オンライン文法圧縮による頻出パターン発見

Online Grammar Compression for Frequent Pattern Discovery

福永祥平　高畠嘉将　井智弘　坂本比呂志 *
Shouhei Fukunaga, Yoshimasa Takabatake, Tomohiro I, Hiroshi Sakamoto

九州工業大学
Kyushu Institute of Tecnology

Abstract: We propose an online algorithm for the problem of approximate frequent pattern in a compressed space. The main contribution of this work is an improvement of the previously best approximation ratio $\Omega(\frac{1}{\lg^2 m})$ to $\Omega(\frac{1}{\lg^* N \lg m})$ where m is the length of an optimal pattern in a string of length N and $\lg^*$ is the iteration of the logarithm base 2. For a sufficiently large N, $\lg^* N$ is practically constant. The experimental results show a significant improvement in memory consumption compared to the offline algorithm.

1　はじめに

文法圧縮は，データの共通部分を文脈自由文法 (CFG) の生成規則として表現することにより，データの冗長性を取り除いて圧縮する枠組みである．これまでに様々な文法圧縮アルゴリズムが提案されており，特に長い頻出パターンが含まれるデータには良い結果を示している．現在，このような頻出で長いパターンを含むデータが増加しており，例えばDNAやバージョン管理されたソースコードなどが挙げられ，それらの多くがネットワーク上で入手可能である．このように，繰り返しの多いデータは急速に増えており，これらの有効利用のために文法圧縮は有用である．

頻出パターン発見はパターンマイニングの典型的な問題で，頻出パターンとは2回以上出現するパターンである．定兼らが提案している圧縮接尾辞 [9] や FM-index [2] を用いた手法によって線形時間でこの問題を解くことができるが，大規模データに対しては非常に大きなメモリが必要となる．またこれらの手法では，データを読み込みながらオンラインデータ処理することは難しい．

この問題を解決する一つの方法として，正確な頻出パターンを発見するのではなく，近似的な頻出パターンを発見する方法がある．文法圧縮では，圧縮率を高めるために，できるだけ多く出現する文字列あるいは2回以上出現するできるだけ長い部分文字列という基準で文字列中の一致箇所を探し出して，それらに対し共通の部分木が生成される．Cormode らによる [1] Edit-sensitive parsing (ESP) は，文字列の拡張された編集距離を近似的に高速に求めることができ，この手法を応用した文法圧縮とそのオンラインアルゴリズムが提案されている [4, 10].

このように，文法圧縮は頻出する部分文字列を発見しそれを構文木に置き換えることが主要な目的であるため，テキスト中に存在する頻出な繰り返し構造を発見する問題と密接に関係している．中原らは ESP-comp [6] を提案し，ESP に基づく文法圧縮に注目し，頻出パターン発見を近似的に解決する手法を示した．彼らの手法では，頻出パターンを P としたとき，近似率は $\Omega(\frac{1}{\lg^2 |P|})$ が保証されている．ここで，P の近似率が α $(0 < \alpha \leq 1)$ であるとは，アルゴリズムが出力する P の部分文字列を Q とするとき，P の出現位置によらず，$|Q|/|P| \geq \alpha$ を満たすことをいう．α-近似アルゴリズムは任意の P に対してこの近似率を満たす．

本研究では，文字列長を N，パターン長を m とするとき，頻出パターンの近似率の下限値を $\Omega(\frac{1}{\lg^* N \lg m})$ に改善し，さらに文法圧縮を用いた近似パターン発見をオンラインにも適用し，ESP-comp よりも省スペースであることを示す．また，提案手法を実装し，実験を行い実際に得られた近似率を示し，メモリ使用量と実行時間を他の手法と提案手法を比較した．

2　準備

集合 C の要素数を $|C|$ とする．Σ を有限アルファベットの集合とし，$\sigma = |\Sigma|$ とする．Σ からなるすべての文字列を Σ^* とし，$\Sigma^q = \{w \in \Sigma^* : |w| = q\}$ とし，さらに Σ^q のある一つの要素を q-gram と呼ぶ．文字列 S の長さを $|S|$ と表記する．文字列は $S = \alpha\beta\gamma$ に分割でき，

