パトリシアトライ構造の効率的な圧縮アルゴリズム

2H－1

獅々堀 正幹　住友 徹　岡田 真　青江 順一

徳島大学工学部知能情報工学科

1．はじめに

近年，文書ソフトウェアの普及に伴い，全文検索手法に関する研究が盛んに行われている．キーワード等の指定語以外の文字列を検索する手法としては，文書内の半無限部分文字列（sistring）をパトリシアトライ構造に登録する手法[1]が有名であるが，索引部となるパトリシア構造を如何にコンパクトに圧縮するかが重要な問題となる．Jonge ら[2]は一般的な２進木トライ構造を圧縮する手法を提案したが，パトリシアトライ構造にそのまま適用することはできない．本稿では，Jonge らの手法を拡張し，パトリシアトライ構造を圧縮する手法を提案する．

2．２進木トライ構造の圧縮法

Jonge ら[2]は，通常の２進木トライに対して，１本の枝しか持たないノードにダミーリーフと呼ばれる擬似的な葉を付加し，先行順ビット列と呼ばれるコンパクトなビット列に変換する手法を提案した．この先行順ビット列は，木構造の状態を示す treemap，葉の状態（ダミーか非ダミーか）を示す leafmap，各キーが蓄えられているバケットへのアドレスを格納する B_TBL から成る．treemap は木を先行順に走査し，内部ノードを辿った際には'0'を，葉の場合には'1'を出力することにより作成する．leafmap も同様に先行順走査し，辿った葉がダミーリーフならば'0'を，非ダミーの場合には'1'を出力する．B_TBL は，先行順に辿った葉に対応したバケットのアドレスを表形式で格納する．尚，ダミーリーフを用いて，全ノードが２本の枝を保持することにより，検索等の処理において，右部分木に進む場合には，treemap 上で'1'のビット数が'0'のビット数より１つ多くなるまで，ビット位置を進めればよくなる．

例えば，キー文字列を構成する各文字の内部コード値の２進数表現（a,b,…の内部コード値を 0,1,…）を登録キーとした場合，キー集合 K＝｛air,art,bag,bus,tea,try,zoo｝に対する２進数表現は表１で与えられ，このキー集合から生成される２進木トライを図１に示す．尚，各葉に対応したバケットに格納可能なキー数（B_SIZE）は２とする．また，図１の２進木トライを圧縮した先行順ビット列を図２に示す．

表１　キー集合Ｋに対する２進数表現

Keys	Internal codes	Binary sequences
air	0 / 8 / 17	00000 01000 10001
art	0 / 17 / 19	00000 10001 10011
bag	1 / 0 / 6	00001 00000 00110
bus	1 / 20 / 18	00001 10100 10010
tea	19 / 4 / 0	10011 00100 00000
try	19 / 17 / 24	10011 10001 11000
zoo	25 / 14 / 14	11001 01110 01110

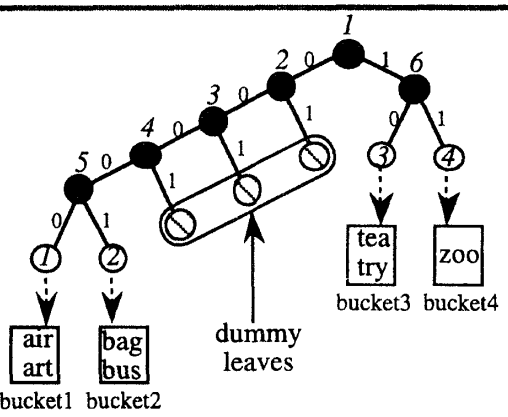

図１　キー集合Ｋに対する２進木トライ

treemap : 0 0 0 0 0 1 1 1 1 1 0 1 1

leafmap : 1 1 0 0 0 1 1

B_TBL :

1	address of bucket 1
2	address of bucket 2
3	address of bucket 3
4	address of bucket 4

図２　図１の２進木トライに対する先行順ビット列

An Efficient Compression Algorithm of the Patricia Trie

Masami Shishibori, Toru Sumitomo, Makoto Okada and Jun-ichi Aoe

Dept. of Information Sciences & Intelligent Systems, Faculty of Engineering, Tokushima University

3．パトリシアトライ構造の圧縮法

パトリシアトライは，通常の２進木トライから１本の枝しか持たないノードを全て削除した木構造であるため，ダミーリーフを用いなくても各ノードが常に２本の枝を持つことが保証されている．しかしながら，削除されたノードに関する情報を保持できるように先行順ビット列を改良する必要がある．

