Vol. 37 No. 8 情報処理学会論文誌 Aug. 1996

テクニカルノート

可変長節パトリシアトライ索引構造

程 亜非[†] 槍垣泰彦[††] 池田宏明[†]

圧縮パトリシアトライ索引構造を提案する．従来のパトリシア構造に対して，前方圧縮を導入し，新たなデータの蓄積構造とそれに対応する検索アルゴリズムを示す．本手法は従来のパトリシアトライ構造と同等な検索手数であるにもかかわらず，記憶量を節約できるという特長を有する．

An Index Structure of the PATRICIA Trie with Variable Length Nodes

YAFEI CHENG,[†] YASUHIKO HIGAKI[††] and HIROAKI IKEDA[†]

A new index structure of the PATRICIA trie based on a front compression of keys is proposed. Searching algorithm for the proposed index structure is described. The most important feature of the proposed method is that required memory is significantly lower than the traditional PATRICIA trie, whereas the retrieval time is almost the same.

1. はじめに

情報検索と自然言語処理の分野ではトライ索引構造[1]がよく使われている．本論文は種々のトライ索引構造[2),3)]のひとつであるパトリシアトライ[4),5)]に基づいて，これに前方圧縮[6)]を適用した可変長節パトリシアトライを新たに提案し，その性能評価を行う．それによって，従来のパトリシアトライと比較して，提案した可変長節パトリシアトライでは記憶容量を減少でき，しかも，検索速度はほぼ同一である[7)]．

2. パトリシアトライ構造

パトリシアトライ（以下Pトライと略する）の構造と検索方法を例示で説明する．キー集合 $\mathcal{W}$ ={A, S, C, H, I} の各キーに対応するビットパターンを {00001, 10011, 00011, 01000, 01001} とする場合のPトライの例を**図1**に示す．Pトライは2分木構造を持っている．ルート節と各中間節では左右へのポインタおよび検索時の検索キーの対象ビット位置（図1の節の中の数字，以下添字と呼ぶ）を有する．Pトライのルート節からの任意の1つの経路上の各節の添字を $I_0, I_1, \cdots, I_{leaf}$ とすれば，$I_0 < I_1 < \cdots < I_{leaf}$ の関係がある．検索時にルート節から節の添字で示している検索キーのビット位置にあるビットの値により，0ならば左へ，1ならば右へ進むことにより次の節へ移る．このようにして葉までたどり，検索キーと葉に置いてあるキーを比較して検索が終了する．キーの数が N の場合，Pトライの検索手数は $O(\log_2 N)$ で，総節数は $N-1$ である．

3. パトリシアトライの改良

3.1 従来のパトリシアトライの問題点

Pトライではキーの頭から共通の文字列部分（接頭辞と呼ぶ）を持つキーどうしに対して，典型的なトライ[1)]のように接頭辞経路上にその共通文字列を置くことができるか，あるいはそれに等価な方法があるかという問題が未検討であると考えられる．この問題に対して，筆者らはキーをPトライのすべての中間節にも置くことにして，空間性能を向上させるために，各経路上のキーの接頭辞に前方圧縮を適用する．これによって，上記問題を等価的に解決する．これを可変長節パトリシアトライ索引構造（以下VPトライ）と呼ぶ．

3.2 前方圧縮法

前方圧縮はソートされた順次ファイルに適用可能な圧縮法である[6)]．ソートされた順次ファイルの隣接し

† 千葉大学大学院自然科学研究科
Graduate School of Science and Technology, Chiba University, Japan
†† 千葉大学工学部電気電子工学科
Department of Electrical and Electronics Engineering, Chiba University, Japan

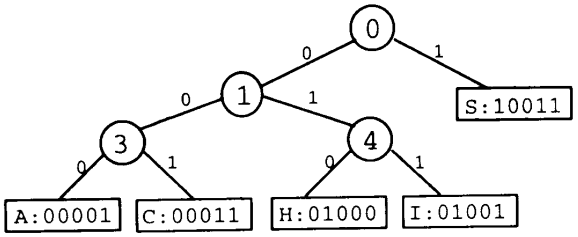


図1 従来のパトリシアトライ
Fig. 1 An example of PATRICIA trie.