*連絡先：九州工業大学
〒 820-8502 福岡県飯塚市川津 680-4
E-mail: hiroshi@ai.kyutech.ac.jp

α,β,$\gamma \in \Sigma^*$ をそれぞれ接頭辞，部分文字列，接尾辞と呼ぶ．文字列 S の i 番目の要素を $S[i](i \in [1, |S|])$ と表記し，文字列 S の部分文字列として，S の i 番目から j 番目の文字列を $S[i, j](1 \leq i \leq j \leq |S|)$ と表記する．文字列 S 中に含まれる文字列 P の頻度を $freqs(P)$ と表記する．また，lg で $\lg_2$ を表し，$\lg^{(1)} u = \lg u$，$\lg^{(i+1)} u = \lg(\lg^{(i)} u)$ とするとき，$\lg^* n = \min\{i \mid \lg^{(i)} n \leq 1\}$ と定義する．ここで，十分に大きい n (例えば $n \leq 2^{65536}$) に対して $\lg^* n \leq 5$ のため，$\lg^* n$ は実用的には定数と考えてよい．

2.1 文法圧縮

文法圧縮とは CFG を構築することによって圧縮する方法である．CFG の標準形のひとつであるチョムスキー標準形は生成規則が $X_i \to X_j X_k$ の形式に制限されている物であり，そのうちすべての生成規則が $i > j, k$ を満たすものを SLP (straight-line program) という．SLP は $G = \{V, \Sigma, P, X_s\}$ で表される．ここで，V は変数の集合，Σ はアルファベットの集合 ($\Sigma \cap V = \emptyset$)，$P$ は生成規則 $X_i \to X_j X_k$ の集合 ($X_i \in V, X_j, X_k \in \Sigma \cup V$)，$X_s \in V$ は SLP の開始記号である．$n = |V|$ とし，X_i から導出される文字列，すなわち，X_i を根とする部分木の葉の列を $val(X_i)$ と表記する．

文法圧縮を計算機上で実現するには，$X_i \to X_j X_k$ に対して X_i から $X_j X_k$ にアクセスするための辞書 D とその逆方向の参照をするための逆引き辞書 D^{-1} の 2 つのデータ構造が必要である．

2.2 近似頻出パターン

パターン P が S に 2 回以上出現するとき，つまり $Freqs_S(P) \geq 2$ のとき，P は頻出という．ただし，以降では S を省略して $Freqs(P)$ などと表記する．本研究では，定義 2.1 に定義される近似頻出パターン発見問題に注目する．

定義 2.1. $S \in \Sigma^*$ を導出する構文木を T とする．T の変数 X は，P の任意の出現 $S[i, j] = P$ が $val(X)$ を含むとき X を P の *core* と呼ぶ．また，$\frac{|val(X)|}{|P|} \geq \delta$ ならば X は P の δ-近似であるという．近似頻出パターン発見問題 *(AFP)* とは，任意の頻出パターン P に対して近似率 δ となることを保証する X を発見する問題である．

中原らが提案したオフラインの手法 [6] の近似率の下限値は $\delta = \Omega(\frac{1}{\lg^2 |P|})$ である．本研究では，省スペースかつオンラインで行い，さらに近似率の下限値の改善を目的とする．提案手法のアルゴリズムでは，ESP(Edit Sensitive Parsing) と POSLP(post-order SLP) で表現される文法圧縮である．SLP や POSLP およびそれに関連するデータ構造を図 1 に示した．次に，SLP をどのように構築するかについて説明する．

2.3 Edit Sensitive Parsing(ESP)

ESP では文字列 S から ESP 木と呼ばれる parse tree を構築する [1]．また，この木は全 2 分木としても表現できる [10]．S を以下の 3 つの Type の部分文字列の列とみなして長さ 2 または 3 文字毎に分割を行う．

Type1 長さが 2 以上の同一文字による連続文字列．

Type2 同一文字による連続文字列を含まない，長さ $\lg^* |S|$ を超える文字列．

Type3 Type1，Type2 共に当てはまらない文字列．すなわち同一文字による連続文字列を含まない長さ $\lg^* |S|$ 以下の文字列である．

Type1，Type3 に関しては左優先で 2 文字ずつをペアにして置き換えを行っていく．長さが偶数の場合は，XY から $t_i \to XY$ が，長さが奇数の場合は最後の 3 文字を XYZ から $t_i \to Xt'$，$t' \to YZ$ のように 2-2 木を構築する．Type2 の文字列に関してはアルファベット変換 [1] と呼ばれる手法で，文字列 S の中に landmark を定義して S の分割を行っていく．$S[i]$ が landmark と判定された場合，$S[i, i+1]$ がペアとなる．Type2 文字列と landmark の定義により，landmark 同士が隣接することはなく，landmark が決定されたときに S の置き換えの仕方は一意に定まる．アルファベット変換による ESP 木がどのように構築されていくかを図 2 に示した．ESP や landmark についての詳細は，論文 [1] などを参照のこと．

