7L-08

文字列探索アルゴリズムを応用したダブル配列構築の高速化

仲村 勇馬 †　　山本 幹雄 ‡
筑波大学 情報学群 情報メディア創成学類 †　　筑波大学 システム情報系 ‡

1 はじめに

トライ木 (Fredkin et al., 1960) の実装手法の一つにダブル配列 (Aoe, 1989) があり, これはトライ木の高い検索性能を維持しつつコンパクトに格納できるデータ構造である. しかし, 大規模なダブル配列の構築には非常に時間がかかる (Norimatsu et al., 2016). これは構築中のダブル配列にトライ木のノードの子ノード列を格納する処理において, 子ノード列の格納位置を探索する処理の最悪計算量が, トライのノード数に対して 2 乗のオーダーとなっているためである.

本稿では, 子ノード列の格納位置を探索する処理として文字列探索アルゴリズムの 1 つである KMP 法 (Knuth et al., 1977) の考え方を応用した手法を用いることでダブル配列の構築が高速化されることを示す.

2 ダブル配列

ダブル配列 (Aoe, 1989) はトライ木のノード間の遷移を 2 本の密な 1 次元配列で表現する効率の良いデータ構造である.

トライ木からダブル配列を構築する際の概念的な処理の流れを図 1 に示す. まずトライ木を遷移表と呼ばれる 2 次元配列として表現する. 遷移表は 2 次元配列の各行をトライ木のノードに対応させ, 配列内に遷移先のノード番号を格納したものである. 次に遷移表の各行を要素が格納されている部分が衝突しないようにずらし next 配列と呼ばれる 1 本の配列にまとめる. この時, 正しく遷移を行うために各行のずらし幅を offset 配列に, 遷移の成否を確認するために親ノード (遷移元) のノード番号を check 配列に格納する. そして, next 配列に格納されているノード番号を offset 配列に格納されているそのノードのずらし幅に書き換えた配列を base 配列とする. 更に, check 配列の値を遷移元のノード番号から next 配列にそのノード番号が格納されている index の値に書き換える. このようにしてトライ木のノードの遷移情報を base と check の 2 本の 1 次元配列に格納できる. ダブル配列の構築において最も時間がかかる処理はトライ木のノードの子ノード列を 1 次元配列に格納するための格納位置を探索する処理である. 様々な候補位置で配列内の要素と子ノード列の要素が衝突し続け探索の試行回数が増える場合, この処理に時間がかかることになる. そこで本研究では格納する子ノード列をあらかじめ走査することで配列の要素と子ノード列の要素が衝突するような候補位置での探索の回数を削減する手法を検討する.

3 提案手法

テキスト中の検索文字列の位置を求める文字列探索アルゴリズムの 1 つである KMP 法 (Knuth et al., 1977) の考え方を利用する. KMP 法は検索文字列を事前に走査し, 検索文字列中の文字とテキストの不一致が発生した際の検索文字列の移動量を求め, その移動量に従って検索文字列の位置を変えながら文字列探索を行うアルゴリズムである.

このアルゴリズムをダブル配列構築に応用する際の違いは, (1) 文字列探索では多数ある文字種がダブル配列構築では配列が埋まっているか空いているかの 2 種類に減ること, (2)KMP 法は文字列が完全に一致する位置を見つけるアルゴリズムだがダブル配列構築は格納する子ノード列の子ノードの位置のみ配列が空であればよく, 子ノード列の中の子ノードが存在しない位置はワイルドカードとなることが挙げられる.

