LD-002

全文索引構築のための省スペースなアルゴリズム
Space-Efficient Algorithms for Constructing Full-Text Indices

定兼 邦彦 † Kunihiko Sadakane
宋 永健 ‡ Wing-Kin Sung
韓 永楷 § Wing-Kai Hon

1. はじめに

文書の全文索引は Web 検索，自然言語処理，生物情報処理などで広く用いられている重要な技術である．索引の例としては転置ファイル，接尾辞木，接尾辞配列などがある．

これらの索引を作成するアルゴリズムはいくつも存在するが，$o(n \log n)$ 時間，かつ $o(n \log n)$-bit 作業領域で動作するものは知られていない．よって大規模データに対する索引の構築には時間またはスペースの問題があった．

本論文では上述の索引を構築する高速・省スペースなアルゴリズムを提案する．長さ n の文字列に対し，圧縮接尾辞配列を $O(n)$ 時間，$O(n)$-bit の作業領域で構築できる．アルファベットを $\mathcal{A}$ とすると，$O(n \log\log |\mathcal{A}|)$ 時間，$O(n \log |\mathcal{A}|)$-bit 作業領域である．$\log |\mathcal{A}| = o(\log n)$ のとき，これは $o(n \log n)$ 時間，$o(n \log n)$-bit 空間で動く初めてのアルゴリズムである．このアルゴリズムを用いると，接尾辞木を $O(n \log^{\epsilon} n)$ 時間，$O(n)$-bit 作業領域で構築できる ($0 < \epsilon < 1$ は任意の定数)．その他の索引については表 1 参照．

計算のモデルは語長 $O(\log n)$ の word RAM である．つまり，$O(\log n)$ ビットの数に対する算術・論理演算や $O(\log n)$ ビットの値のメモリへの読み書きが定数時間で行えるとする．長さ n の文字列へのポインタは $\log_2 n$ ビット必要であるため，このモデルは一般的である．

表 1: 全文索引の構築のための既存の最適時間および最適スペースアルゴリズムと本研究の比較．$0 < \epsilon < 1$．ST，SA，CST，CSA，FM はそれぞれ接尾辞木 [8]，接尾辞配列 [7]，圧縮接尾辞木 [5]，圧縮接尾辞配列 [4]，FM-index [3] を表す．

索引	アルゴリズム	時間	領域 (bits)
SA CSA FM	最適時間 [2]	$O(n)$	$O(n \log n)$
	最適領域 [6]	$O(n \log n)$	$O(n)$
	本論文	$O(n)$	$O(n)$
ST CST	最適時間 [2]	$O(n)$	$O(n \log n)$
	最適領域 [6]	$O(n \log n)$	$O(n)$
	本論文	$O(n \log^{\epsilon} n)$	$O(n)$

2. 準備

文字列を $T[1..n] = T[1]T[2]\cdots T[n]$ と表す．T の接尾辞は $T[j..n]$ $(j = 1, 2, \ldots, n)$ である．T の接尾辞配列 $SA[1..n]$ とは T の接尾辞を辞書順にソートし，その添字 j を格納した整数の配列である．そのサイズは $n \log_2 n$ ビットとなる．接尾辞配列を用いると T 中のパタンの検索が行える．

圧縮接尾辞配列は接尾辞配列の替わりに $\Psi[i] = SA^{-1}[SA[i] + 1]$ で定義される Ψ 関数を格納する．そのサイズは $O(n \log |\mathcal{A}|)$ ビットであり，各要素は定数時間で読み出せる [4]．

