学　位　論　文　内　容　の　要　旨

博士の専攻分野の名称　　博士（情報科学）　　氏名　髙木　拓也

学　位　論　文　題　名

Studies on Efficient Index Construction for Multiple and Repetitive Texts
（複数テキストと繰り返しテキストに対する効率の良い索引構築の研究）

センサー技術の発展により, 観測値や測定値, 行動ログ, ゲノム情報などの多様で膨大なデータが生み出されており, これら大規模データの利活用が緊急の課題となっている. 大規模データの中で最も基礎的なデータ形式は, 記号の連結により表現されるテキストデータである. 実際, 上記の観測値や測定値, 行動ログ, ゲノム情報はテキストデータとして表現される. 限られたメモリと計算時間を用いて, 大規模データから有用な情報を取り出すためには, 前処理によって問合せ (クエリ) への高速な応答を可能にした索引構造が鍵となる. そこで本研究ではテキストデータに対する索引構造の研究を行う.

テキストに関する索引の中でも, 全文索引と呼ばれる索引は与えられたテキストに含まれる全ての部分文字列を表現するデータ構造である. これはパターン照合 (クエリ文字列がテキスト中に出現するかを答える存在問合せや, 出現回数を答える回数問合せ, 出現する位置を全て返す位置問合せがある) だけでなく, いくつかの誤りを許す近似マッチングや, テキストの圧縮表現の計算, DNA アライメント, データマイニングなど様々な問合せや応用をサポートする. ウァイナーは 1973 年に接尾辞木と呼ばれる全文索引を提案した. これは元のテキスト長に対して線形個のポインタで表現される木構造 (枝にラベル文字列がついており, コンパクトトライと呼ばれる) であり, これ以降に開発される全文索引の中でも最も多様な問合せをサポートするという特徴がある. 1995 年にウッコネンは接尾辞木のオンライン構築アルゴリズムを提案した. すなわち文字列の末尾に文字が追加されるごとに接尾辞木を更新することができる. 具体的には, 文字の集合であるアルファベット (サイズを σ とする) 上の要素を文字列の末尾に追加すると, 与えられた文字列に対する接尾辞木をならし $O(\log \sigma)$ 時間で更新することができる. すなわち, 最終的な長さが n のテキストに対しては接尾辞木を $O(n \log \sigma)$ 時間で構築可能である.

本論文第 3 章では全文索引の課題の 1 つであるクエリ時間の改善について考察する. 文字列に対するアルゴリズムとデータ構造の分野では Word RAM と呼ばれる計算モデルを仮定することが標準となっている. Word RAM モデルにおいては, コンピュータはサイズ $\Theta(\log n)$ のレジスタ (Word) を持ち, レジスタ上の演算は定数時間で行えることを仮定する. ここで n はメモリ上に置かれる入力データのサイズである. Word RAM を仮定した際に, 2011 年にベンキキらはパターン照合をおおよそレジスタ倍高速化する手法を提案した. また, ジャンソンらは 2007 年にトライ木と呼ばれるデータ構造に対しての高速化を実現している. しかし, 接尾辞木に対する Word RAM 上での高速化手法は提案されていない. そこで接尾辞木の基となるデータ構造であるコンパクトトライにおける各種クエリ (パターン照合, 挿入, 削除) を, おおよそレジスタ倍高速化する手法を提案する. これは 2011 年にベンキキらが提案した文字列詰込み技法と順序辞書と呼ばれるデータ構造を組み合わせ, 木構造を深さレジスタ長毎に区切ることで実現する. これにより, コンパクトトライの各クエリだけでなく, コンパクトトライを用いる様々な応用, 例えば, スパース接尾辞木と呼ばれる接尾辞木の

1 種のオンライン構築や LZD と呼ばれる圧縮アルゴリズムを高速に計算することが可能になる.

