D-017

パトリシアを多分木に拡張したデータ構造の構築方法
Construction Method of Patricia Extended to Multiple Tree

松浦 寛生† 蔵満 琢麻† 尾崎 拓郎† 望月 久稔†
Nobutaka MATSUURA Takuma KURAMITSU Takuro OZAKI Hisatoshi MOCHIZUKI

1． はじめに

自然言語処理システムの辞書を中心に広く用いられるキー探索法として，木構造で表現される基数探索法には，遷移先が 1 つしか存在しない内部節点を削除したパトリシア [2] や，マルチウェイ基数探索法がある．マルチウェイ基数探索法のデータ構造として節点間の遷移を $O(1)$ で決定できる高速性とコンパクト性をあわせもつダブル配列 [1] がある．しかし，トライ上には遷移先が 1 つしか存在しない節点が存在する．

本論文では，比較位置情報を付加したダブル配列を用いてパトリシアを多分木に拡張し，トライ上の節点数を抑制した有効的なデータ構造の構築方法を提案する．

2． ダブル配列のデータ構造

ダブル配列は 2 つの配列 BASE，CHECK によりトライを表現し，トライにおける節点 $node$ からラベル $trans$ による節点 $next$ への遷移を式 (1) で定義する．ここで，BASE 値には遷移先節点の位置に対する基底位置を，CHECK 値には遷移元節点を格納する．以下，ダブル配列上の要素 x に関する BASE 値，CHECK 値をそれぞれ $B[x]$，$C[x]$ とする．

$$\begin{cases} next = B[node] + trans \\ C[next] = node \end{cases} \quad (1)$$

接尾辞において他キーとの非共通部を一次元配列 TAIL に格納する．このとき，葉の BASE 値を配列 TAIL 上の要素番号とし，内部節点と区別するために負値とする．以下，配列 TAIL 上の p 番目の要素を $T[p]$ とし，$T[p]$ 以降の要素からなる配列を $T + p$ とする．ダブル配列は，離散探索法を多分木とすることで遷移数を抑制するが，遷移上の経路に一方向分岐が存在する．

3． パトリシアを多分木に拡張した基数探索法

探索処理における木構造上での遷移数を抑制するため，ダブル配列上の一方向分岐を削除し，パトリシアを多分木に拡張する．以下，提案手法のデータ構造と追加処理について述べる．

3.1 データ構造

ダブル配列にキーの比較位置を格納する配列 POS を新たに用いる．キー key に対する節点 $node$ からの遷移先 $next$ を，配列 POS を用いて式 (2) と定義する．以下，ダブル配列上の要素 x に関する POS 値を $P[x]$ とする．

$$next = B[node] + key[P[node]] \quad (2)$$

キー集合 K={“decide#”, “deciable#”} に対して提案手法が実現するトライを図 1 に示す．ラベル#は終端記号を表し，#, a, b, . . . , s の内部表現値をそれぞれ 0, 1, 2, 3, . . . , 19 とする．また，配列 TAIL にはキー全体を格納する．

3.2 追加アルゴリズム

提案手法の追加処理は，トライ上を根から葉まで辿り，葉でキーを比較する．その後，新規節点の挿入位置まで遷移し，新規に葉を作成する．図 1 に “disk#”を追加したトライを図 2 に示す．以下，キーを追加する関数 Insert とこの関数内で分岐を作成するための関数 MakeBranch を示す．

関数 Insert(key)

(手順 1) 失敗比較位置の取得

トライ上でキー key の探索を行い，遷移が成功した最終の節点を $node$ とし，$node$ から葉まで各節点におけるラベルの内部表現値が最小の遷移を辿り，その葉を $leaf$ に設定する．その後，key と $leaf$ が持つキー（以下，$tail$ と呼ぶ）を比較し，key における失敗比較位置を $diff$ とする．