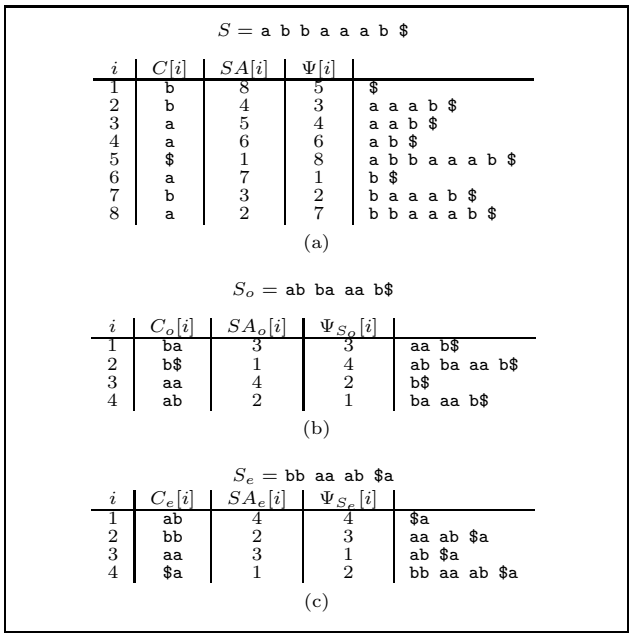

図 1: S, S_o, S_e に対する接尾辞配列, Ψ 関数, C

BW 変換 [1] はブロックソート圧縮法の基本であり，文字列 $T[1..n]$ を $C[i] = T[SA[i]-1]$ (if $SA[i] \neq 1$) または $C[i] = T[n]$ (if $SA[i] = 1$) で定義される文字列 $C[1..n]$ へ変換することである (図 1(a) 参照)．

補題 1 *C と Ψ は双対の関係にあり，互いに $O(n)$ 時間，$O(n \log |\mathcal{A}|)$-bit スペースで変換できる．*

3. アルゴリズム

本論文のアルゴリズムは，まず圧縮接尾辞配列の Ψ 関数を作成し，その後接尾辞配列，接尾辞木などを構築する．後者については既に省スペースのアルゴリズムが提案されているため [5]，本論文では前者についてのみ扱う．

$h = \lceil \log_2 \log_{|\mathcal{A}|} n \rceil$ とする．文字列 T の長さ n は 2^{h+1} の倍数とする．そうでない場合はダミー文字を付け加えて Ψ を作成した後，それを T に対する Ψ に変換する．$0 \leq k \leq h$ について T^k を T 中の連続する 2^k 文字を 1 つの文字とした文字列と定義する．つまり，$T^k[i] = T[(i-1) \cdot 2^k + 1..i \cdot 2^k]$ $(1 \leq i \leq n/2^k)$ となる．定義より $T^0 = T$ である．T^k のアルファベットは $\mathcal{A}^{2^k}$ となる．なお，T^k のアルファベットサイズは $|\mathcal{A}|^{2^k} \leq 2n$ であるため，$1 + \log_2 n$ ビットで表せる．つまりどの 1 文字も定数時間で読み書きできる．

†九州大学大学院システム情報科学研究院 sada@csce.kyushu-u.ac.jp
‡School of Computing, National University of Singapore ksung@comp.nus.edu.sg
‡Department of Computer Science, The University of Hong Kong wkhon@csis.hku.hk

また，任意の文字列 $S[1..m]$ に対し，S_o と S_e をそれぞれ $S[1..m]$ と $S[2..m]S[1]$ 中の連続する 2 文字を併合した文字列と定義する．直感的には，S_o (S_e) はそれぞれ S の接尾辞のうち添字が奇数 (偶数) のものに対応する．

3.1 概要

本論文のアルゴリズムのアイデアは以下の通りである．Farach の接尾辞木構成アルゴリズム [2] と同様に，長さが半分の文字列に対する Ψ 関数を再帰的に構成し，それをもとに T の Ψ 関数を構成する．その際に，Ψ と双対の関係にある BW 列 C を一旦作成し，その後 Ψ 関数に変換する．こうすることで，既存手法 [6] には存在した Ψ 関数更新のオーバーヘッドをなくした．

本論文のアルゴリズムではまず文字列 T^h に対する Ψ_{T^h} を作成する．その後，$\Psi_{T^{i+1}}$ から Ψ_{T^i} を作成することを繰り返す．なお，$\Psi_{T^i_o} = \Psi_{T^{i+1}}$ が成り立つ．つまり，(1) $\Psi_{T^{i+1}} (= \Psi_{T^i_o})$ から $\Psi_{T^i_e}$ を作成し，(2) $\Psi_{T^i_o}$ と $\Psi_{T^i_e}$ から Ψ_{T^i} を作成すればよい．

3.2 $\Psi_{T^i_e}$ の作成

$\Psi_{T^i_e}$ を作成するにはまず T^i_e を BW 変換した文字列 C_e を作成し，次にそれを $\Psi_{T^i_e}$ に変換する．

$S = T^i$ とする．S の長さを m, アルファベットを Δ とする．文字列 S_o の接尾辞配列を SA_o で表す (図 1(b))．

補題 2 Ψ_{S_o} と S が与えられたとき，C_e は $O(m + |\Delta|)$ 時間，$O(m \log |\Delta|)$-*bit* 空間で計算できる．

