大規模文字列データベースに対する索引方式の考察
Study on an indexing method for large-scale string databases

グエン　フーン　バック*　　高橋　誉文*　　田村　慶一**　　北上　始**
Nguyen Phuong Bac　　Yoshifumi Takahashi　　Keiichi Tamura　　Hajime Kitakami
広島市立大学大学院情報科学研究科
*{mt67022,dt65002}@edu.ipc.hiroshima-cu.ac.jp　　**{ktamura,kitakami}@hiroshima-cu.ac.jp

Abstract— **In order to achieve high-speed searches on a large-scale string database, we implement an indexing system based on DynaCluster algorithm that constructs an index structure, called a suffix tree on disk, and evaluate performance of the system. In experiments conducted in this study, we have created an indexing structure for the database with 223MB of DNA sequences. The experiments resulted in achieving high performance related to query processing time.**

I. はじめに

現在，テキストデータや分子配列データなどを対象にした文字列データベースの高速な質問検索を達成するために，様々な索引構造の構築手法が考えられている．著者らは，文字列データとして，DNA やタンパク質などの分子配列データに着目している．ヒトの DNA は，30 億個も存在すると言われているが，これらはヒトゲノム計画によって 2003 年に全て解読され，現在，DDBJ などを初めとする DNA データバンクに保管されている．しかしながら，DNA 配列データベースは大規模であるため，高速検索を実現する索引構造の研究が重要となっている．

本研究では，索引構造の実現手段として，線形時間で構築が可能で，定数時間で検索が可能なサフィックス木に注目する．また，DNA のデータは大規模な配列データであるため，OS が提供する仮想メモリ上にサフィックス木を構築すると，実メモリとディスクとの間のページ I/O が多発し，大規模的な索引構造の構築が困難になるという問題がある．この問題に対処するため，我々は，バッファ管理モジュールを開発し，それを経由して，実メモリとディスクとの間のデータページのやりとりが可能なディスク版のサフィックス木を構築する．なお，サフィックス木の構築には，DynaCluster アルゴリズムを採用している．

DynaCluster アルゴリズム[1]は，Cheung らにより 2005 年に提案されたサフィックス木構築アルゴリズムである．彼らによる DynaCluster の実証実験では，小規模データに対しての有効性が確認されてはいるものの，ゲノムスケールの文字列データベースで見られるような大規模データに対する有効性の確認が未だなされていない．

本論文では，DynaCluster アルゴリズムに基づいて，サフィックス木構築プログラムとバッファ管理モジュールのプログラムを作成し，ヒトの DNA 配列データが含まれている大規模文字列データベースを用いて，ディスク上の索引構造の構築や検索の性能について実験的に考察する．

II. DynaCluster

本章では，2005 年に公開された DynaCluster アルゴリズムを概観するために，DynaCluster アルゴリズムについて，最初に，用語の定義を行った後，アルゴリズムの特徴，サフィックス木の構築例について説明する．

A. 用語の定義

DNA の文字列は四つの A, C, G, T の文字から構成されるが，サフィクス木の構築に際しては，それらのデータを符号化する．それぞれ文字を符号化したときのコードは c(A) = 0, c(C) = 1, c(G) = 2, c(T) = 3 とする．Σ は文字列を構成する文字種の集合である．DNA の文字列は四つの文字で構成されている，すなわち，Σ={A,T,C,G} なので，|Σ|は 4 である．

・位置情報: サフィックス木のあるノードにおいて，ルートノードからそのノードまで辿ったラベルはサフィックスのプレフィックス部分である．その部分が入力文字列の中にどこにあるかを表す情報をそのノードの位置情報と呼ぶ．ノードの位置情報は，入力文字列の先頭からのオフセットである．

・閾値: サフィックス木の構築途中において，ある親ノードが作成されたとき，そのすべての子ノードについての，位置情報が列挙される．DynaCluster アルゴリズムでは，ある閾値 τ を設け，位置情報の数が閾値 τ より小さければ，親ノードの子ノードを作成しないことにしている．したがって，閾値 τ により，サフィックス木の深さを制限することができるが，サフィックス木を利用した文字列検索を行う際，検索コストに影響するので適切な閾値 τ を選ぶ必要がある．なお，DynaCluster の論文では，性能上，閾値 τ = 1024 が最適であると報告されている．