(手順 2) 挿入先節点の決定

比較位置が $diff$ 以下の節点まで，$node$ から各節点の CHECK 値を辿り，辿り着いた節点を新規挿入先節点 $branch$ とする．

(手順 3) 新規節点の作成

挿入先節点の位置に応じて，新規節点を作成する．そのケースは条件 A，B における関数 MakeBranch による分岐の作成と，条件 C における既存の分岐に新規節点を追加する三通りである．以下にその条件 A，B，C および各処理を示す．

[条件 A] 挿入先節点がトライ上の葉であった場合

$branch$ から key と $tail$ への分岐を作成するため，関数 MakeBranch($branch$, key, $T + (-B[leaf])$, $diff$) を呼ぶ．

[条件 B] 挿入先節点の比較位置が失敗比較位置未満であった場合

提案手法は，一方向分岐に対応する節点を作成していないため，トライの節点間に新たに節点を挿入する必要がある．このとき，$branch \leftarrow B[branch] + key[P[branch]]$ として，$branch$ を再設定し，条件 A と同様に関数 MakeBranch を呼ぶ．

[条件 C] 挿入先節点が葉でなく，挿入先節点の比較位置が失敗比較位置と同値であった場合

新規作成節点を new とし，$new \leftarrow B[branch] + key[diff]$ とする．その後，new を新たな葉として作成するが，new が未使用要素でない場合には，衝突を回避するため，トライ上の該当節点を移動 [1] させて，葉を作成する．

関数 MakeBranch(s, key, $tail$, $diff$)

節点 s を新たな分岐点として設定する．s からの新たな遷移種 $key[diff]$，$tail[diff]$ による新たな遷移先 $base + key[diff]$，$base + key[diff]$ が未使用要素となる BASE 値 $base$ を決定する．

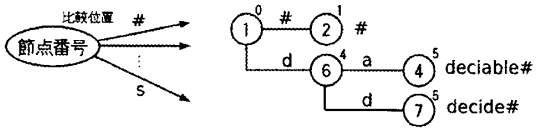
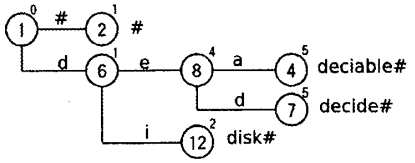


図 1 提案手法におけるトライの例

† 大阪教育大学, Osaka Kyoiku University



	1	2	3	4	5	6	7	8	9	10	11
BASE	2	-1		-10		3	-2	3			-20
CHECK	0	1		-6		1	-6	6			6
POS	0	1		5		1	5	4			2

	2		10		20
TAIL	decide#		deciable#		disk#

図 2　提案手法におけるトライに “disk#”を追加した例

次に $key[diff]$ に対応する新規節点 t' を作成する．$B[t']$，$C[t']$ および $P[t']$ には，それぞれ $B[s]$，s および $P[s]$ を格納する．このとき，t' が葉である場合，（手順 2）における $node$ の設定のために，$P[t']$ は $diff$ に 1 を加えた値とする．

その後，$tail[diff]$ に対応する新規節点 t を作成する．$B[t]$ にキー情報，$C[t]$ に s を格納する．$P[t]$ には $P[t']$ と同様に，$diff+1$ を格納する．最後に，遷移元節点 s において，$B[s]$ には $base$，$P[s]$ には $diff$ を格納する．

4．　実験による評価

提案手法の有効性を示すため，パトリシア [2]（以下，比較手法 P）とマルチウェイ基数探索法を実現したダブル配列 [1, 3]（以下，比較手法 D）を対象として，探索，追加に関する比較評価を行った．実験は，Intel Pentium 4 2.8GHz，Fedora Core 4 上で行い，URI をランダムに抽出した 500 万件（以下，キー集合 U とする）を用いた．遷移となる構成文字の内部表現値は ASCII コードとした．表 1 にその実験結果を示す．表 1 の総使用空間は実験環境におけるメモリ使用空間を byte 単位で示したものである．

はじめに，キー集合 U より 50 万件から 50 万件毎にキー数を増やした部分キー集合 U' を作成し，各手法について各キー集合 U' を登録したトライの道長を図 3 に示す．