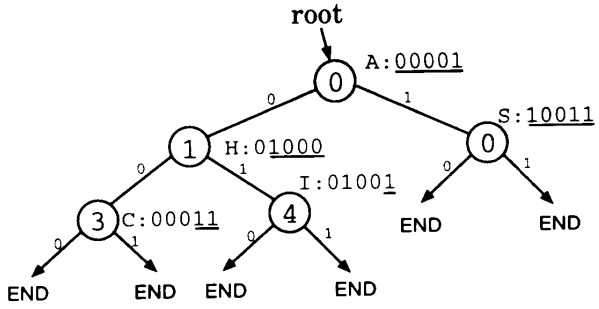


図2 VP トライの例
Fig. 2 An example of VP trie.

た第 $k-1$ 番目のレコードと第 k 番目レコードを次のように表現する.

$k-1$: $X_p = a_1 a_2 \cdots a_c \cdots a_p$

k: $Y_q = b_1 b_2 \cdots b_c \cdots b_q$

ここで, a_c と b_c は文字データであるとする. もし $X_c = Y_c$ であれば, 文字数情報 c (これを圧縮情報と呼ぶ) で置き換えることによって Y_q を $Y'_q = <c>b_{c+1} \cdots b_q$ と圧縮できる.

3.3 VP トライの定義

キー K の文字列のビットパターンを $x_0 x_1 \cdots x_k \wedge$ とする. 符号 '∧' はキーの文字コードに使われないコードのビットパターン (たとえば ASCII のコントロールコード) であり, キーの中止符号とする. 以下 '∧' を含めた文字列のビットパターンを $x_0 x_1 \cdots x_n$ で表す.

提案する VP トライ構造を次のように定義する.

[定義 1] VP トライは 2 分木構造を持ち, 節は (1) 添字 I, (2) 左ポインタ LP, (3) 右ポインタ RP, (4) キー K で構成される. I, LP と RP は固定長, K は可変長のデータとする.

[定義 2] 添字 I は正の整数で, ルート節の添字は 0 であるとする. ルート節から任意の経路上の各節の添字に, $I_0 \le I_1 \le \cdots \le I_{leaf}$ の関係を持つ.

[定義 3] 各節に置くキー K はその節の添字 I で示している位置から K の尾部分のビットパターン, すなわち, $K = x_I x_{I+1} \cdots x_n$ だけである.

[定義 4] ある節の LP と RP は両方とも, あるいはいずれかが指す子節がなければ, この節は葉節であり, そのポインタ値を END とする.

3.4 可変長節パトリシアトライの例

キー集合 $\mathcal{W}$ の VP トライを**図 2** に例示した. 各節の数字は添字 I である. 定義 3 により各節に置いてあるキーの内容はアンダラインで示しているキーの尾部分のビットパターンだけである. VP トライの添字 I は P トライの添字と同じ機能を持っているほかに, 前方圧縮の圧縮情報 <c> の機能も持たせる.

4. VP トライの検索アルゴリズム

VP トライではキーが中間節にも置かれているので, 葉節までたどる途中で検索キーとキーの照合を行うのが特徴である. VP トライの検索アルゴリズム **search** を**図 3** に示す. **search** では節を指すためのポインタとして, p と p_c の 2 つを使う☆. 検索キー K' と経路上のどこの節のキー K を照合するのかを p_c で示す. p は経路に沿って, 逐次的に下の葉節に移行するために使う. p の移り方は P トライと同じである. すなわち, 検索キー K' の節 p の添字で指示しているビットの値が 0 ならば, $p \leftarrow p.LP$; 1 ならば, $p \leftarrow p.RP$.