定理 2.1. $S[i]$ が *landmark* であるか否かは，高々 $S[i - \lg^* |S|, i + O(1)]$ のみによって決定される．

3 提案手法

本論文で行った実験では辞書の作成に Fully-online LCA(FOLCA)[5] を利用している．FOLCA はオンラインで SLP の情報理論的下限に漸近的に一致するサイズの辞書 D を構築する．FOLCA を用いた頻出パターンの近似発見アルゴリズムの概略を述べる．また，この提案アルゴリズムが保障する近似率の下限を与える．なお，アルゴリズムの詳細と近似率の証明は，プレプリントサーバの論文 [3] を参照のこと．

S の ESP 木を構築したとき，i 回目の ESP で生成される文字列を S_i とする．ここで $i = 0, 1, \ldots, \lceil \lg^* |S| \rceil$ であり，$S_0 = S$ である．アルゴリズムは S_i を計算する

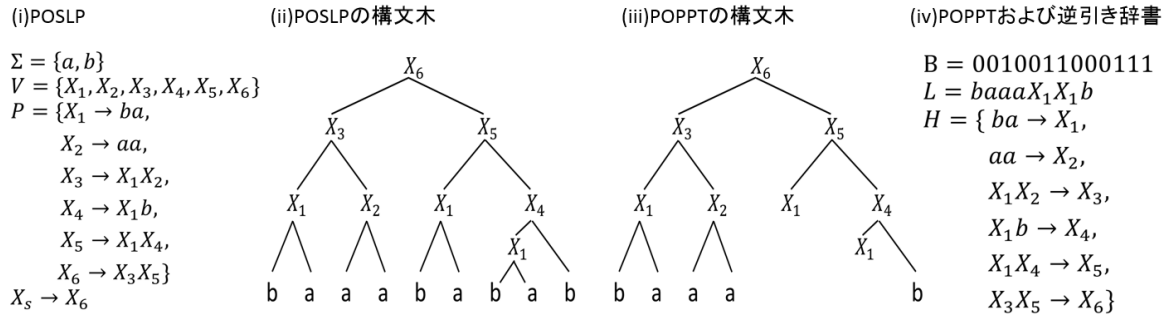


図 1: SLP と関連するデータ構造：(i)〜(iii) はそれぞれ同じ文字列を導出する構文木を表す．(iv) は (i) に対する逆引き辞書を表す．

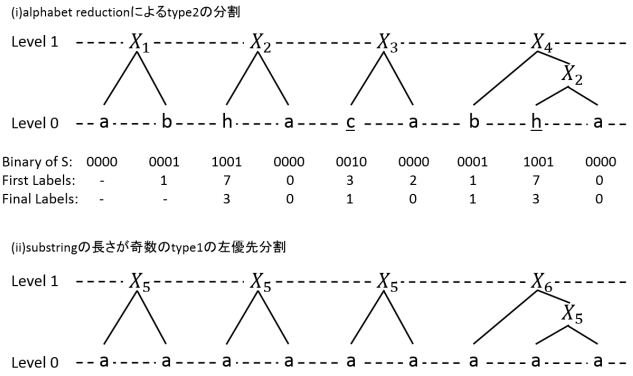


図 2: アルファベット変換による ESP 木の計算

ために各 i に対してキュー q_i を用いる．q_i は，S_i の高々 $O(\lg^* |S|)$ の部分文字列を保持し，新しい文字が FIFO にしたがって追加される．計算開始時に，入力 S が読み込まれ，いくつかの文字が q_0 に追加される．

もし S の接頭辞が Type1 すなわち a^+ (a の連続) ならば，それを左から順にペアにして生成規則 $A \to aa$ を定義し，a^+ は q_0 から削除される．そしてそれを生成する A の列が上位の q_1 に挿入される．S の接頭辞が Type3 ならば，それを同様に左から順にペアにして生成規則を定義する．それ以外のとき，すなわち S の接頭辞が Type2 であるときは，定理 2.1 により，Type2 で landmark を含まない文字列の長さは $O(\lg^* |S|)$ であるので，そこまでの文字列をやはり同様に左から順にペアにする．以上のようにして，q_i に挿入された文字列に対してペアを定義し，対応する生成規則の左辺の文字を q_{i+1} に挿入していくことで，構文木全体をメモリに保持することなく構文木の構築を q_i によって模倣できる．

