CDAWG による LZ78 部分文字列圧縮

柴田 紘希 *　　クップル ドミニク †

1　はじめに

部分文字列圧縮とは、事前に与えられたテキスト文字列 T ($|T| = n$) の部分文字列 $T[i..j]$ ($1 \le i \le j \le n$) に対する圧縮表現を高速に計算する問題である。この問題は文字列圧縮の代表的な手法である LZ77 分解に対して提案され [1]、近年では、LZ78 分解などのその他の圧縮手法に対する効率的な解法が提案されている [2]。

部分文字列圧縮の高速な処理では、まず与えられたテキスト T に対する何らかの索引構造 $\mathcal{D}(T)$ を構築し、与えられた部分文字列圧縮クエリに対して $\mathcal{D}(T)$ を用いて計算を行う。圧縮する部分文字列の長さは元のテキスト文字列 T の長さ以下であるため、部分文字列圧縮のためのメモリ使用量は多くの場合その索引構造 $\mathcal{D}(T)$ の大きさに依存する。そのため、部分文字列圧縮の省領域な計算には索引構造の省領域化が必要不可欠である。

本研究では、LZ78 分解に対する部分文字列圧縮を圧縮領域で行う手法を提案する。提案手法では、索引構造 $\mathcal{D}(T)$ として圧縮表現の一つである Compacted Acyclic Word Graph (CDAWG) [3] を用いることで、$O(z \lg n)$ 時間、$O(e+z)$ 領域での部分文字列圧縮を達成する。ここで、e は CDAWG の辺数であり、z は部分文字列圧縮によって得られる出力のフレーズ数である。従来手法 [2] の時間計算量は $O(z)$、空間計算量は $O(n+z)$ であったため、提案手法は対数倍の速度低下で圧縮索引化を達成したといえる。

*九州大学 マス・フォア・イノベーション連係学府
†山梨大学 大学院 総合研究部 工学域 電気電子情報工学系

2　LZ78 分解

LZ78 分解とは、文字列をいくつかの factor に分解する圧縮手法の一つである。文字列 T の LZ78 分解とは、各 F_i が整数 $j < i$ と文字 c を用いて $F_i = F_j c$ または $F_i = c$ と表せるような T の分割 $F_1, F_2, \ldots, F_k$ の中で、各 factor の長さを並べた列 $|F_1|, \ldots, |F_k|$ が辞書順最大となるような分割のことである。T の LZ78 分解は、まず $\ell_0 = 0$ とし、各 $i = 1, 2, \ldots$ に対して $T[\ell_{i-1}+1..\ell_i-1] = F_j$ となる $j < i$ が存在するような最大の ℓ_i を求めて $F_i = F_j T[\ell_i]$ を新たな factor とする手続きを繰り返すことで得ることができる。

3　CDAWG による LZ78 分解

CDAWG とは、文字列の任意の接尾辞を受理する Trie 木の次数 1 の頂点を縮約して同型な部分木をまとめることで得られる辺ラベル付き DAG である。文字列 T を表現する CDAWG の各頂点・各辺はそれぞれ T の極大繰り返し・右拡張と対応していることが知られている。CDAWG の辺数 e は最悪で $\Theta(n)$ となるが、文字列が繰り返し構造を持つ場合 e の値は小さくなるため、CDAWG は圧縮表現の一つとして研究されている。

CDAWG は他の圧縮表現に比べて多くのクエリを圧縮領域上で行うことができる。本研究では、テキスト T やその接尾辞配列 SA の任意の位置へのアクセスを $O(e)$ 領域かつ $O(\lg n)$ 時間で行うことができる索引構造 [4] をもとに、この索引上で T の部分文字列 $T[l, r]$ に対応する SA 上の区間を求める手法を新たに考案し、圧縮領域上での LZ78 部分文字列

圧縮のために用いる。

本研究では、文字列 T の CDAWG と一次元区間に対する stabbing-max query [5] のための索引を用いて LZ78 分解を行う。stabbing-max query とは、各区間が重み w と値 v を持つような区間の集合 $\{([l_1, r_1], w_1, v_1), \ldots, ([l_m, r_m], w_m, v_m)\}$ に対して、整数 p を入力に、$p \in [l_r, r_i]$ を満たす中で重み w_i が最大となるような区間の値 v_i を出力するクエリのことである。 stabbing-max query を効率的に処理するための索引として、空間計算量が $O(m)$ 、区間追加の計算量がならし $O(\lg m)$、クエリの時間計算量が $O(\lg m / \lg \lg m)$ であるような索引構造が知られている [6]。