具体的な照合は変数 i, j $(i \le j)$ で表している p_c 節のキー K の部分列 ($p_c.K[i \cdots j]$ で表す) と変数 l および節 p の添字値 ($p.I$) で表している検索キー $K'[l \cdots p.I]$ $(l \le p.I)$ の部分列の間で行う. 照合失敗の場合, $K'[l \cdots p.I]$ の左から最初の失敗のビットの位置を P_{fail} $(l \le P_{fail} \le p.I)$ とする.

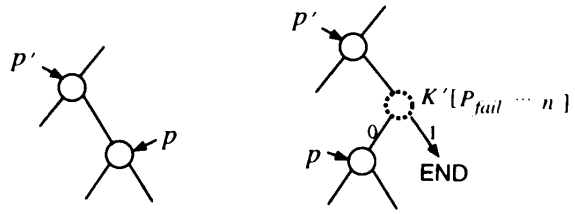
ここで, K' =H の場合について **search** による検索例を説明する. まず, $i \leftarrow 0, l \leftarrow 0$, p などのポインタがルート節 A を指すように初期化する (行 2). $j \leftarrow 0$ (行 8). 行 9 では $K'[0] \equiv p_c.K[0]$ ='0' であるので, 行 19 で両区間の開始位置 $l \leftarrow 0$, $i \leftarrow 0$ を更新してから, 行 20 により p は左の H 節へ行く. 行 8 で $j \leftarrow 1 - 0 + 0 = 1$ の更新をし, 今度は $K'[0 \cdots 1]$="01" と $p_c.K[0 \cdots 1]$ ="00" が等しくなく, K' の失敗位置は $P_{fail} \equiv p.I = 1$ であるので, 行 15 ～ 16 で p_c が H 節を指すように設定し, 両区間の開始位置 $i \leftarrow 0$☆☆, $l \leftarrow 1$ の更新をする. 行 20 で $K'[1] \equiv$ '1' であるので, p が H 節を指すように設定する. 再び行 8 で $j \leftarrow 4 - 1 + 0 = 3$. 次に $K'[1 \cdots 4] \equiv p_c.K[0 \cdots 3]$ ="1000" となり検索が成功する.

☆ または p' も用意して, それがいつも p の親節を指すようにすることもできる. VP トライの更新時に使うためである.

☆☆ 新しい照合節 (p_c) の K の最初の位置からである.

```
search(K'[0 ··· n])
1  begin
2    i, l を 0 で, p, p', p_c, next を root で初期化する;
3    if (p ≡ END) then 検索失敗;(空トライである)
4    while (next ≠ END) do
5     begin
6       p' ← p;
7       p ← next;
8       j ← p.I − l + i;
9       if (K'[l ··· p.I] ≠ p_c.K[i ··· j])
10      then
11         begin
12           if (P_fail < p.I)
13           then 検索失敗;(中間節位置)
14           else begin
15                  p_c ← p;(照合節の変更)
16                  i ← 0; l ← p.I;(区間位置の変更)
17                end;
18         end;
19      else begin l ← p.I; i ← j;(区間位置の変更) end;
20      if (K'[p.I] ≡ 0)
21      then next ← p.LP;
22      else next ← p.RP;
23    end;
24  if (K'[p.I ··· n] ≠ p_c.K[i ··· n])
25  then 検索失敗;(葉節位置)
26  else 検索成功;
27 end;
```

図3 VP トライの検索アルゴリズム
Fig. 3 Searching algorithm of VP trie.



(i) 挿入前の状態 Before insertion.
(ii) 挿入後の状態 After insertion.

図4 VP トライの節の挿入処理
Fig. 4 Insertion of a node to VP trie.