次に，この生成された文法をできるだけ小さく表現するため，簡潔データ構造によって符号化する．FOLCA の辞書は succinct post order SLP(SPOSLP) と呼ばれる．POSLP とは SLP を開始記号の根とする構文木を見たとき，各非終端記号を後置順にラベリングし，一度出現した内部ノードはその子孫を枝刈りした SLP 表現である．SPOSLP は POSLP を後置順に辿り，葉ノードなら 0，内部ノードなら 1 を並べたビット配列 B と POSLP の葉ノードのラベルを後置順に並べた配列 L の組 (B, L) である．B は動的区間最大最小木 [7] によって符号化される．符号化された B と L を用いると様々な操作が可能となり，$D(X_i)$ は $O(\frac{\lg n}{\lg \lg n})$ 時間で実行できる [5].

このようにして計算される (B, L) に対して，S に出現する頻出パターン P の近似変数 X_i は以下のように発見される．S の ESP の構文木を T，その SPOSLP を $t = (B, L)$ としよう．まず，T 上で頻出な変数のみを t 上で重複なく列挙する手続きについて述べる．T の内部頂点はある生成規則 $X_i \to X_j X_k$ に対応している．また，POSLP の内部頂点は後置順にラベル付けされていることに注意しよう．したがって，t の内部頂点 i は変数 X_i が最初に出現したことを表している．よって，それ以外，すなわち L に含まれる任意の変数は 2 回目以降の変数の出現を表しているので，L に含まれている最初の出現のみを列挙すれば，L に含まれているすべての頻出な変数を重複なく列挙できる．また，枝刈りによって削除され，L に陽には出現しない頻出な変数については，$X_i \to X_j X_k$ を生成するときに，$D^{-1}(X_j X_k)$ への参照によって X_i がはじめて生成されるか否かがわかる．以上のように，$t = (B, L)$ および変数の 2 回目以降の出現かどうかをチェックするためのビット列を用いて T のすべての頻出な変数を列挙できる．

以下の定理は，この手続きで列挙されたある変数 X が，S の任意のパターン P を近似的にカバーしていることを示す．

定理 3.1. 任意のパターン P に対して P の *core* X が存在し，$|val(X)| = \Omega\left(\frac{|P|}{\lg^* |S| \lg |P|}\right)$ が成り立つ．

定理 3.2. *AFP* の近似率は $\delta = \Omega\left(\frac{1}{\lg^* |S| \lg |P|}\right)$ であり，そのアルゴリズムの時間計算量および領域計算量はそれぞれ $O\left(\frac{|S| \lg n}{\alpha \lg \lg n}\right)$ および $O(n + \lg |S|)$ である．ただし，n は *ESP* 木の変数の数であり，$\alpha \in (0, 1]$ はハッシュテーブルにおけるロードファクタである．

4 実験

環境は，OS:CentOS 5.11，Memory:144GB RAM，CPU:Intel Xeon E5504(2.00GHz，Quad Core)，Compiler:gcc4.1.2 である．文法圧縮の手法として fully-online ESP を用いた FOLCA[5] を利用し，提案手法を実装した．

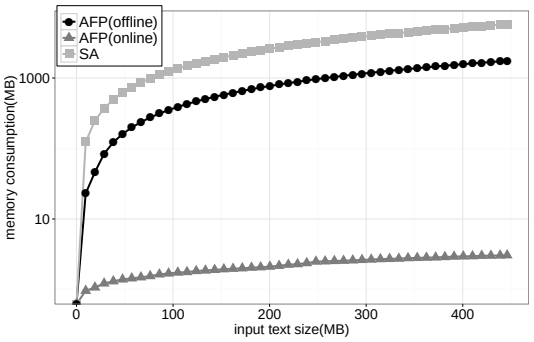
実験には，Pizza& Chili Corpus [1] で公開されているテキストデータを用いた．入力データは表 1 に示す繰り返しの多いテキストとそうでないテキストの両方を用いた．実際の近似率を得るために次のような手法を行った．SA[9] を用い，各テキスト S に対し全ての頻出パターンを取得し，頻出パターン P の長さ $|P|$ の上位 100 個を選んだ．ただし $Q \subset P$ となる頻出パターン Q は取り除いた．各頻出パターンの substring P と P に対応する core X は記録し，近似率 $\frac{|val(X)|}{|P|}$ を求めた．次に SA で得られた core の長さ，提案手法で得られた core の長さと提案手法で得られた core の近似率のそれぞれの最小，最大，平均をテキストごとに表 2 に示す．また，SA，オフラインによる手法，提案手法それぞれのメモリ消費量を図 3，実行時間を図 4 に示す．