略証: 配列 $x[1..m/2]$ を $x[i] = S[2SA_o[i] - 2]$ と定義し，文字列 X_i を $x[i]S_o[SA_o[i]..m/2]S[1]$ と定義する．すると X_i は S_e の接尾辞となり (図 2 参照)，$x[i]$ を $(x[i], i)$ に従ってソートしたときの順序と X_i (つまり S_e の接尾辞) の辞書順は一致する．このソートは $x[i]$ に関する安定な基数ソートで求まる．$y[i]$ を X_i の直前の 2 文字とすると，ソート後の $y[i]$ が $C_e[i]$ となる． □

C_e から $\Psi_{T^i_e}$ は補題 1 により計算できる．

3.3 Ψ_{T^i} の作成

$S = T^i$ とする．Ψ_S を作成するには，S の全ての接尾辞の辞書順を計算し，それを元に S の BW 列 C を求め，最後に C から Ψ_S を作成すればいい (図 3)．

補題 3 Ψ_{S_o} と Ψ_{S_e} が与えられたとき，Ψ_S は $O(m \log\log |\Delta|)$ 時間，$O(m \log |\Delta| + |\Delta|)$-*bit* 空間で計算できる．

略証: S の接尾辞の集合は S_o の接尾辞の集合と S_e の接尾辞の集合に分割される．S のある接尾辞 s の辞書順は，S_o の接尾辞の間での s の辞書順と，S_e の接尾辞の間での s の辞書順の和に等しい．これらの辞書順は Ψ_{S_o} と Ψ_{S_e} を用いた後方検索 [9] によって求まる． □

定理 1 T の Ψ 関数は $O(n \log\log |\mathcal{A}|)$ 時間，$O(n \log |\mathcal{A}|)$-*bit* 空間で計算できる．

謝辞

この研究の一部は文部科学省科学研究費の援助を受けた．

		X_i		
i	$y[i]$	$x[i]$	$S_o[SA_o[i]..m/2]$	$S[1]$
1	bb	a	aa b$	a
2	ab	$	ab ba aa b$	a
3	aa	a	b$	a
4	$a	b	ba aa b$	a

(a)

i	$y \to C_e$	$x \to$ sorted x
1	ab	$
2	bb	a
3	aa	a
4	$a	b

(b)

図 2: $S =$ abbaaab$. (a) $x[i]$, $y[i]$, X_i の関係．X_i は S_e の接尾辞に対応 (図 1(c) 参照)．(b) x に関する安定ソート後の x と y.

1. For $i = 1$ to $m/2$
   (a) $S_o[m/2 - i + 1..m/2]$ の，S の奇数接尾辞中の辞書順と，偶数接尾辞中での辞書順を求める．その和を k とする．
   (b) $C[k] = S[m - 2i + 1]$ とする．
2. S_e の接尾辞の辞書順を同様に計算し，$C[k]$ を埋める．
3. C から Ψ_S を計算する．

図 3: Ψ_S の計算

参考文献

[1] M. Burrows and D. J. Wheeler. A Block-sorting Lossless Data Compression Algorithms. Technical Report 124, Digital SRC Research Report, 1994.

[2] M. Farach. Optimal Suffix Tree Construction with Large Alphabets. In *38th IEEE Symp. on Foundations of Computer Science*, pages 137–143, 1997.

[3] P. Ferragina and G. Manzini. Opportunistic Data Structures with Applications. In *41st IEEE Symp. on Foundations of Computer Science*, pages 390–398, 2000.

[4] R. Grossi and J. S. Vitter. Compressed Suffix Arrays and Suffix Trees with Applications to Text Indexing and String Matching. In *32nd ACM Symposium on Theory of Computing*, pages 397–406, 2000.

[5] W. K. Hon and K. Sadakane. Space-economical Algorithms for Finding Maximal Unique Matches. In *Proc. Combinatorial Pattern Matching*, pages 144–152. LNCS 2373, 2002.

[6] T. W. Lam, K. Sadakane, W. K Sung, and S. M Yiu. A Space and Time Efficient Algorithm for Constructing Compressed Suffix Arrays. In *Proc. COCOON*, pages 401–410. LNCS 2387, 2002.

[7] U. Manber and G. Myers. Suffix arrays: A New Method for On-Line String Searches. *SIAM Journal on Computing*, 22(5):935–948, October 1993.

[8] E. M. McCreight. A Space-economical Suffix Tree Construction Algorithm. *Journal of the ACM*, 23(12):262–272, 1976.

[9] K. Sadakane. Succinct Representations of *lcp* Information and Improvements in the Compressed Suffix Arrays. In *Proc. ACM-SIAM SODA 2002*, pages 225–232, 2002.