用　語　解　説

Suffix Array

suffix array とは高速な文字列検索を可能にするデータ構造である。UNIX の grep コマンドのような「テキストに対するあらゆる部分文字列の検索」を高速[*1] に行なうことができる。ただし、あらかじめ検索用インデックス (= suffix array) を作成しておく必要がある。

suffix array のしくみについて説明する前に、まず suffix について説明する。suffix とは検索対象となるテキスト中のある位置から始まりテキスト末尾までの範囲の文字列である。どの suffix も開始位置が特定されれば一意に決まる。この開始位置をインデックスポイント (index point) と呼ぶ [Baeza-Yates 99]。例えば、テキスト「さくさくさくら」に対し、suffix 「さくら」 は元テキストの 5 文字目から始まるのでインデックスポイントは 5 となる (図 1)。

Text	さ	く	さ	く	さ	く	ら
Index point	1	2	3	4	5	6	7

図 1　インデックスポイント

このインデックスポイントの配列を、それぞれに対応する suffix の辞書順に従ってソートしたものが suffix array である。例えば、図 1 のインデックスポイントの配列「1 2 3 4 5 6 7」を対応する suffix でソートすると、「2 4 6 1 3 5 7」となる (図 2)。この配列が suffix array である。

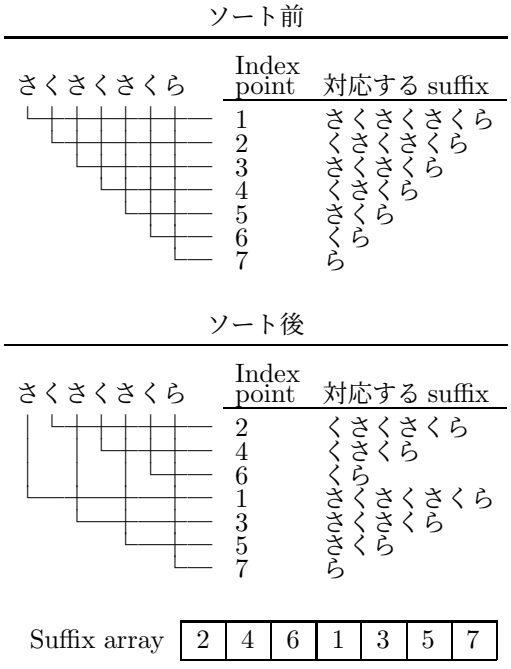

図 2　Suffix array の作り方

文字列の検索には二分探索 (binary search) を用いる。図 3 に検索キー「くさくさ」で検索する例をあげる。二分探索は中心の要素と検索キーを比較して検索範囲を狭めていく探索手法である。まず、suffix array の中心 (4 番目) であるインデックスポイント 1 に対応する suffix「さくさくさくら」と検索キー「くさくさ」を比較する (phase 1)。辞書順で考えると「くさくさ」が小さいので、suffix array の中心より前半分 (1 番目から 3 番目) に検索範囲が絞られる。次に、その前半分の中心であるインデックスポイント 4 の suffix「くさくら」と検索キーを比較し (phase 2)、また検索範囲を絞る。このようにして、インデックスポイント 2 が最終的な検索結果となる (phase 3)。

Phase	Suffix array	対応する suffix
3 →	2	**くさくさ**くら
2 ⟶	4	くさくら
	6	くら
1 ⟶⟶	1	さくさくさくら
	3	さくさくら
	5	さくら
	7	ら

図 3　Suffix array による文字列検索

このように、suffix array は非常に単純な方法ということもあり、1970 年代にはすでに使われている [Bentley 00]。1990 年になり Manbar [Manber 90] により suffix array と命名された。suffix array についての教科書的な文献として [Baeza-Yates 99] をあげておく。巨大な suffix array の作成方法、圧縮、高速化、正規表現、転置インデックスとの比較など、関連する話題が言及されている。また、suffix array を使用したフリーの高速文字列検索ライブラリとして、奈良先端科学技術大学院大学情報科学研究科自然言語処理学講座から SUFARY [*2] が公開されている。

最後に、suffix array の特徴を簡単にまとめておく。

- どんな部分文字列でも検索可能。日本語テキストへのインデクキシングで、形態素解析などの単語分割処理が必要無い。
- しくみが単純なので実装が簡単。
- 検索時に必ず元テキストが必要。WWW サーチエンジンには不向き。サイト内検索ならば問題無い。
- suffix array のサイズはインデックスポイントの数、つまりファイルサイズに比例する。

◇ 参　考　文　献 ◇

[Baeza-Yates 99]　Baeza-Yates, R. and Rieiro-Neto, B.: *Modern Information Retrieval*, ACM Press/Addison Wesley (1999).

[Bentley 00]　Bentley, J.: *Programming Pearls*, ACM Press/Addison Wesley, second edition (2000).

[Manber 90]　Manber, U. and Myers, G.: Suffix arrays: A new method for on-line string searches, in *1st ACM-SIAM Symposium on Discrete Algorithms*, pp. 319–327 (1990).

〔山下達雄（富士通研究所）〕

*1　二分探索を用いるので、計算量は $O(\log(n))$: n = テキストサイズ。grep のようなテキスト全体を走査する方法では $O(n)$ かかる。巨大なファイルの検索において、その検索速度の差が顕著になる。

*2　http://cl.aist-nara.ac.jp/lab/nlt/ss/