D-020

プロパティ接尾辞木: メタデータ付き系列データのための効率よい索引構造
Property Suffix Tree: An Efficient Data Structure for Stream Data with Meta Data

上村 卓史* 喜田 拓也* 有村 博紀*
Takashi Uemura Takuya Kida Hiroki Arimura

1. 背景

テキスト処理において，特定の条件を満たす区間だけを対象として文字列照合を行う場合がある．このような区間の集合をプロパティと呼ぶ．**プロパティ付きテキスト照合問題**とは，テキスト T とプロパティ π に対して，与えられたパターン P の T 中の出現位置の内で，π の区間に含まれるもの全てを求める問題である．

一方で，大規模な検索システムにおいては，同一のテキストに対して，多くのパターンの照合を高速に行いたいという要求がある．接尾辞木 [3] は全文検索を可能にするよく知られたデータ構造である．接尾辞木の変種としては，高々数単語程度の部分文字列のみを効率的に扱う木構造 [4] 等が提案されている．

近年，Amir ら [1] は接尾辞木を基に，プロパティ付きテキストに対して理論的に最適な検索時間を実現する索引構造である**プロパティ接尾辞木**を提案した．定数サイズのアルファベット上の長さ n のテキストに対し，そのプロパティ接尾辞木は，$O(n)$ 領域のデータ構造である．Amir らは，プロパティ付きテキストから，プロパティ接尾辞木をオフラインに $O(n \log\log n)$ 時間で構築するアルゴリズムを与えている [1, 2]．本稿では，Amir らのアルゴリズムを改良して，プロパティ接尾辞木を $O(n)$ 時間で構築するアルゴリズムを示す．

2. 準備

アルファベットを Σ とする．以降では Σ は定数サイズであると仮定する．Σ 上の文字列全体の集合を Σ^* で表す．文字列 $x \in \Sigma^*$ について，x の長さを $|x|$ で表す．長さが 0 の文字列を**空語**といい，ε で表す．

ある文字列 $w \in \Sigma^*$ に対して，$w = xyz$ となる $x, y, z \in \Sigma^*$ が存在するとき，x, y, z をそれぞれ w の**接頭辞**，**部分文字列**，**接尾辞**と呼ぶ．w の i 番目の文字を $w[i]$ と表し，w の i 番目から j 番目までの部分文字列 x を $w[i \ldots j]$ で表す．$i > j$ のときは $x = \varepsilon$ と定義する．文字列 $w \in \Sigma^*$ の接頭辞全体の集合を $Pre(w) = \{w[1 \ldots i] | 1 \leq i \leq |w|\}$ で表す．また同様に，接尾辞全体の集合を $Suf(w) = \{w[1 \ldots i] | 1 \leq i \leq |w|\}$ で表す．

プロパティ付きテキストとは，長さ n の文字列 T とプロパティ π の組 $I = (T, \pi)$ である．T の**プロパティ** π とは，集合 $\pi = \{(s_1, f_1), \ldots, (s_m, f_m)\}$ であり，任意の $1 \leq i \leq m$ について，**区間** $(s_i, f_i) \in \pi$ は $1 \leq s_i \leq f_i \leq n$ を満たす．ここで，(i) 任意の $1 \leq i \leq n$ について，$s = i$ となるような区間 $(s, f) \in \pi$ は高々一つしか存在せず，(ii) $s_1 < s_2 < \cdots < s_m$ であるとき，π は**標準形**であるという．本稿では標準形のプロパティのみを扱う．

長さ $n \geq 1$ のテキスト T を，$T[1 \ldots n-1]$ に現れない終端記号 \$ で終わる文字列と仮定する．T の**接尾辞木**は根付き木 $ST(T) = (V, root, E, suf)$ である．ここで，$root$ は木の根であり，V はノードの集合である．E は辺の集合で，$E \subseteq V^2$ である．ノード s, t に対し，$e = (s, t)$ となる辺 $e \in E$ が存在するとき，s を t の**親**と呼び，t を s の**子**と呼ぶ．子を持たないノードを**葉**と呼ぶ．葉でない任意のノードを**内部ノード**と呼ぶ．葉全体の集合は $Suf(T)$ と 1 対 1 で対応する．任意の辺 $e \in E$ には空でないラベル文字列が付加される．任意の内部ノードは少なくとも 2 つの子を持ち，子に接続する全ての辺はそれぞれ異なる文字で始まるラベル文字列を持つ．任意のノード $s \in V$ に対し，s の**表す文字列**は，$root$ から s までの道の辺のラベル文字列を順に連結した文字列であり，$\overline{s}$ と書く．任意の内部ノード $s \in V$ について，$\overline{s} = a \cdot \overline{t}$ となる内部ノード $t \in V$ が存在する [3]．s から t へのリンク $t = suf(s)$ を**接尾辞リンク**と呼ぶ．