表1 VP トライの圧縮効果
Table 1 Compression ratio of VP trie.

file	size (byte)	N	S_{VP} (byte)	S_P (byte)	r
News1	2,987,411	292,930	3,526,807	5,037,928	0.700
News2	1,635,113	202,471	2,185,015	3,052,410	0.716
News3	224,031	30,701	348,207	438,938	0.793
Book1	520,335	48,605	638,130	860,570	0.742
Book2	305,308	34,278	389,976	545,254	0.715
Term1	902,408	97,956	1,147,948	1,585,384	0.724
Term2	266,014	36,506	389,878	521,556	0.748
Term3	88,876	11,226	125,785	167,458	0.751
web2	2,477,182	233,614	2,626,380	4,112,480	0.639
words	201,039	24,474	273,770	372,357	0.735

5. VP トライの挿入

新いキー K' を VP トライに挿入する際に，(1) 空トライ，(2) 葉節の END ポインタの下，(3) 2 つの節の間（中間節）の 3 種類の挿入状態がある．**search** の 3 つの検索失敗のところはこの 3 種類の状態に対応している．(1) と (2) の場合は検索失敗時点の p 節の下に挿入する．(3) の場合は検索失敗時点の p' と p 節の間に挿入する．3 種類の挿入処理をすべて定義 2 ～定義 4 によってすればよい．すなわち，挿入節に置く K' のビットパターンは照合失敗位置 P_{fail} からの尾部分 $K'[P_{fail} \cdots n]$ であり，P_{fail} も挿入節の添字とする．経路上の添字の順序関係は **search** によって保証される．図 **4** は中間節の挿入の例示（K' の位置 P_{fail} で '1' と仮定）である．なお，挿入節の親子節（p' と p 節）では図 4 のようなポインタの変更があるが，それぞれの I と K の変更は必要ない．

6. プログラムの実現と評価

6.1 データ構造

VP トライは可変長節を用いるので，VP トライに適用するデータ構造としては通常の配列構造が適切である．本論文で実現したプログラムでは節の I, LP, RP, K が配列構造の連続領域を用いる．I は 1 バイトで，LPと RP はそれぞれ 3 バイトで表す．VP トライの K の中止符号 'Λ' も節の区切りとして使う．

比較のために用意した P トライは 2 つの配列構造を使った．1 つは中間節を置き，節の I は 1 バイトで，LP と RP はそれぞれ 3 バイトである．すべてのキーをもう 1 つの配列上に置く．

6.2 空間性能評価

大規模な全文書（新聞記事，書誌情報，用語解説）から機械的に求めた[8]性質の異なる複数のインデックスファイル (News1～3, Book1～2, Term1～3)，UNIX-BSD 版の英単語ファイル web2☆ および UNIX のツール spell の辞書ファイル words を実験の対象とした．VP トライサイズを S_{VP}，従来の P トライのサイズを S_P として圧縮率 $r = S_{VP}/S_P$ を**表 1** に示した．この結果，VP トライのサイズは P トライより 20 ～ 36 %減少できることが分かった．

6.3 時間性能評価

VP トライと従来の P トライは同じ 2 分木索引構造である．したがって，検索手数の面では両構造が同等で，キーの総数を N とすれば，VP トライの検索手数は $O(\log_2 N)$ である．しかし，具体的なプログラムにした場合は両構造の時間性能を対比する必要もある．表 1 の各サンプルトライの平均深さ $\bar{d}$ に対して，

☆ Webster's Second International English Dictionary.

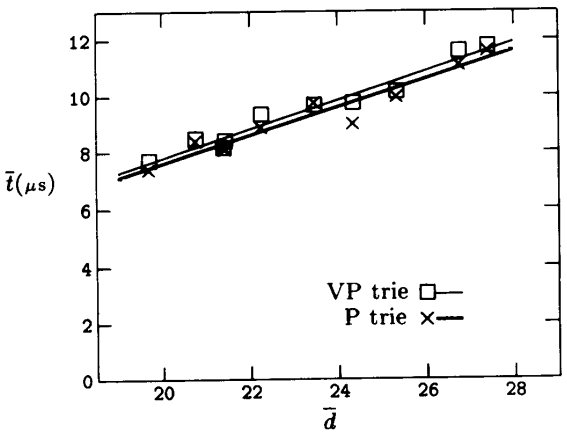


図5 VPトライとPトライの平均検索時間の対比
Fig. 5 Comparison of the average search time between VP and P tries.