・接頭辞コード: 構築されるサフィックス木の深さとノードの種類をユーザが自由に制御できるようにするために，任意長の文字列を単位とする接頭辞に基づく符号化を可能にしている．この符号化により得られる整数は，接尾辞コード（整数）と呼ばれ，以下の式で計算されている．

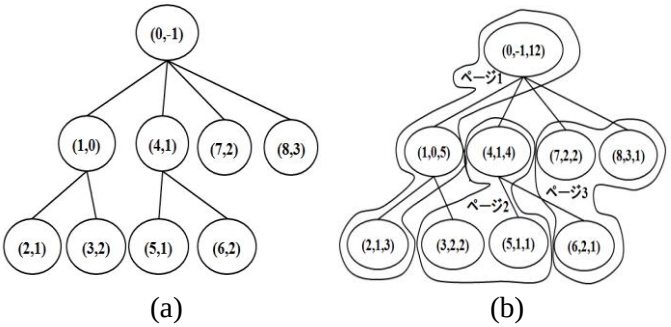
$$P(S_i^l) = \sum_{j=0}^{l-1} c(s_{i+j}).|\Sigma|^{l-j-1} \qquad (1)$$

ここで $P(S_i^l)$ はサフィックス S_i の接頭辞コードを表しており，そのコードは，接頭辞の長さが l の部分列を符号化することにより得られる．$c(s_{i+j})$ は，サフィックス S_i の接頭辞の長さが l の部分列について，先頭から j 番目に位置する文字に対応するコードである．

B. アルゴリズムの特徴

DynaCluster アルゴリズムでは，サフィックス木を作成するために，入力データの接尾部の集合からトライ構造を構築するアプローチを採用しておらず，*Pattern Growth* アプローチと呼ばれる Han らのデータマイニングアルゴリズム[2]を利用し，支持数を 1 とした列挙木を作成することによりサフィックス木を構築している．このため，以下の特徴をもつ．

- 深さ優先に親ノードから子ノードを作成することで，構築が進められている．
- 兄弟ノードの作成は，接頭辞コードに関して昇順となるように実施されている．
- ある親ノードが作成されたとき，その子ノードの位置情報も計算される．
- ある親ノードの子ノードを作成するか否かの判定は，作成される子ノードの位置情報の数が閾値 τ を超えるか否かにより実施している．すなわち，子ノードに関する位置情報の数が閾値 τ より大きければ親ノードの子ノードを作成している．そうでなければ，その親ノードを頂点とする部分木の作成を打ち切る．
- このアルゴリズムでは，動的クラスタリング技術を用いており，木を構築する際，同じ親ノードを持つような子ノードなどは距離の近いノードであることから，それらのノードは同ページ，もしくは隣接したページに格納され，ノード挿入時に起こる edge-splitting と呼ばれる枝の分割によるディスクページアクセス数の増加を抑える工夫がなされている．



(a) (b)

図 1. サフィックス木の比較
(a)は DynaCluster のサフィクス木であり，(b)は本研究で構築されるサフィックス木である．

C. サフィックス木の構築例

例として，二つの文字列 *S* = <AGACAC>, K = <TCAGTC>を与えて，DynaCluster により構築されるサフィックス木を以下に示す．

図 1(a)は，閾値 τ を 2 とし，接頭辞の長さを 1 としたときに構築されるサフィックス木を示す．図中の各ノードには，ノードが作成された順番(Node Index)，およびノードの接頭辞コードが保存されている．図 1(a)に表記されている(1, 0)は，ノードの作成順番が 1, ノードの接頭辞コードが 0 であることを表現している．なお，ルートノードの接頭辞コードは−1 としている．

III. サフィックス木の実装方式

本節では DynaCluster アルゴリズムに基づき，大規模な文字列データに対してサフィックス木を実装する方法について説明する．また，部分文字列の出現回数の求め方やバッファ管理モジュールの実現方式について説明する．

