2018 年度冬の LA シンポジウム [19]

PLCP 配列に基づくテキスト圧縮

Patrick Dinklage* Jonas Ellert* Johannes Fischer* Dominik Köppl†
Manuel Penschuck‡ ,

平成 31 年 2 月 6 日

概 要

大規模な入力を効率的に圧縮するという問題を対象とした解法はまだ存在しない。LCPcomp は LZ77 と同等の圧縮率を達成する双方向な参照圧縮方法である。この方法を実装する現在のアルゴリズムは理論的には線形時間で動作するにも関わらず、実際には遅く、膨大に主記憶を消費する。この論文は、この課題に取り組み、PLCPcomp と呼ばれる新しいアルゴリズムを提案する。

序論

LCPcomp [1] は LZ77 [5] のようなテキスト圧縮方法である。LZ77 は線形的に入力テキストを読み込み、i 番目の文字に到着した際、i と 任意 i の前の位置 j の両方を、開始位置とする出現がある最長部分文字列を計算し、開始位置 i を持つ出現を、 j を持つ出現についての参照で置換する。すなわち、 LZ77 の参照先は常に左を指す。一方で、 LCPcomp の参照先はそのような制限がない。そのため、より良い圧縮率を実現できる可能性がある。LCPcomp は最長な繰り返し部分文字列を選び（重複も許す)、1 つの出現を参照に置換し、出現を 2 つ持つ部分文字列がなくなるまで繰り返す。

LCPcomp と LZ77 は、どちらも線形時間で計算できるにも関わらず、実用的に LCPcomp のアルゴリズムは遅く、膨大に主記憶を消費する。これは、様々なテキストデータ構造を使うためである。最適化のため、必要なテキストデータ構造を削減したい。しかしながら、その場合は同じアルゴリズムを使用できない。そこで、PLCPcomp という新しいアルゴリズムを提示する。このアルゴリズムはテキスト構造を 2 つしか必要としない。

準備

Σ を整数アルファベットとし、Σ^* の要素を**文字列**と呼ぶ。文字列 T の長さを $|T|$ と表記する。文字列 $T = XYZ$ について X, Y, Z をそれぞれ文字列 T の**接頭辞、部分文字列、接尾辞**と呼ぶ。$T[i]$ は T 中の i 文字目の文字、$T[i..j]$ は T の i 文字目から j 文字目までの部分文字列を表す。$j = |T|$ のとき、$T[i..j]$ は開始位置 i を持つ接尾辞であり、$T[i..]$ とも書く。

序論で述べたテキスト構造は PLCP 配列と Φ 配列である。任意の位置 dst に対して、Φ[dst] は T[dst..] という接尾辞の辞書順による最寄りの小さい接尾辞の開始位置である [3]。PLCP[dst] は T[dst..] と $T[\Phi$[dst]..] の、両方の最長一致の接頭辞の長さである。Φ 配列はテキストから直接に作ることができ [2]、Φ 配列とテキストから PLCP 配列を求めることができる [3]。$\mathcal{O}(|\Sigma| \lg n)$ ビット追加領域を用いることで、このアルゴリズムは PLCP と Φ を線形時間で計算できる。

アルゴリズム

PLCPcomp は、Φ 配列と PLCP 配列を使って、LCPcomp を計算できる。手順の本質は以下の通

*TU Dortmund
†九州大学 / 日本学術振興会
‡Goethe University Frankfurt

り：PLCP 配列の最大値 PLCP[dst] を見つけて、T[dst..dst + PLCP[dst] − 1] を T[Φ[dst]..Φ[dst] + PLCP[dst] − 1] についての参照で置換して、PLCP 配列の更新を繰り返す。複数の最大値 PLCP[dst] があれば、最左の位置を選ぶ。

PLCP 配列の更新は 2 つの命令から成り立つ：

- 各 $j \in$ [dst − PLCP[dst]..dst) について、PLCP[j] ← min(PLCP[j], dst − j) とする。すなわち、前位置の PLCP 値を適当に減らす。
- 各 $j \in$ [dst..dst + PLCP[dst]) について、PLCP[j] ← 0 とする。すなわち、置換した位置を PLCP 配列から消す。

上記の方法で $\mathcal{O}(n^2)$ 時間で動作する LCPcomp のアルゴリズムを作ることができる。さらに、以下で示される方法は線形時間で動作する。全部の PLCP 値を管理する必要がないことが最初の手がかりである。つまり、以下に定義する山という位置にのみ注目すればよい。位置 dst が山と呼ばれるのは、dst が以下の 3 つのうち 1 つ以上の条件を満たすとき。

- dst = 1,
- PLCP[dst − 1] < PLCP[dst], または
- dst − 1 は置換した部分文字列の終わりである。

次の手がかりは以下の通り：dst の PLCP 値が十分に大きければ、T[dst..dst + PLCP[dst] − 1] は LCPcomp の方法で必ず置換される。詳細な説明のため、以下の 2 つの定義が必要である:

- 山 dst は、 $j <$ dst $< j +$ PLCP[j] を満たす各位置 j について、PLCP[j] < PLCP[dst] を満たすと、 dst は**面白い山**と呼ばれる。
- dst が面白い山であり、かつ各 $j \in$ (dst..dst + PLCP[dst]) が面白い山ではないとき、dst は**最高の山**と呼ばれる。

このとき、LCPcomp が最左の最高の山を置換することを証明できる。（省略）

PLCPcomp はこの手がかりを利用し、 線形的にテキストを走査し、最左の最高の山 dst を探す。この山 dst を見つけたとき、 T[dst..dst + PLCP[dst] − 1] を置換する。その後、T[1..dst] で新しい最高の山を探す。PLCP[1..dst − 1] で最大の値を持つ位置は最高の山なので、その最大の値を探す。そのために、dst を見つけるまで、全部の面白い山をリスト L で保存することで、L[$|L|$ − PLCP[dst]..$|L|$] で新しい最高の山を見つけることができる。L のサイズは高々 $\mathcal{O}(\sqrt{n \lg n})$ [4] なので、T[1..dst − 1] を LCPcomp の方法通りに再帰的に線形時間で置換できる。最後に、初めの線形的な走査を T[dst + PLCP[dst]..] から続行する。

今後の課題　記憶を節約する Φ 配列の表現方法を提案し、外部メモリのモデルでどうやって PLCPcomp を応用して、双方向な参照を使う方法で圧縮したデータを効率的に展開できるかどうか。

未解決問題　置換した部分文字列の数について、LZ77 は左の参照を使う参照圧縮方法の中で、この数が最小である。双方向な参照を許す方法について、この数に関する結果はまだ知られていない。

参考文献

[1] P. Dinklage, J. Fischer, D. Köppl, M. Löbel, and K. Sadakane. Compression with the tudocomp framework. In *Proc. SEA*, volume 75 of *LIPIcs*, pages 13:1–13:22, 2017.

[2] K. Goto and H. Bannai. Space efficient linear time Lempel-Ziv factorization for small alphabets. In *Proc. DCC*, pages 163–172, 2014.

[3] J. Kärkkäinen, G. Manzini, and S. J. Puglisi. Permuted longest-common-prefix array. In *Proc. CPM*, volume 5577 of *LNCS*, pages 181–192, 2009.

[4] J. Kärkkäinen, D. Kempa, and M. Piatkowski. Tighter bounds for the sum of irreducible LCP values. *Theor. Comput. Sci.*, 656:265–278, 2016.

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