提案手法では子ノード列の格納位置を求める前に子ノード列に対して走査を行い, 子ノードそれぞれに対し

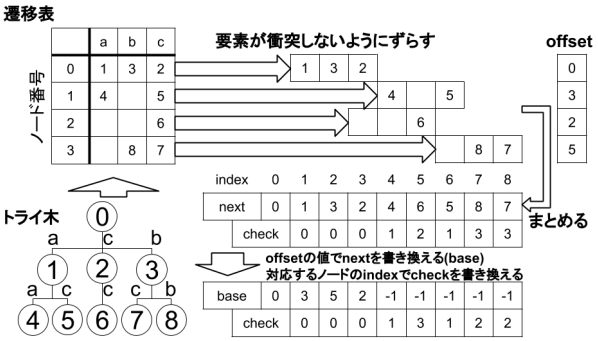

図 1　ダブル配列の構築

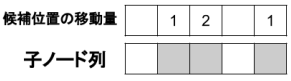

図 2　移動量の例

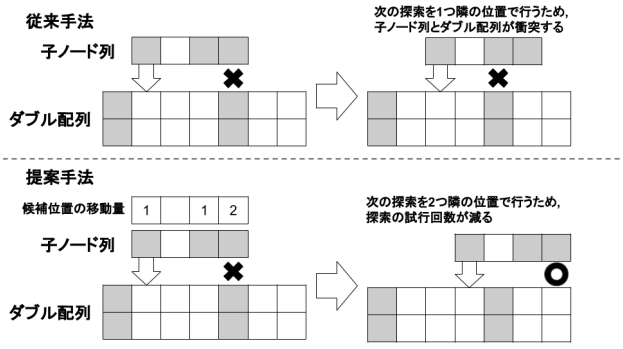

図 3　探索処理の実行例

Speeding up double array construction using string search algorithms
† Yuma NAKAMURA, College of Media Arts, Science and Technology, School of Informatics, University of Tsukuba
‡ Mikio YAMAMOTO, Faculty of Engineering, Information and Systems, University of Tsukuba

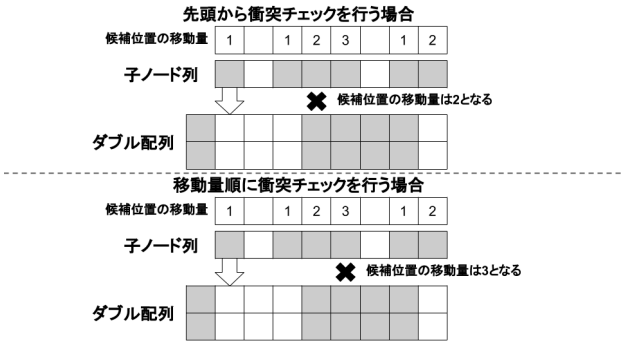

図 4　衝突チェックの順序

てその位置で衝突が起きた際の候補位置の移動量を求める．そして子ノード列の格納位置を求める際にその移動量に従って格納位置の候補を更新する．候補位置の移動量は，通常の移動量である 1 にその子ノードより前方にあり連続している子ノードの数を加えた数となる．

候補位置の移動量の例を図 2 に示す．1 つめの子ノードでは前方方向に子ノードが存在しないため移動量は 1 となる．2 つめの子ノードでは前方方向に 1 つの子ノードが連続しているため，移動量は 1 増えて 2 となる．3 つめの子ノードは前方に 2 つの子ノードがあるが，前方の子ノードと 3 つめの子ノードの間には空白があり連続していないため移動量は 1 となる．

子ノード列格納位置の探索処理の実行例を図 3 に示す．子ノード列の 3 番目の子ノードの位置で衝突が起きた場合，従来手法では子ノード列の候補位置を 1 つずらして次の探索を行うため，前回の探索で 3 番目の子ノードが衝突した位置で 2 番目の子ノードが衝突してしまう．しかし，提案手法では事前の走査によって，3 番目の子ノードに対応する候補位置の移動量が 1 の場合は次の探索で配列と子ノード列の衝突が必ず起きることがわかるため，候補位置の移動量は 2 となり，不要な探索の回数を削減し子ノード列の格納位置の探索を高速化できる．

また，図 4 に示すように子ノード列の衝突チェックを子ノード列の先頭からではなく，衝突が起きた際の候補位置の移動量について降順に行うことで常に最大の候補位置の移動量が得られる．しかし，移動量順に衝突チェックを行うためには子ノードを移動量順にソートする必要がある．