第 4 章では文字列集合に対する索引のオンライン構築アルゴリズムを与える. センサーデータは, 複数のセンサーがデータを逐次的に送信するため, 全文索引を構築するためには文字列集合に対するオンライン構築が必要になる. 1 本の文字列ではなく, 文字列集合を 1 つの接尾辞木で表現した全文索引は一般化接尾辞木と呼ばれる. ウッコネンにより提案された接尾辞木のオンライン構築アルゴリズムは, 与えられる文字列が 1 本であることを仮定している. 文字列集合に対するオンライン構築 (一般化接尾辞木のオンライン構築) では, 1997 年にガスフィールドによって複数の文字列を区切り文字を挟みながら連結し 1 本の文字列と見做すことでウッコネンのアルゴリズムが適応可能であることを示した. しかしながらこの仮定では入力文字の到着順序に制限がある. 具体的には, まず 1 つ目の文字列が伸長し, 文字の追加が何かしらの方法で終わったと判定したとき, 1 つ目の文字列の末尾に区切り文字を追加して, 2 つ目の文字列の伸長が可能になる. すなわち i 番目の文字列を更新すると, 1 番目から $i-1$ 番目の文字列は更新不可能になってしまう. 現実世界のセンサーデータは, 複数設置された各センサーが非同期にデータを送信するため, この仮定を適用することはできない. 文字列集合の各文字列が制限なく伸長するときに接尾辞木をオンライン構築する問題は, ウァイナーが接尾辞木を提案してから約 30 年の未解決問題であった. 4 章では, この問題を完全オンライン構築問題と呼び, 肯定的に解決する. 本研究では, 有向非巡回語グラフ (Directed acyclic word graph, DAWG と略す) と呼ばれる全文索引と接尾辞木の関係に着目することで, まず DAWG のオンライン構築アルゴリズムと文字列の先頭に文字が追加される状況での接尾辞木 (逆接尾辞木) のオンライン構築 (right-to-left online construction) が同一であることを示し, 両者の完全オンライン構築アルゴリズムを提案する. その後, DAWG(逆接尾辞木) の更新情報を用いて文字が文字列の末尾に追加される場合の接尾辞木の完全オンライン構築を示す.

第 5 章では元文字列を陽に持つ必要がない圧縮索引構造を提案する. 接尾辞木に代表される全文索引の最大の問題点は, その領域が元のテキストに対して大きいことである. 元の文字列が n 文字, すなわち $O(n \log \sigma)$ ビットの時, 接尾辞木は n 個のポインタ, すなわち $O(n \log n)$ ビット必要になる. 現実的には元テキストと比べて接尾辞木は 12 から 20 倍程度大きくなってしまう. これに対して, ブルーマーらは 1985 年に接尾辞木の同型な部分木をまとめた DAG 構造であるパス圧縮有向非巡回語グラフ (Compacted directed acyclic word graph, CDAWG と略す) を提案した. これは文字列の全ての部分文字列を受理する最小のオートマトンになっている. CDAWG の枝数は与えられた文字列に繰り返しが多い時 n より小さくなることが知られており, 最小の場合 $O(\log n)$ 程度になる. 具体的には, CDAWG の枝数は与えられた文字列の極大繰り返しの右拡張数と呼ばれる数に等しい. しかしながら, CDAWG の枝数が n より少なくなったとしてもラベル文字列への間接参照に必要なため元文字列を保持する必要があり, CDAWG 全体としては $n \log \sigma + O(e_r \log n)$ ビット必要になる. ここで e_r は CDAWG の枝数である. 第 5 章では, 明示的に元文字列を持たないのにも関わらず CDAWG での各種操作と元文字列へのアクセスを可能にする索引を提案する. これは self-index と呼ばれ, 2010 年ごろから盛んに研究されている. 結果として, $O((e_r + e_l) \log n)$ ビットで CDAWG を表現するとこに成功した. ここで e_l は与えられた文字列 T の逆文字列に対する CDAWG のサイズである.

最後に第 6 章ではいくつかの未解決問題を示し, 本研究の今後の方向を与える.