様　式　Ｃ－１９、Ｆ－１９、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成 ２７ 年　 ５ 月 ２７ 日現在

機関番号：１６１０１
研究種目：基盤研究(C)
研究期間：2012～2014
課題番号：２４５００１１８

研究課題名（和文）最適状態探索とＣＨＥＣＫ領域の削除によるダブル配列の辞書圧縮手法に関する研究

研究課題名（英文）Methods to Retrieve Optimal States and to Compress Dictionaries for Double Array Structures by Deleting CHECK

研究代表者
泓田　正雄（Fuketa, Masao）

徳島大学・ソシオテクノサイエンス研究部・准教授

研究者番号：１０３０４５５２

交付決定額（研究期間全体）：（直接経費）　　3,200,000円

研究成果の概要（和文）：コンピュータで，検索は様々なアプリケーションで使用される非常に重要な処理である．ダブル配列は，単語辞書などを高速に検索する手法であり，名前の通り二つの配列(BASEとCHECK)を使用するが，このCHECK配列を削除することにより，高速性を維持したまま，記憶容量を小さくする手法を提案した．実験の結果，限定されたキーセットにおいて，有効に活用できる手法を提案することができた．

研究成果の概要（英文）：Retrieval is very important processing for various applications in computers. Double Array is the method to retrieve words at fast speed. Although it uses two arrays called BASE and CHECK, the proposed method is to compress space usage and keep the retrieval speed of Double Array by deleting CHECK. From experimental results, the proposed method is efficient in some keysets.

研究分野：情報検索

キーワード：トライ　ダブル配列

１．研究開始当初の背景

近年，計算機を始めとする通信ハードウエアの向上，またインターネットに代表されるネットワーク網の発達などにより，大量のデータを扱う必要性が急速に高まっている．また，携帯電話やスマートフォンなどの小型端末において大量のデータを少ない領域中に格納しなければならない状況も多くなっている．このような環境において，データを扱う最も基本的な操作である「検索」には，高速化や省メモリ化が必須の条件となっている．検索のためのデータ構造に，木構造の頂点を遷移することにより検索を行うトライ（オートマトンの一種）がある．トライは共通接頭辞を併合しているので，入力語と完全に一致する単語を検索できるだけでなく，入力語から始まる単語や，前方部分一致検索を行うことができ，自然言語処理などにおける各種の辞書（データベース）として利用されている．トライを実現するためのデータ構造として，マトリックス表現，リンクリスト，LOUDS などの手法があるが，サイズが大きい，検索速度が遅いなどの問題がある．BASE,CHECK と呼ばれる配列を用いて，このトライを表現するデータ構造であるダブル配列法は O(1)で検索を行うことができる優れた検索手法である．しかし，このダブル配列は，辞書のサイズが大きいという問題点がある．

２．研究の目的

ダブル配列は,現在の状態番号を s, 遷移文字を c,遷移先の状態番号を t としたとき,

t=BASE[s]+CODE[c]　　(1)
CHECK[t]=s　　(2)

の２式を満たすように BASE と CHECK の 配列の値を決定する．CHECK 配列に遷移文字である c を格納して辞書サイズを小さくする手法も提案されている．

(1)最適状態探索

ダブル配列は，式(1)(2)を満たせば構築できるので，構築結果は一つではなく，BASE,CODE の値により，何通りものダブル配列を作成することができる．そこで，あらゆる CODE 値の組み合わせを考え，ダブル配列のサイズが最小となる CODE 値を求める．ダブル配列の２式は変わらないので，ダブル配列の高速性を維持したままサイズを小さくできる．

(2)CHECK 配列の削除

ダブル配列の持つ高速性を維持したまま，CHECK配列に格納される情報をBASE配列に保持させることにより，CHECK 配列を削除して，ダブル配列のサイズを小さくする手法について研究を行う．この手法においても検索は２式で行われるので，高速生は失われない

３．研究の方法

(1) 最適状態探索

従来の BASE を削除してダブル配列を構築する SAMC 手法に対し，列挙アルゴリズムを用いてすべての組み合わせの CODE 値を求め，求めた CODE 値を用いてダブル配列を構築する処理を，複数台のコンピュータによる並列処理を用いて，高速に最適解を求める．並列処理には，Hadoop，OpenMPI を用いる．

(2)CHECK 配列の削除

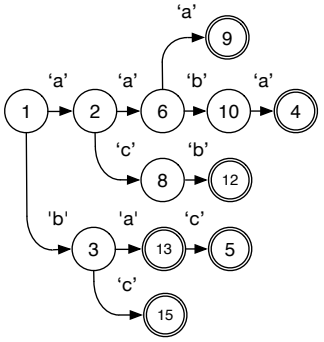
BASE に CHECK に格納されていた遷移文字の情報を保持させるため，状態 t へ文字 c で遷移した時,

BASE[t] % 全文字数 = CODE[c]　　(3)

を満たすように BASE 値を決定する．登録するキー集合に出現する文字をσ，その数を｜σ｜としたとき，CODE[c](c∈σ)のとり得る範囲を 0～｜σ｜-1 とし，各文字 c における CODE の値を全てユニークな値とすれば，式(3)を満たすように BASE 値を決定することが可能となる．トライの葉ノードの場合，BASE 値を負の値にするので，関数 ABS を絶対値を求める関数とした時，ダブル配列の２式を

t=BASE[s]+CODE[c]　　(4)
ABS(BASE[t]) % ｜σ｜ = CODE[c]　　(5)

とすることができる． この提案手法は，O(1)で遷移できるので，ダブル配列の検索の高速性を維持したまま，CHECK 配列を削除でき，ダブル配列のサイズを小さくすることが可能となる．



