Vol. 35 No. 5 情報処理学会論文誌 May 1994

最左部分語検索向き辞書データ構造：Prefix-Closed B-tree

冨浦洋一[†] 中村貞吾[††] 日高達[†]

最左部分語検索向き単語辞書データ構造として，Prefix-Closed B-tree を提案し，その性質，および，検索・挿入・削除・編集アルゴリズムとその効率について考察した．日本語は通常べた書きされるため，日本語の単語機械辞書は与えられた文字列のすべての最左部分語（“くるまだいそげ”の場合，“苦”，“来る”，“車”，“車代”）を効率よく検索できることが望ましい．Prefix-Closed B-tree は，拡張 B-tree と同じく，外部検索向きのデータ構造である B-tree を拡張したもので，一回の検索で与えられた文字列のすべての最左部分語を検索できる．挿入・削除アルゴリズムの効率の点で拡張 B-tree に劣るが，検索効率に優れたデータ構造である．

A New Data Structure for Searching Prefix Words: Prefix-Closed B-tree

Yoichi Tomiura,[†] Teigo Nakamura[††] and Toru Hitaka[†]

A new data structure called Prefix-Closed B-tree is proposed and its basic properties and algorithms for search, insertion and deletion and their efficiency are given. Japanese sentences are usually written without any spaces between words. It is therefore desirable that a dictionary system for the machine processing of Japanese sentence has the function to retrieve every prefix word of a given string efficiently (a prefix word in STRING means a real word which is in the leftmost part of STRING). Prefix-Closed B-tree is an extension of a data structure for external search called B-tree and can retrieve every prefix word in a given string in one search the same as Extended B-tree. It is superior to Extended B-tree in the efficiency of search, while it is inferior to Extended B-tree in insertion and deletion.

1. はじめに

データ構造論における辞書とは，任意に与えられた argument k に対して，k が集合 D に属すかどうかを判定する検索機能，D を $D \cup \{k\}$ に置き換える挿入機能，D を $D-\{k\}$ に置き換える削除機能の三つの基本機能を実現するためのデータ構造をいう．D が大きな集合の場合には，辞書を複数のブロックに分割してディスクパック等の二次記憶上に分散配置しておき，検索要求が生じるごとに，検索に必要なブロックを一次記憶に転送して検索 (external search) することになるが，二次記憶から一次記憶へのデータ転送は，一次記憶上での操作に比べ非常に時間がかかるため，

（a） 一回の検索で発生する，二次記憶から一次記憶へのデータ転送回数が少ない．

ことは検索効率を高める上で最も重要である．また，

（b） 二次記憶上の利用効率が良い．

（c） 挿入，削除の効率が良い．

ことも重要である．

単語単位にわかち書きされる欧米語と異なり，通常べた書きされる日本語の機械処理では，入力文の文字列を単語辞書を用いて単語列に変換する特殊な工程を必要とする．この工程では，与えられた文字列の左端に位置するすべての単語（最左部分語；例えば，“くるまだいそげ”の場合，“苦”，“来る”，“車”，“車代”）を求める必要があり，このような日本語の特殊性を考慮すると，処理全体の効率を上げるためには，

（d） 与えられた文字列のすべての最左部分語を効率よく検索することができる．

ことが重要である．

(a)〜(c) の要件を満たすデータ構造として，B-tree[1]

† 九州大学工学部情報工学科
Department of Computer Science and Communication Engineering, Faculty of Engineering, Kyushu University
†† 九州工業大学情報工学部知能情報工学科
Department of Artificial Intelligence, Faculty of Computer Science, Kyushu Institute of Technology

があるが，B-tree を単語辞書のデータ構造として用いた場合は，与えられた文字列の左部分列（“くるまだいそげ” の場合，“く”，“くる”，“くるま”，…，“くるまだいそげ”）を検索キーとして，複数回の検索をすることになり，(d) の要件を満たさない．そこで，(a)〜(c) に加え，(d) の要件も満たすデータ構造として拡張 B-tree[2] が提案されている．拡張 B-tree は，キー k の最左部分語が k の検索パス上に配置されるように B-tree を拡張したもので，その結果，拡張 B-tree は B-tree に比べて，（節の容量が等しい場合で比較すると）一つの節から出るポインタが減り，木が高くなる（すなわち，(a) の点で劣る）傾向にあるが，一回の検索で与えられた文字列のすべての最左部分語を検索でき，(d) の点で B-tree よりも優れている．