*北海道大学大学院情報科学研究科, Hokkaido University

手続き Find-border$(ST(T), \{end(1), \ldots, end(n)\})$;
入力: 接尾辞木 $ST(T)$，終了位置 $\{end(1), \ldots, end(n)\}$
1 根から $T[1 \ldots end(1)]$ の各文字で辺をたどり，$bd(1)$ を求める
2 **for** $i = 2$ **to** n
3 　接尾辞リンクをたどり，$T[i \ldots end(i-1)]$ に移動する
4 　$T[end(i-1)+1 \ldots end(i)]$ で辺をたどり，$bd(i)$ を求める

図 1: 全ての境界ノードをたどる手続き．

3. プロパティ接尾辞木

本節では，Amir ら [1] に従って，プロパティ接尾辞木を導入する．長さ n のプロパティ付きテキスト $I = (T, \pi)$ に対する**プロパティ接尾辞木** $PST(T, \pi)$ は，T の接尾辞木 $ST(T) = (V, root, E, suf)$ から，辺の除去と併合，および補助情報の付加を行って得られる根付き木 $PST(I) = (V', root, E', end, pos)$ である．$V' \subseteq V$ はノードの集合であり，$E' \subseteq V'^2$ は辺の集合である．

T 上の任意の位置 $1 \leq i \leq n$ に対して，**終了位置** $end(i)$ と書き，次のように定義する: $end(i)$ は $\sigma = (s, f) \in \pi$ かつ $s \leq i \leq f$ を満たす最大の終了位置 f である．上の条件を満たす σ が存在しない場合は，$end(i) = i - 1$ と定義する．次の補題は，構築手続きの正当性に関して重要である．

補題 1 長さ n の任意のプロパティ付きテキスト $I = (T, \pi)$ について，$end(1) \leq \cdots \leq end(n)$ が成立する．

接尾辞木のノード $s \in V$ に対し，s に**属する文字列**を $X(s) = Pre(s) \setminus Pre(t)$ と定義する．ここで，t は s の親である．位置 i から始まる T の**プロパティ接尾辞**とは，$T^i_\pi = T[i \ldots end(i)]$ である．任意のノード $s \in V'$ に対して，**添え字集合関数** pos は，$pos(s) = \{1 \leq i \leq n | T^i_\pi \in X(s)\}$ である．文献 [1] では，全てのノード $s \in V'$ に対する $pos(s)$ の部分的なソートを行う方法を述べている．この前処理を行うことで以下の定理が成り立つ．

定理 1 (*Amir,et al.,2006*) プロパティ付きテキスト $I = (T, \pi)$ と長さ m のパターン P に対し，プロパティ接尾

辞木 $PST(I)$ によって，$s \leq i$，$i+m-1 < k$ かつ $P = T[i \ldots i+m-1]$ である区間 $(s,k) \in \pi$ が存在する全ての位置 $i_1, \ldots, i_{tocc}$ を $O(m + tocc)$ 時間で求められる．

テキスト T の接尾辞木 $ST(T)$ について，任意の位置 $1 \leq i \leq n$ に対する**境界ノード** $bd(i)$ とは，$i \in pos(s)$ であるノード $s \in V$ である．これを用いて V' と E' を定義する．まず $root$ から全ての境界ノードまでの道に存在するノードを残し，その他の全てのノードと，そのノードに接続する全ての辺を接尾辞木から除去する．次に，子を一つしか持たない内部ノードをすべて除去し，それに接続する辺を併合する．こうして得られた木のノードの集合と辺の集合を V' と E' とおく．

4. 構築アルゴリズム

本稿で提案する境界ノードの計算アルゴリズムについて述べる．この手続きでは，任意の時点 $1 < i \leq n$ について前回計算した境界ノード $bd(i-1)$ を保持し，接尾辞リンクを用いて，根からの道を全てたどらずに次の境界ノードを計算する．補題 1 から，T_π^{i-1} と T_π^i の差異は，先頭の文字 $T[i-1]$ と，末尾の $T[end(i-1)+1 \ldots end(i)]$ である．$\overline{bd(i-1)} = T_\pi^{i-1}$ ならば，接尾辞リンクを用いて，$suf(bd(i-1))$ から末尾の差分の文字列 $T[end(i-1)+1 \ldots end(i)]$ をたどることで $bd(i)$ が求められる．$\overline{bd(i-1)} \neq T_\pi^{i-1}$ の場合，$\overline{bd(i-1)}$ の親 s の接尾辞リンクを使い，$T[i \ldots end(i-1)]$ の接頭辞を表すノードに移動し，さらに残りの文字列で辺をたどることで，$T[i \ldots end(i-1)]$ を表すノードが求められ，結局 $\overline{bd(i-1)} = T_\pi^{i-1}$ の場合と同様に $bd(i)$ が求められる．