A. 大規模データ処理方式

実際の DNA 文字列データでは，一つの文字列が極めて長いので，C 言語の String 型を使うことができない．我々の実装では，大規模な文字列をすべてメモリ上に常駐せずに，後述するバッファ管理モジュールを用いて，メモリとファイルとの間でデータのやりとりを実施しながら処理を行っている．

3GB の容量を持つヒトの DNA 配列データの利用について考えてみよう．ルートノードで一時的記憶される位置情報は，深さ 1 のノードを作成するときのスキャン開始点として利用される．したがって，位置情報の活用は， DNA 配列データの先頭から繰り返しスキャンするのを回避するのに役立つ．しかし，その位置情報を一時記憶するのに 24GB（ = 8byte $\times 3 \times 10^9$）ものメモリを必要とする．ただし，位置情報を表現するオフセットデータとしては，8bytes の long int 型を利用する．符号長 l を 4 とすると，深さが 1 のノードで一時記憶する位置情報は，平均で, 24GB/256 = 95MB が必要となる．また，深さ 2 のノードの位置情報については，平均で, 95MB/ 256 = 384KB が必要となる．

以上により，サフィックス木の構築中に，ルートノードおよび深さ 1 のノードにおける位置情報はメモリ上に一時記憶するには大きすぎると判断している．従って，我々は，それらのノードにおける位置情報の作成を止め，位置情報の作成および一時記憶は深さ 2 以降のノードにおいて実施している．3GB を遥かに超える文字列データについては，位置情報を持たせないノードの深さを適切に選択すれば，サフィックス木の構築時間は延びるが，メモリ不足による処理性能の劣化を大幅に回避できると考えている．

B. 出現回数の計算と保存

ある部分文字列の出現回数とは，その部分文字列が，文字列データベースに出現する回数をさす．ルートノードからある中間ノードに対応

する部分文字列は，両ノード間の経路上に現れる文字の並びである．その部分文字列の出現回数は，その中間ノードがもつ位置情報の数に相当する．我々の実装では，サフィックス木の各ノードには，子ノードの列挙（サフィクス木の成長）に必要な位置情報から出現回数を計算し，図 1(b)に示されるように，作成されたノードに計算された出現回数を保管している．例えば，図 1(b)に表記されている(1, 0, 5)は，ノードが作成された順番が 1, ノードの接頭辞コードが 0, 出現回数が 5 であることを表現している．

C. バッファ管理モジュール

実メモリとディスクとの間でデータのやりとりをしながら，サフィックス木をディスク上に構築するためには，サフィックス木を構成するノードの情報をファイルに書き込んだり，読み込んだりしなければならない．実メモリとディスクとの間でデータのやりとりを OS が提供する仮想記憶にまかせると，I/O コストが大きくなり，サフィックス木の構築に多大な時間を要する．我々は，I/O コストを削減するために，図 2 に示されるバッファ管理モジュールを作成している．

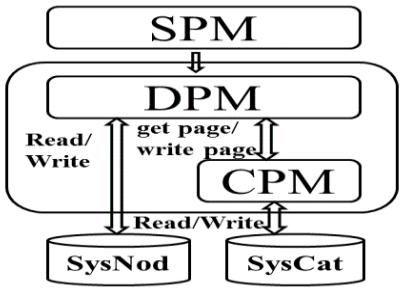


図 2. バッファ管理モジュールの構成

図中の STM(Suffix Tree Module)は，サフィックス木を構築するモジュールであり，このモジュールからバッファ管理モジュールが利用される．バッファ管理モジュールは，DPM, CPM から構成されている．DPM(Data Page Manager Module)は，データページ管理モジュール，CPM(Catalog Page Manager Module)はカタログページ管理モジュールである．ノードデータは，ページ単位で SysNod と呼ばれるファイルに格納される．ノードデータを格納するページの管理情報は，ページ単位で SysCat と呼ばれるファイルで管理される．