表 1: テキストデータ

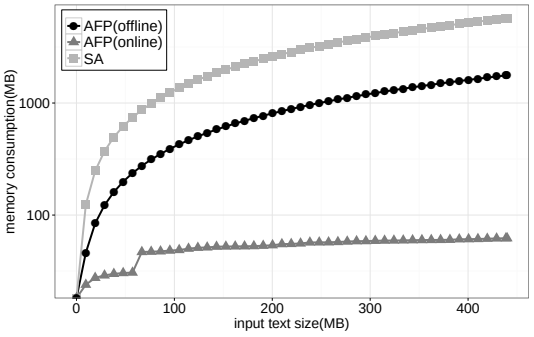
	einstein	**cere**	**english**	**dna**
テキストサイズ	446	446	200	200
文字種類数	139	5	239	16

表 2: SA と提案手法で得られた core の長さおよび core の近似率 (%)

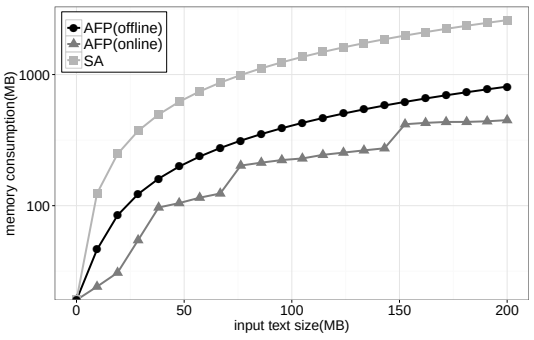
	einstein	**cere**	**english**	**dna**
	198,606	4,562	43,985	3,271
最小	18,625	4,096	3,382	268
	7.6	**2.3**	**7.3**	**7.1**
	935,920	303,204	98,7770	97,979
最大	342,136	58,906	16,1320	24,834
	50.0	**62.1**	**50.8**	**63.9**
	259,451	111,284	116,920	8,241
平均	56,584	12,723	24,703	1,926
	21.6	**11.0**	**23.0**	**22.9**



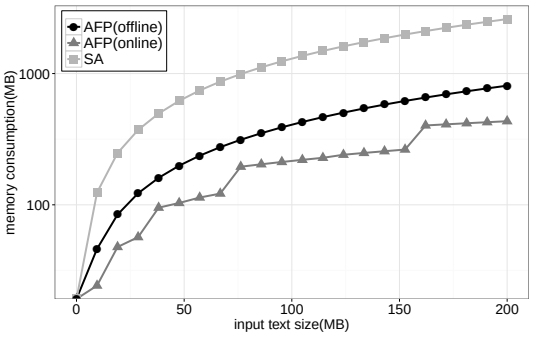
(a) einstein



(b) cere



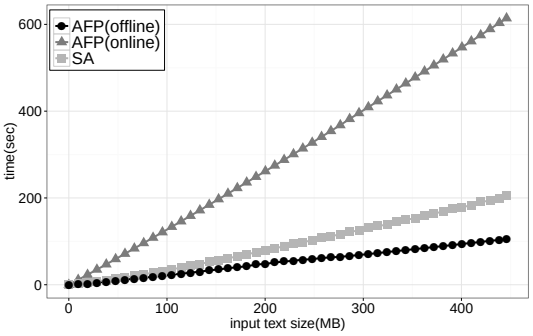
(c) english



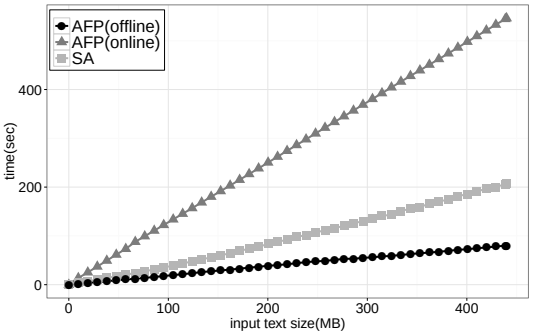
(d) dna

図 3: メモリ消費量 (MB)