本論文では，拡張 B-tree よりもさらに効率的に与えられた文字列のすべての最左部分語を検索できる辞書データ構造として，Prefix-Closed B-tree[3] を提案する．Prefix-Closed B-tree は，拡張 B-tree と同じく，B-tree の拡張であり，一回の検索で与えられた文字列のすべての最左部分語を検索できる．拡張 B-tree がキー k の検索パス上にある節に k の最左部分語を配置したのに対して，Prefix-Closed B-tree は（必要ならばデータを重複させることにより）k を含む葉に k の最左部分語が含まれるようにしたもので，木の高さを拡張 B-tree よりも低く構成できる（すなわち，(a) (d) の点で拡張 B-tree よりも優れている）．しかし，(c) の点で拡張 B-tree に劣るため，検索向きの辞書データ構造である．本論文では，まず，Prefix-Closed B-tree の定義とこのデータ構造が持つ性質を述べた後，検索アルゴリズムとこれが (a), (d) の要件を満足することを示す．次に，挿入，削除，編集アルゴリズムおよびその効率を示す．

(d) の要件を満たすデータ構造として他に TREI[1] があるが，これは一次記憶上での検索，すなわち，内部検索 (internal search) 用のデータ構造であり，大規模辞書には向かない．なお，拡張 B-tree および本論文で提案する Prefix-Closed B-tree ともに節内のデータ構造を TREI にして，節内の最左部分語検索を高速にすることが可能である．

2. Prefix-Closed B-tree とその性質

定義 1 Prefix-Closed B-tree で扱うキー集合では，全順序 $\leq$ と順序 $\leftleftharpoons$ が定義されており，任意のキー x, y, z に対して，

a) $x \leftleftharpoons y \rightarrow x \leq y$,

b) $(x \leq y \leq z \wedge x \leftleftharpoons z) \rightarrow x \leftleftharpoons y$

が成立する． □

単語辞書の場合，$x \leq y$ を ‘辞書式全順序で x が y に等しいか y より小さい（前である）’ という関係，$x \leftleftharpoons y$ を ‘x は y の最左部分語である’ という関係と考えると，定義 1 が成立する．また，‘$<$’, ‘$\llcorner$’ を

$$x < y \overset{\text{def}}{=} x \leq y \wedge x \neq y,$$

$$x \llcorner y \overset{\text{def}}{=} x \leftleftharpoons y \wedge x \neq y$$

と定義する．

以後の証明で頻繁に使われる関係として次の補題を上げておく．これは定義 1 から容易に証明できる．

補題 1 任意のキー x, y, z に対して，

$$(x \leq y \wedge z \leftleftharpoons y) \rightarrow (x \leq z \leq y \vee z \leftleftharpoons x). \qquad (1)$$

証明 $x \leq y \wedge z \leftleftharpoons y$ と仮定する．$z \leftleftharpoons y$ であるから，定義 1 より，$z \leq y$. したがって，$x \leq z \leq y \vee z \leq x \leq y$ が成立する．$z \leq x \leq y$ ならば，$z \leftleftharpoons y$ と定義 1 より，$z \leftleftharpoons x$. したがって，(1) 式が成立する． □

辞書の見出し語の全体集合を D と記す．D はキー集合の部分集合である．D が持つ性質として，以下の次数を定義する．

定義 2 任意のキー z に対して，$|\{y | y \llcorner z,\ y \in D\}|$, $|\{y | z \llcorner y,\ y \in D\}|$ がそれぞれ M_d, M_u 以下であるとき，D は下位次数が M_d，上位次数が M_u であるという．ただし，S が集合のとき，$|S|$ は S の要素数を示す． □

$x \llcorner y$ を ‘x は y の真の（つまり，$x=y$ の場合を除く）最左部分語である’ という関係と考えると，下位次数 M_d は，一つのキーの最左部分語となる D 中のキーの個数の上限を，上位次数 M_u は，共通のキーを最左部分語とする D 中のキーの個数の上限を示している．

本論文で提案する Prefix-Closed B-tree は大容量の辞書向きのデータ構造として広く用いられている B-tree を拡張したものであり，次のように定義される．