平均検索時間 $\bar{t}$☆を**図5**に示す．VPトライの検索時間は従来のPトライに比べて平均2.5%しか増加していない．

7. おわりに

本論文ではパトリシアトライ（Pトライ）の空間性能を改善するために，可変長節パトリシアトライ（VPトライ）索引構造を提案した．中間節にキーを置くことによって，各キーに前方圧縮法を適用することが可能となった．これによって，従来のパトリシアトライに比較してより少ない空間で索引構造を表現できることを示した．具体的な例では，メモリ空間を20~36%減少できた．

本論文では，また，提案した構造に対して，検索アルゴリズムを示し，その実行時間は従来のパトリシアトライのそれとほぼ変わりないことを示した．

VPトライの削除アルゴリズムの検討は別の機会にゆずる．なお，解放されたメモリ空間の再利用は今後の課題である．

参考文献

1) Fredin, E.: Trie Memory, *Comm. ACM*, Vol.3, No.9, pp.490–500 (1960).
2) 青江順一：トライとその応用，情報処理学会学会誌，Vol.34, No.2, pp.224–251 (1993).
3) 熊谷 毅，中川征己：多字種に対応したトライの構成方法，電子情報通信学会論文誌，D-I, Vol.J77-D-I, No.11, pp.767–769 (1994).
4) Morrison, D.R.: PATRICIA-Practical Algorithm to Retrieve Information Coded in Alphanumeric, *J. ACM*, Vol.15, pp.514–534 (1968).
5) Knuth, D.: *The Art of Computer Programming*, Vol.3, Addison Wesley, Mass. (1973).
6) 程 亜非，桧垣泰彦，池田宏明：順次インデックスファイルに対する差分圧縮法の具体的提案，情報処理学会論文誌，Vol.36, No.9, pp.2175 2182 (1995).
7) 程 亜非，桧垣泰彦，池田宏明：前方圧縮を用いた可変長節パトリシアトライ索引構造，信学技報，DE95-1-8, pp.1–8 (1995).
8) 桧垣泰彦，池田宏明：字種分類インデキシングによるフルテキスト検索システム（CLAX），著作権法施行令第24条の規定によるプログラム登録番号P第2148号-1 (1990).

(平成7年9月27日受付)
(平成8年5月10日採録)

程 亜非（正会員）

昭和31年生．昭和57年中国北京工業大学第二分校計算機ソフトウェア工学科卒業．同年中国科学院計算技術研究所入所．平成3年千葉大学大学院修士課程修了．現在同大学大学院博士課程在学中．データ工学，分散環境での情報検索および計算機ネットワークに興味を持つ．電子情報通信学会会員．

桧垣 泰彦（正会員）

昭和57年千葉大学工学部電子工学科卒業．昭和59年同大学大学院工学研究科電子工学専攻修了．同年千葉大学工学部電子工学科（現電気電子工学科）助手となり現在に至る．インターネット上の情報検索システムに興味を持っている．電子情報通信学会会員．

池田 宏明（正会員）

昭和17年生．昭和41年千葉大学工学部電気工学科卒業．昭和43年千葉大学大学院工学研究科電子工学専攻（修士課程）修了．同年同大学工学部助手となり，現在同大学教授．この間，東京工業大学より工学博士の学位を取得．現在はマルチメディアシステムに興味を持っている．電気学会，電子情報通信学会，テレビジョン学会，IEEE各会員．

☆ DECstation 5000/25 Ultrix-4.2 gcc-2.6.3 の環境．