本手法では，leafmapの代わりにnodemapと呼ばれるビット列を生成する．nodemapは各内部ノードが削除ノードを格納しているか否かを表す．nodemapの作成方法は，木を先行順に走査し，辿った内部ノードが削除ノードを保持していれば，その削除ノード数と同じ数だけビット値'1'を出力した後，その内部ノード自身を表す１個のビット値'0'を出力する．また，内部ノードが削除ノードを保持していなければ，ビット値'0'を１回だけ出力する．このような方法で生成されたnodemapの長さは，通常の２進木トライ内に含まれる内部ノードの総数と等しくなる．キー集合Kに対するパトリシアトライを図３に，それに対する先行順ビット列を図４に示す．

ここで，treemapの長さは，図１の２進木トライに含まれるダミーリーフ数の２倍だけ短くなっている．即ち，２進木トライ内に含まれるダミーリーフ数が多ければ多いほど，より短いtreemapがパトリシアトライに対して生成される．

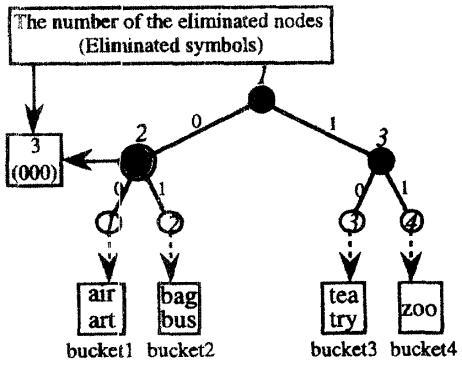

図３　キー集合Kに対するパトリシアトライ

treemap :　0 0 1 1 0 1 1
nodemap :　0 1 1 1 0 0

B_TBL :

1	address of bucket 1
2	address of bucket 2
3	address of bucket 3
4	address of bucket 4

図４　パトリシアトライに対する先行順ビット列

表２　実験結果

key sets	Japanese nouns		English nouns	
kinds of tree	**Ordinary**	**Patricia**	**Ordinary**	**Patricia**
nodes	19,301	12,003	29,139	12,317
external nodes	9,651	6,002	14,570	6,159
dummy leaves	3,649	0	8,411	0
dummy rate (%)	**37.8**	**0**	**57.7**	**0**
size of (Kbyte)				
treemap	2.41	1.50	3.64	1.54
leafmap, nodemap	1.21	1.21	1.82	1.82
B_TBL	12.00	12.00	12.32	12.32
decrease rate (%)	**37.8**		**57.7**	

また，ビット列上での検索処理では，内部ノードが削除ノードを格納しているか否かを検証するため，nodemap内のビット値'1'に出会した場合は，ビット値が'0'になるまで，各ビット位置をスキップする．

4．評価

従来の先行順ビット列と本手法による先行順ビット列との比較を日本語名詞，英単語，各５万語に対して行った実験結果を表２に示す(B_SIZE=16)．

本手法を用いることにより，treemapが40～60%に減少した．また，この減少率はダミー率と同値であり，treemapはダミーリーフの比率だけ短くなる．一般的に，２進木トライのダミー率は40～60%と報告されており[3]，本手法は如何なるキー集合に対しても，treemapを40～60%短縮可能であると言える．更に，ビット列を主記憶上に格納することで１回のディスクアクセスで検索できるため，PAT array[1]を用いた場合よりも高速な検索が可能となる．

5．まとめ

本稿では，パトリシアトライをコンパクトなデータ構造に圧縮する手法を提案した．今後の課題としては，日本語文書に適したsistringの切り出し法を考案し，本手法を全文検索に応用する計画である．

参　考　文　献

[1] G. H. Gonnet, "Handbook of Algorithms and Data Structures," Addison-Wesley, Reading Mass., Ch. 3 (Searching Algorithms) pp.25-147 (1984).

[2] W. D. Jonge, A. S. Tanenbaum and R. P. Riet, "Two access methods using compact binary trees," IEEE Trans. Softw. Eng., Vol.SE-13, No.7, pp.799-810 (1987).

[3] 獅々堀正幹，清原聡，青江順一，“階層化による２進ディジタル探索(BDS)木の改善”，電子情報通信学会論文誌，Vol.J79-D-I, No.2，pp.79-87 (1996).