次の補題は，提案アルゴリズムがテキスト長に比例した時間で動作することを示すために重要である．

補題 2 長さ n のプロパティ付きテキスト $I = (T, \pi)$ と接尾辞木 $ST(T)$ が与えられたとき，図 1 の手続き **Findborder** は，合計で高々 $2n$ 個のノードをたどる．

ノードの除去等，その他の手続きに関しては [1] の方法を用いることで，次の定理が得られる．

定理 2 長さ n のプロパティ付きテキスト $I = (T, \pi)$ に対して，$O(n)$ 時間でプロパティ付き接尾辞木 $PST(I)$ を構築するアルゴリズムが存在する．

5. 実験

前節で与えたプロパティ接尾辞木構築アルゴリズムを実装し，人工データを用いて計算機実験を行った．本節では，根や葉から境界ノードを線形探索する手法ではテキスト長の自乗時間かかる例を実験的に示す．

データは，長さ n のテキスト $T = AA \cdots A\$$ と，プロパティ $\pi = \{(s, s + \lceil \frac{(n-i)}{2} \rceil) | 1 \leq s \leq n\}$ とした．

この実験では，接尾辞木を構築した後の，全ての境界ノードを計算する部分のみの実行時間を測定する．システムは Intel Pentium M 1.3GHz, 1GB RAM, Windows XP 搭載の PC である．コンパイラは Visual C++ 2005 付属のものを用いた．アルゴリズムは，提案手法のほか，比較のため，各境界ノードを毎回根から探索する手法，および毎回葉から探索する手法を実装した．これらの手法をそれぞれ，suf，$root$，$leaf$ と呼ぶことにする．

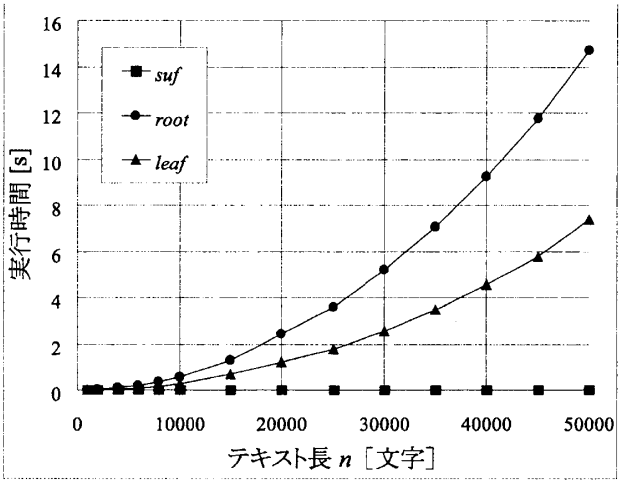

図 2: T =AA···A\$ に対する実行時間の測定．

結果を図 2 に示す．$root$ と $leaf$ ではテキスト長の増加分以上に実行時間が増えているのに対し，suf ではテキスト長にほぼ比例した時間で実行された．この実験では，根から接尾辞 $T[i \ldots n]$ を表す葉までのノードの道の長さが $n-i$ となり，さらに境界ノードはこの道の中間に位置する．したがって，根からも葉からも毎回平均 $O(n)$ 個の実ノードをたどって境界ノードを見つける必要があるため，このような結果になったと考えられる．

6. 結論と今後の課題

本稿では，プロパティ接尾辞木の構築において，接尾辞木から不要なノードを削除する際，その境界となる全ての位置を $O(n)$ 時間で計算する手法を提案し，プロパティ接尾辞木の $O(n)$ 時間構築が可能であることを示した．更なるアルゴリズムの改良として，接尾辞木を作らずに，与えられたプロパティ付き文字列からプロパティ付き接尾辞木を直接オンライン構築する手法が望まれる．

また，実データに対する実験が必要である．例えば，サーバのログに対し，クライアントの接続している時間をプロパティの区間としたり，もしくは負荷の高い時間帯を区間として，解析を行うという応用が考えられる．

参考文献

[1] A. Amir, E. Chencinski, C. Iliopoulos, T. Kopelowitz, and H. Zhang, "Property matching and weighted matching, " Proc. of the 17th Annual Symposium on Combinatorial Pattern Matching, LNCS4009, 188–199, Barcelona, Spain, July 2006.

[2] A. Amir, D. Keselman, G.M. Landau, M. Lewenstein, N. Lewenstein, and M. Rodeh, "Text indexing and dictionary matching with one error, " Proc. of the Journal of Algorithms, vol.37, no.2, 309–325, 2000.

[3] E. Ukkonen, "On-line construction of suffix trees, " Algorithmica, vol.14, no.3, 249–260, 1995.

[4] 上村卓史，喜田拓也，有村博紀，"単語幅を制約した接尾辞木の効率のよい構築アルゴリズム，" 情報科学技術レターズ，vol.5, 5–8, 2006.