DPM で用意されたバッファページが不足すると，DPM で使われていないページをディスク上のファイルへ書き込み，新たに利用可能なバッファページを用意している．

IV. 評価実験

本章では，先ず，閾値 τ，符号長 l を変化させ，適切な閾値と符号長をさがし，次に，それらを用いて，入力データのサイズを増加させながら，構築時間を測定する．最後に，今回の実験で最大サイズとなった入力データを取り上げ，それに対するサフィックス木を用いて検索性能の評価を行う．本評価実験での実験環境は，表 1 に，示すとおりである．

表 1　実験環境

CPU	Intel Xeon E5420(2.50GHz,12MB L2/1333MHz) x2
メモリ	32GB DDR2 FBD RAM(667MHz, ECC, Register, 2GBx16)
OS	Ubuntu11.04 64bit
コンパイラ	gcc

A. 構築時間の測定結果

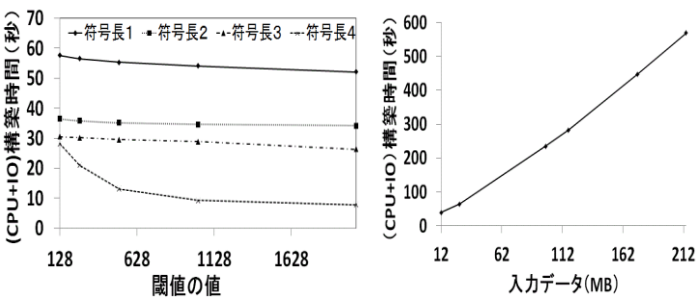
本節では，接頭辞の長さ l と閾値 τ の値により構築時間に与える影響について考察する．

・実験手法：l = 1, 2, 3, 4 を選び符号化する．それぞれの長さ l に対して，閾値 τ を 128 から 2048 まで，2 倍ずつ増やして構築実験を行い，構築時間を測定する．この実験では，ヒトの DNA 配列の第一染色体の先頭から 12MB の文字列データを取り出して，利用した．図 3(a)に実験結果を示す．

・実験結果および考察：図中の縦軸は構築時間で秒の単位で，横軸は閾値 τ の値である．どの接尾辞の長さ l でも，閾値 τ の値が大きくなると，構築時間が短縮された．その理由として，閾値の値が大きいほど，サフィックス木が浅くなり，ノードの数が少なくなることが挙げられる．また，長さ l = 4，閾値 τ = 512 のところで，構築時間が大きく変化した．その理由としては，接頭辞の長さ l が大きいほど列挙するデータが少なくなるため，構築スピードが速くなると考えられる．次節では，この実験結果を踏まえ，接尾辞の長さ l = 4，閾値 τ = 1024 を選んで，入力データと構築時間との関係について実験を行う．

B. 入力データと構築時間との関係

本実験では，ヒトの第一染色体の先頭から 12MB，27MB，98MB，117MB，174MB，214MB の DNA 配列データを取り出し，それぞれの DNA 配列データに対するサフィックス木の構築時間を測定する．なお，実験では，接尾辞の長さ l = 4 で符号化し，閾値 τ = 1024 を利用する．この実験結果を図 3(b) に示す．



(a)　(b)

図 3. 構築性能

(a)は 閾値，プレフィックス長さ，構築時間の関係であり，(b)は入力データと構築時間との関係である．

この結果を見ると，入力データのサイズに対して，構築時間はほぼ線形と見られる．214MB の入力データに対する構築時間は約 550 秒（9 分程度）になった．この実験により，DynaCluster アルゴリズムを基本にすることにより，3GB のデータ構築は，2 時間程度で達成できると予想できる．従って，大規模な文字列データに対するサフィックス木の構築が可能であることがわかった．

また，入力データ 214MB のところで閾値を 64 から 1024 まで変えながら, ディスク上で得られたサフィクス木のサイズを測定した. この測定結果を図 4 に示す.