図 3 より，提案手法は他の比較手法よりトライの道長を抑制する．これは，表 1 からも明らかで，道長が比較手法 D に対しては約 0.33 倍，比較手法 P に対しては約 0.31 倍となった．これに伴い，トライ上の探索における遷移回数が抑制され，表 1 に示すとおり他手法と比べて探索時間が向上した．

次に，キー集合 U' における追加時間を図 4 に示す．比較手法 D に対して一方向分岐を削除し，比較手法 P に対して道長を抑制して多分木を構築したため，表 1 に示すとおり平均分岐数が向上する．それに伴い，提案手法が新規節点の追加位置を決定する際，未使用要素パターンにおける探索処理の回数が他手法に比べて多くなる．そのため，図 4 より，提案手法は他手法に比べて追加時間を要する．

また，表 1 に示すとおり，提案手法は比較手法 D に対して，使用節点数を約 0.49 倍に抑制するが，配列 TAIL の使用要素数を約 5.49 倍要するため，総使用空間を約 2.22 倍要する．比較手法 P に対して提案手法は，配列 TAIL の使用要素数において同じであり，使用節点数を約 1.49 倍要するが，一節点あたりの使用空間を比較手法 P より抑制するため，提案手法は比較手法 P と総使用空間がほぼ変わらない．

表 1　追加処理，探索処理による実験結果（500 万件）

	提案	比較 D	比較 P
トライの道長 (100 万)	79.82	243.22	257.15
平均分岐数	2.97	1.49	2.00
追加時間 (s)	39.85	31.75	23.95
探索時間 (s)	13.00	20.31	21.04
トライの使用節点数 (10 万)	74.94	151.32	50.00
配列 TAIL の使用要素数 (1000 万)	29.73	5.41	29.73
総使用空間 (1000 万 byte)	38.72	17.41	37.73

5．　おわりに

本論文では，ダブル配列上の一方向分岐を削除し，パトリシアを多分木に拡張した構築方法を提案し，実験で道長の削減による探索速度の向上を示した．今後の課題として追加処理時間向上のため，未使用要素パターンの探索処理時間の向上を図ることが挙げられる．

参考文献

[1] J. Aoe. An efficient digital search algorithm by using a double-array structure. *IEEE Trans. on Software Engineering*, Vol. 15, No. 9, pp. 1066–1077, 1989.

[2] D. R. Morrison. Patricia practical algorithm to retrieve information coded in alphanumeric. *Journal of the ACM*, Vol. 15, pp. 514–534, 1968.

[3] 中村康正，望月久稔．圧縮ディジタル探索木における辞書情報更新の高速化手法．情報処理学会：データベース，Vol. 47, SIG 13（TOD 31），pp. 16–27, 2006.

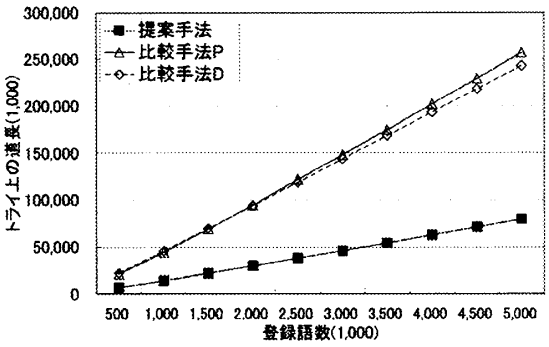


図 3　キー集合 U' に対する各手法におけるトライの道長

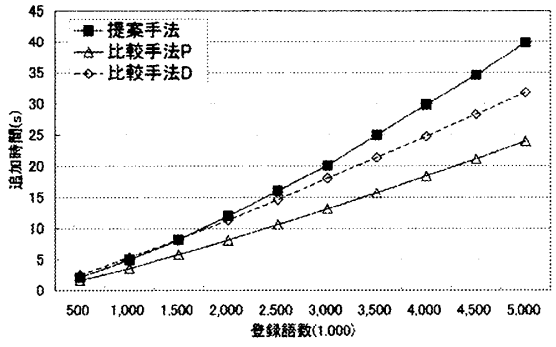


図 4　キー集合 U' に対する各手法における追加時間