CDAWG による LZ78 分解では、通常の場合の手続きと同様に、factor の分割位置 ℓ_i と factor F_i を i の昇順に求める。位置 $\ell_i + 1$ を始点とする i 番目の factor F_i の計算は以下のような手順で行われる。

1. CDAWG を用いて $\mathsf{SA}[\ell_{i-1} + 1]$ を計算する。
2. $j = \mathrm{query}_{\mathcal{I}}(\mathsf{SA}[\ell_{i-1} + 1])$ を求め、$\ell_i = \ell_j + |F_j| + 1, F_i = F_j T[\ell_i]$ とする。
3. CDAWG を用いて $F_i = T[\ell_{i-1} + 1..\ell_i]$ に対応する SA 上の区間 $[l, r]$ を求め、新たな要素 $([l, r], |F_i|, i)$ を $\mathcal{I}$ に追加する。

ここで、$\mathcal{I}$ は stabbing-max query のための索引であり、$\mathrm{query}_{\mathcal{I}}(p)$ は索引上が保持する区間の集合に対して整数 p を入力とするクエリを行った結果を表す。索引 $\mathcal{I}$ は計算済の各 factor $F_1, \ldots, F_{i-1}$ に対応する SA 上の区間を保持しており、各区間は対応する factor の長さによって重みづけされている。そのため、$\mathrm{query}_{\mathcal{I}}(\mathsf{SA}[\ell_{i-1} + 1])$ によって $T[\ell_{i-1} + 1..n]$ の接頭辞として現れるような長さ最大の factor が得られる。

i 番目の factor F_i を求める操作全体の計算量は、CDAWG を用いて SA へのアクセスを行うための計算量と、stabbing-max query のための索引 $\mathcal{I}$ を用いてクエリを処理し、新たな区間を追加するために必要な計算量の和になる。SA へのアクセスや部分文字列に対応する SA 上の区間を求める計算量は $O(\lg n)$ であり、$\mathrm{query}_{\mathcal{I}}(p)$ の計算量は $O(\lg n / \lg \lg n)$ であり、$\mathcal{I}$ への区間追加の計算量はならし $O(\lg n)$ であるため、factor を 1 つ計算するための計算量は合計でならし $O(\lg n)$ となる。したがって、LZ78 分解の計算全体の計算量は $O(z \lg n)$ となる。

上述のアルゴリズムは部分文字列圧縮にも同様に適用可能である。T の部分文字列 $T[l, r]$ の LZ78 分解を計算する場合は、$\ell_0 = l - 1$ とし、$\ell_i \geq r$ となるまで計算を行えばよい。この場合の計算量も同様に $O(z \lg n)$ となる。

参考文献

[1] Storer, J. A & Szymanski, T. G. (1978) *The Macro Model for Data Compression (Extended Abstract)* eds. Lipton, R. J, Burkhard, W. A, Savitch, W. J, Friedman, E. P, & Aho, A. V. (ACM), pp. 30–39.

[2] Köppl, D. (2021) Non-overlapping LZ77 factorization and LZ78 substring compression queries with suffix trees. *Algorithms* **14**, 44.

[3] Blumer, A, Blumer, J, Haussler, D, Ehrenfeucht, A, Chen, M. T, & Seiferas, J. I. (1985) The smallest automaton recognizing the subwords of a text. *Theor. Comput. Sci.* **40**, 31–55.

[4] Belazzougui, D & Cunial, F. (2017) *Representing the Suffix Tree with the CDAWG*, LIPIcs eds. Kärkkäinen, J, Radoszewski, J, & Rytter, W. (Schloss Dagstuhl - Leibniz-Zentrum für Informatik), Vol. 78, pp. 7:1–7:13.

[5] Gupta, P & McKeown, N. (2001) Algorithms for packet classification. *IEEE Netw.* **15**, 24–32.

[6] Nekrich, Y. (2011) *A Dynamic Stabbing-Max Data Structure with Sub-Logarithmic Query Time*, Lecture Notes in Computer Science eds. Asano, T, Nakano, S, Okamoto, Y, & Watanabe, O. (Springer), Vol. 7074, pp. 170–179.