4　実験

提案手法の有効性をダブル配列を用いた言語モデルである DALM (Double Array Language Model) (Norimatsu et al., 2016) の構築実験によって示す．

4.1　実験の条件

構築実験では従来手法，提案手法 (前方からチェック)，提案手法 (移動量順にチェック) の 3 つの手法を用いて DALM を構築し，その構築の中でダブル配列の構築にかかった時間を計測し，評価を行う．入力データとなるトライ木にはネット上のニュースサイトの記事データから作成したエントリ数 (ノード数) の概数が 5000 万，1 億，2 億，5 億のものを用いた．実験に使用した計算機の性能は CPU が Xeon E5-2620 v4 2.10GHz，コア数は 16，メモリが 256GB である．DALM 構築の並列化には 16 並列の細粒度並列化 (石井他 2018) を用い，配列の分割は行わなかった．

4.2　実験結果と考察

構築実験の結果を図 5 に，従来手法による構築時間を 1 とした場合の提案手法の構築時間を図 6 に示す．提案手法はどちらも従来手法と比較して短い時間でダブル配列の構築ができている．特にトライ木のノード数が 5 億の場合での提案手法 (移動量順にチェック) での構築時間は 77,530 秒であり従来手法での構築時間の 130,551 秒の約 60% の時間でダブル配列が構築されている．提案手法の中では移動量順にチェックする手法は前方からチェックする手法よりも高速にダブル配列を構築できており，これから常に最大の移動量を得られることによる処理速度の減少量が，子ノードのソート処理による処理速度の増加量を上回っていることがわかる．

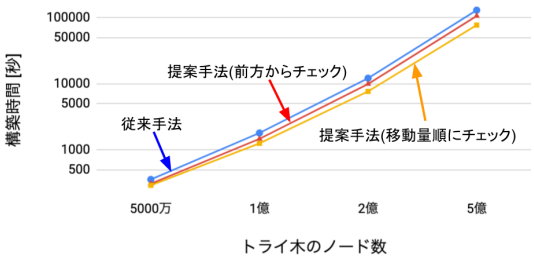

図 5　構築時間

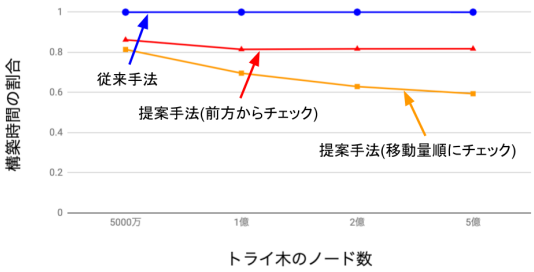

図 6　従来手法と比較した提案手法の構築時間の割合

5　おわりに

本稿では，ダブル配列の構築を高速化するために，KMP 法を応用して子ノード列の格納位置探索を高速化する手法を提案しその有効性を示した．今後は他の文字列探索アルゴリズムのダブル配列構築への応用について検討していく．

参考文献

Aoe, Jun-ichi (1989). “An efficient digital search algorithm by using a double-array structure”, IEEE Transactions on Software Engineering, Vol. 15, No. 9, pp. 1066–1077.

Fredkin, Edward, (1960). “Trie Memory”, Communications of the ACM, Vol. 3, No. 9, pp. 490–499.

Knuth, Donald E., James H. Morris, Jr., and Vaughan R. Pratt(1977). ”Fast pattern matching in strings” SIAM Journal on Computing, 6, 1, pp. 323 – 350.

Norimatsu, J., M. Yasuhara, T. Tanaka, and M. Yamamoto (2016). “A fast and compact language model implementation using double-array structures”, ACM TALLIP Vol. 15, No. 4, 27 pages.

石井瑛彦，芳賀駿平，竹中孝介，大隅賢二，山本幹雄 (2018) ”細粒度並列処理によるダブル配列言語モデルの構築高速化”，言語処理学会 第 24 会年次大会発表論文集, pp. 216 - 219.