，重複なし逆 LZ 分解は貪欲な性質を持つと見なす．

前述の分解は項の表を用いて線形時間で計算することができる，ただし項の表は各テキスト位置 i に対して，i から始まる可能性がある項の最大の長さを格納する．longest previous factor 表 LPF [2] は LZSS に対して特徴的な表になり，longest previous non-overlapping factor 表 LPnF と longest previous non-overlapping reversed factor 表 LPnrF [3] で重複なし LZSS 分解と重複なし逆 LZ 分解，それぞれの分解をを計算できる．

2　本研究

本稿では重複なし LZSS 分解と 重複なし逆 LZ 分解を少領域を持つ接尾辞木の表現で計算するアルゴリズムを提案する．提案したアルゴリズムは LPnF と LPnrF の計算でも応用できる．計算モデルは word RAM，入力を長さ n を持つテキスト T とする，ただし T の文字は大きさ $\sigma = n^{O(1)}$ を持つ整数アルゴリズムから成り立つ．設定の上で，LPF のように [1] 各表を $2n + o(n)$ bits で表現できることが示せる．

分解の計算の場合，以下の定理が得られる．

定理 1 ([5, Theorem 1]) *T の重複なし LZSS 分解を*

- *$O(\epsilon^{-1}n)$ 時間と $(1+\epsilon)n\lg n + O(n)$ ビット領域 (読み取り専用の入力テキスト T の領域を省く), または*
- *$O(n\lg^{\epsilon} n)$ 時間と $O(n\lg\sigma)$ ビット領域で*

計算できる，ただし $\epsilon \in (0, 1]$ は任意に選択可能である．

定理 2 ([6, Theorem 1]) *T の重複なし 逆 LZSS 分解を*

- $O(\epsilon^{-1}n)$ 時間と $(2+\epsilon)n\lg n + O(n)$ ビット領域 *(*読み取り専用の入力テキスト T の領域を省く*)*, または
- $O(\epsilon^{-1}n)$ 時間と $O(\epsilon^{-1}n\lg\sigma)$ ビット領域で

計算できる，ただし $\epsilon \in (0,1]$ は任意に選択可能である．

項の表も同様に計算できる．LPnF を Thm. 1 の計算量で構築できる．Thm. 2 のアルゴリズムを LPnrF の計算に応用できるが， $O(n\lg\sigma)$-ビット領域の結果としては線形時間より遅くなる：

定理 3 ([6, Theorem 2]) LPnrF の $2n + o(n)$ ビット表現を

- $O(\epsilon^{-1}n)$ 時間と $(2+\epsilon)n\lg n + O(n)$ ビット領域 *(*読み取り専用の入力テキスト T の領域を省く*)*, または
- $O(\epsilon^{-1}n\log_\sigma^\epsilon n)$ 時間と $O(\epsilon^{-1}n\lg\sigma)$ ビット領域で

計算できる，ただし $\epsilon \in (0,1]$ は任意に選択可能である．

3 謝辞

この研究は科研費番号 JP22H03551, JP21K17701, と JP21H05847 の支援金を受けたものである．

参考文献

[1] Hideo Bannai, Shunsuke Inenaga, and Dominik Köppl. Computing all distinct squares in linear time for integer alphabets. In *Proc. CPM*, volume 78 of *LIPIcs*, pages 22:1–22:18, 2017.

[2] Maxime Crochemore and Lucian Ilie. Computing longest previous factor in linear time and applications. *Inf. Process. Lett.*, 106(2):75–80, 2008.

[3] Maxime Crochemore, Costas S. Iliopoulos, Marcin Kubica, Wojciech Rytter, and Tomasz Walen. Efficient algorithms for three variants of the LPF table. *J. Discrete Algorithms*, 11:51–61, 2012.

[4] Roman Kolpakov and Gregory Kucherov. Searching for gapped palindromes. *Theor. Comput. Sci.*, 410(51):5365–5373, 2009.

[5] Dominik Köppl. Non-overlapping LZ77 factorization and LZ78 substring compression queries with suffix trees. *Algorithms*, 14(2)(44):1–21, 2021.

[6] Dominik Köppl. Reversed Lempel–Ziv factorization with suffix trees. *Algorithms*, 14(6)(161):1–26, 2021.

[7] James A. Storer and Thomas G. Szymanski. Data compression via textural substitution. *J. ACM*, 29(4):928–951, 1982.

[8] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Trans. Information Theory*, 23(3):337–343, 1977.