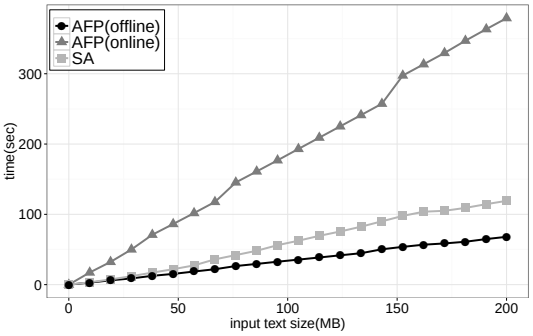
[1] http://pizzachili.dcc.uchile.cl/texts.html



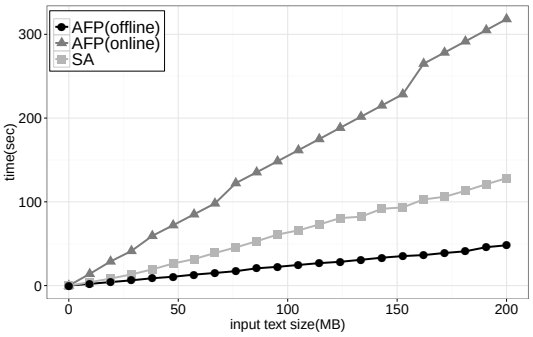
(a) einstein



(b) cere



(c) english.



(d) dna

図 4: 実行時間 (sec)

5 まとめと今後の課題

本研究では，オンライン文法圧縮を利用した頻出パターン発見を提案し，頻出パターンの近似率の下限値を改善した．実験から得られた近似率の平均は 20%前後であることがわかる．また，提案手法と SA，オフラインアルゴリズムのメモリ消費量を比較してみると提案手法が最もメモリ消費量が少ないことが分かる．特に繰り返しの多い圧縮率が良いテキストに対して顕著に差が表れている．一方で，実行時間を注目してみると他の 2 つの手法に比べ提案手法は 2 倍以上遅くなっていることが分かる．以上のことから提案手法は実行時間は比較的遅いが他 2 つの手法よりメモリ消費量が少なく，大規模データにより有効な手法といえる．

今後の課題として，実行時間の改善や本研究では最大で 446MB のデータを使用しているので，1GB を超えるより大きなデータに対してのパターン発見や実データへの応用などが考えられる．

参考文献

[1] G. Cormode and S. Muthukrishnan. *The String Edit Distance Matching Problem With Moves*, ACM Trans. Algor., Vol3, No3, pages Article 2(2007)

[2] P. Ferragina and G. Manzini and V. Mäkinen and G. Navarro. *An Alphabet-Friendly FM-index*, SPIRE, 150–160(2004)

[3] S. Fukunaga and Y. Takabatake and T. I and H. Sakamoto. Online Grammar Compression for Frequent Pattern Discovery arXiv:1607.04446.

[4] S. Maruyama and M. Nakahara and N. Kishiue and H. Sakamoto. *ESP-Index: A Compressed Index Based on Edit-Sensitive Parsing*, Journal of Discrete Algorithms, Vol.18, pages 100–112(2013)

[5] S. Maruyama, Y. Tabei, H.Sakamoto, and K. Sadakane. *Fully-online grammar compression*, In SPIRE, pages 218–229(2013)

[6] M. Nakahara and S. Maruyama and T. Kuboyama and H. Sakamoto. *Scalable Detection of Frequent Substrings by Grammar-Based Compression*, IEICE Transactions, Vol.96-D(3), pages 457–464(2013)

[7] G. Navarro and K. Sadakane. Fully-functional static and dynamic succinct trees. *ACM Transactions on Algorithms*, 10(3):Article 16, 2014.

[8] R. Raman, V. Raman and S. S. Rao. *Succinct indexable dictionaries with applications to encoding kary Trees and Multisets*, SODA,, pages 233–242(2002)

[9] K. Sadakane. *Compressed text databases with efficient query algorithms based on the compressed suffix array*, ISAAC, 410–421(2000)

[10] Y. Takabatake, Y. Tabei, and H. Sakamoto. *Improved ESP-Index: A Practical Self-Index for Highly Repetitive Texts*, *In SEA*, pages 338–350(2014)

[11] 岸山直也. *Edit-Sensitive Parsing* による圧縮索引, 九州工業大学院情報工学府 情報科学専攻　知能情報工学分野 修士論文 (2011)