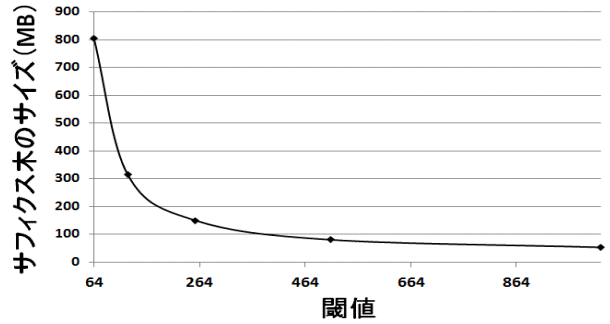


図 4. 閾値とディスク上に構築された サフィクス木のサイズとの関係

縦軸はサフィクス木のサイズの MB 単位である. 横軸は閾値の値である. 閾値が大きければ大きいほどサフィクス木のサイズは小さくなると見られる.

C. 検索キーとスピードとの関係

検索処理の実験では，サフィックス木を使わない場合とディスク版サフィックス木（SPM, CPM, DPM）を使う場合の検索性能上の違いを見るために，両者の検索時間を測定する．

サフィックス木を使わない場合は，入力データ 214MB を符号化したファイルに対して検索を行う．ディスク版サフィックス木を利用する場合は，214MB の入力データからディスク版サフィックス木を構築し，検索時間を測定する．

なお，データベースの検索キーとして利用する文字列は，同じ文字 A だけからなる文字列 <AAAAAAA……>の形式をしており，文字列の長さは，4 文字, 16 文字, 32 文字の 3 種類とする.

表２.検索キーの長さと検索時間との関係

	4	16	32
ST 有り	0.000017	0.000047	0.000081
ST 無し	2.680881	2.715286	2.72064
出現回数	3372581	81450	2312

表 2 の「ST 有り」はディスク版サフィックス木を利用する場合の検索をさす．また，「ST 無し」はサフィックス木を使わない場合の検索をさす．実験結果より，サフィクス木を用いた検索では，検索速度が検索キーと比例しており，サフィックス木を使わない場合よりも遥かに高速な検索が達成されていることを確認できた.

V. おわりに

本研究では，大規模な文字列データベースの高速検索を達成するために， DynaCluster アルゴリズムに基づくサフィクス木構築システムを実装した．このシステムでは，大規模文字列データに対する大規模な索引構造を構築可能にするために，(1) 位置情報の一時記憶については，ルートノードおよび深さ 1 のノードで実施せずに深さ 2 以降のノードに対して実施，(2) 各ノードで保管するデータについては，部分文字列の入力データ上での出現回数を保存， (3) ノードデータのディスクアクセスについては，OS が提供する仮想ページの使用を止め，実メモリとディスクとの間のデータページのやりとりを可能とするバッファ管理モジュールを実装・利用した. 評価実験の結果，214MB の文字列データに対して，ディスク上のサフィクス木を構築することができた.

今後の課題としては，100GB 規模の DNA 配列データに対するディスク上のサフィクス木構築がある．また, DynaCluster アルゴリズムの特徴の一つとして，隣接ノード作成されたときバッファ上で一つのページに格納するという動的クラスタリングの技術がある．しかし，図 1(b) からも予想されることだが，検索キーの与え方によっては，必ずしも，その特徴がうまく利用されない場合がある．従って，どのような検索キーが与えられても，平均的に良好なページアクセスを可能にする方法の検討が挙げられる.

謝辞

本研究の一部は，日本学術振興会・科学研究費補助金(基盤研究(C)，課題番号：20500137) ，文部科学省・科学研究費補助金（若手研究(B), 課題番号：23700124）の支援により行われた.

参考文献

[1] Ching-Fung Cheung, Jeffrey Xu Yu, and Hongjun Lu: Constructing Suffix Tree for Gigabyte Sequences with Megabyte Memory, *IEEE transactions on knowledge and data engineering*, Vol. 17, No. 1, January 2005.

[2] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, Meichun Hsu: Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach, *IEEE Transactions on Knowledge and Data Engineering*,Vol.16, No.11, pp.1424-1440, 2004.

問い合わせ先
〒731-3194
広島市安佐南区大塚東 3 丁目 4 番 1 号
広島市立大学 情報科学研究科 知能工学専攻
グエン　フーン　バック(Nguyen Phuong Bac)