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BASE	2	6	13	-3	-2	9		11	-3	4		-1	3		-2
CHECK		a	b	a	c	a		c	a	b		b	a		c
LEAF				1	1				1			1	1		1

	a	b	c
CODE	0	1	2

図１．提案手法のダブル配列

図１に提案手法のキー集合{“aaa“, “aaba“, “acb“, “bac“, “bc“}におけるダブル配列を示す．出現する文字は‘a‘, ‘b‘, ‘c‘の３種類なので，｜σ｜=3 となる．図１では CHECK 配列を示しているが，ABS(BASE[t]) % ｜σ｜ = CODE[CHECK[t]]となっているので，実際は CHECK 配列を削除できる．また二重丸は終端ノードを表し，配列 LEAF で示している．LEAF は各要素に 1 ビットしか使用しないので，BASE と LEAF を合わせて 32 ビットの整数値で

表現可能であるため，ダブル配列のサイズは4×全ノード数（バイト）で表すことが可能となる．
状態3から状態13への文字‘a‘による遷移を例として説明する．まず，式(4)より，t=BASE[3]+CODE[‘a‘] = 13+0=13 となり，次に式(5)より ABS(BASE[t]) % | σ | = ABS(BASE[13]) % 3 = 3%3 = 0 となる．この値は,遷移文字によるCODE値であるCODE[‘a‘]と一致するので，遷移を確認できた．
ダブル配列の構築は，状態sから遷移している文字集合 S について，式(1)による遷移先である状態tが，ダブル配列上で未使用である最小の値を BASE[s]に設定する．この値を求める関数を X_CHECK(S)としたとき，通常のダブル配列では x=0 から順番に S に対して式(1)を計算し，すべて未使用状態となる x を返す．ダブル配列では，BASE値にどのような値でも設定できたが，提案手法では，|σ|で割った余りが状態sに遷移している文字cにおける CODE[c]と一致しなければならない．そのため,X_CHECK で探索の開始は x=CODE[c]から始め，あとは x=x+|σ|として，式(4)における t が未使用である x を求めればよい．この処理により，X_CHECK が返す値を|σ|で割った余りは必ずCODE[c]となる．

４．研究成果

(1) 最適状態探索
ダブル配列を深さ毎に分割するダブル配列の一手法である SAMC 法において，深さ毎にCODE値を求める手法について実験した．並列処理には Hadoop と OpenMPI を用いた．この手法では，深さ毎にすべての組み合わせのCODE値について処理をしなければならず,解析に時間がかかる上，キー集合によっては，解が見つからない場合もあった．さらに，サイズに閾値を設け，構築途中に閾値を超える場合にはその処理を中断する枝刈りの処理も組み込んだが結果は変わらなかった．
(2)CHECK配列の削除
提案手法の有効性を示すため，要素数，記憶容量，検索速度について，従来法との比較を行った．使用したコンピュータは，Mac Pro-Mid 2010(CPU：2 x 2.4 GHz Quad-Core Intel Xeon, メモリ：16GB 1066 MHz DDR3)である．キー集合としては，

- 郵便番号：10万語
- EDR英単語辞書：10万語
- DNA（接尾辞）：1000語
-

を用いた．キー集合の情報と，実験結果を表１に示す．従来法は，CHECK に文字を格納した通常のダブル配列である．検索速度は，すべてのキーを検索した時間を計測した．
検索速度の理論的評価は二手法とも１回の繊維に対して O(1)であるが，従来法の式(1)(2)に比べ，提案手法では式(4)(5)となるので，一回の遷移における演算回数が多くなってしまっている．そのため，提案手法の法が遅いという結果となった．しかし，LOUDSなど，他のトライを実現するデータ構造に比べると高速に検索することができる．

表１．CHECK削除における実験結果

	郵便	EDR	DNA
ノード数	127,973	532,862	491,371
キー情報			
平均長	7	11.4	500.5
最大長	7	71	1000
文字種	10	41	4
要素数			
提案手法	147,456	1,819,904	921,088
従来法	131,840	535,296	526,592
記憶容量 (byte)			
提案手法	589,824	7,279,616	3,684,352
従来法	659,200	2,676,480	2,632,960
検索速度 (micro sec)			
提案手法	5,103	25,133	2,978
従来法	3,214	7,207	1,579

郵便番号においては，記憶容量は従来法よりも小さくなった．しかし，他のキー集合では大きくなってしまった．これは，要素数が従来法よりも多くなってしまっているからである．従来手法は自由に BASE の値を決められるので,BASE,CHECK の前から順に空き要素を埋めることができた．しかし，提案手法では式(5)を満たすようにBASE値を決めなければならず，空き要素が多くなってしまい，結果的にサイズが大きくなってしまったと考えられる．この傾向は，文字種が多くなるほど顕著になる．文字種が少ないDNAでもサイズが大きくなったのは，遷移文字の出現数に偏りがあったためだと考えられる．

５．主な発表論文等

（研究代表者、研究分担者及び連携研究者には下線）

〔学会発表〕（計 ２ 件）

① Masao Fuketa, Kazuhiro Morita and Jun-ichi Aoe, Comparisons of Efficient Implementations for DAWG, 7th International Conference on Computer Science and Information Technology (ICCSIT 2014), 2014年12月23日, Barcelona(Spain).

② Masao Fuketa, Kazuhiro Morita and Jun-ichi Aoe, A Retrieval Method for Double Array Structures by using Byte N-Gram, 6th International Conference on Computer Science and Information Technology (ICCSIT 2013), 2013年12月21日, Paris(France).

６．研究組織

(1)研究代表者

泓田 正雄(FUKETA MASAO)
徳島大学・大学院ソシオテクノサイエンス研究部・准教授
研究者番号：10304552