定義 3 Prefix-Closed B-tree は，各節が

$[p_0\ x_1\ p_1 \cdots x_m\ p_m]$

の形をした多分木（multiway tree）で，以下の性質a)〜e）を満足するものである．ただし，x_i はキーであり，p_i は i 番目の子へのポインタである．葉である節の場合には，ポインタはなく，キーのみで構成される．

a）節中のキーはソートされており，$x_1<x_2<\cdots<x_m$ である．

b）ポインタ p で指される，葉以外の任意の節

$[p_0\ x_1\ p_1\ \cdots\ x_m\ p_m]$

に対して，

$D(p_0)=\{y\mid y<x_1,\ y\in D(p)\}$,

$D(p_i)=\{y\mid x_i\le y<x_{i+1}\vee y\vdash x_i,\ y\in D(p)\}$　$(0<i<m)$,

$D(p_m)=\{y\mid x_m\le y\vee y\vdash x_m,\ y\in D(p)\}$

が成立する．ただし，$D(p)$ はポインタ p で指される部分木の葉に含まれるキーの集合を表す．

c）根からすべての葉へのパス長は等しい．

d）根以外の節は，$\lfloor C/2\rfloor$ 以上 C 以下の個数のキーを持つ．ただし，$\lfloor C/2\rfloor>M_d$．$\lfloor a\rfloor$ は実数 a 以下の最大の整数を表す．

e）根には1個以上のキーが含まれる．　□

$C=6$ の Prefix-Closed B-tree の例（ただし，$M_d=2$ とする）を，**図1**に示す．

ポインタ p で指される節を $N(p)$ で表し，ポインタ p で指される（部分）木を $T(p)$ で表す．定義3の中で示したように，$D(p)$ は $T(p)$ の葉に含まれるキーの集合を表す．特に Prefix-Closed B-tree の根へのポインタを root と記す．見出し語の全体集合 D は D(root) に等しい．また，葉に含まれるキーと葉以外の節に含まれるキーを区別し，前者をデータキー，後者を検索キーと呼び，$T(p)$ 中の検索キーの集合を $I(p)$ で表す．検索キーの集合もキー集合の部分集合である．x x_i x' などで検索キーを表し，y y_i y' などでデータキーを表し，z z_i z' などで検索キーまたはデータキーを表す．節 N から葉までのパス長を N の高さと呼び，根から N までのパス長を N の深さと呼ぶ．葉は高さ0，根は深さ0である．

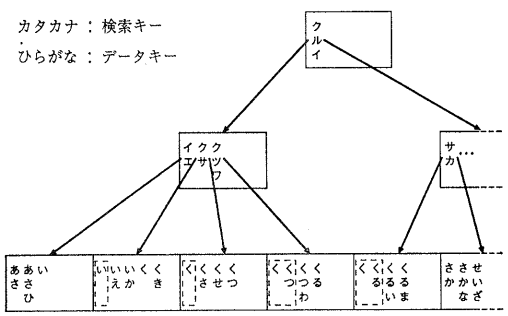

図1　Prefix-Closed B-tree の例
Fig. 1　An example of Prefix-Closed B-tree.

以下に示すように，検索キーに関して B-tree と同様の性質が成り立つ．

定理1　Prefix-Closed B-tree の葉以外の任意の節 $N(p)$

$[p_0\ x_1\ p_1\ \cdots\ x_m\ p_m]$

に対して，

$\forall x[x\in I(p_0)\to x<x_1]$,

$\forall x[x\in I(p_i)\to x_i<x<x_{i+1}]$　$(0<i<m)$,

$\forall x[x\in I(p_m)\to x_m<x]$.

が成立する．

証明　$N(p)$ に対して，

$\forall x[x\in I(p_i)\to x<x_{i+1}]$　$(0\le i<m)$,　（2）

$\forall x[x\in I(p_i)\to x_i<x]$　$(0<i\le m)$　（3）

が成立することを証明する．

（2）式が成立しない，すなわち，

$x\in I(p_i)\wedge x_{i+1}\le x$

なるキー x が存在すると仮定する．x が含まれる節へのポインタを q，x のすぐ右のポインタを r とする．定義3 b）より，

$D(r)\subseteq\{y\mid x\le y\vee y\vdash x,\ y\in D(q)\}$.　（4）

$D(q)\subseteq D(p_i)$ であるから，定義3 b）より，

$D(q)\subseteq\{y\mid y<x_{i+1},\ y\in D(p)\}$.　（5）

（5）式を（4）式に代入して，

$D(r)\subseteq\{y\mid(x\le y\vee y\vdash x)\wedge y<x_{i+1},\ y\in D(p)\}$.　（6）

$x_{i+1}\le x$，$D(p)\subseteq D$ および（6）式より，

$D(r)\subseteq\{y\mid y\vdash x,\ y\in D\}$.

したがって，$|D(r)|\le M_d$ である．これは，定義3 d）より得られる関係 $M_d<|D(r)|$ に矛盾する．したがって，（2）式が成立する．

次に，（3）式が成立しない，すなわち，

$x\in I(p_i)\wedge x\le x_i$

なる x が存在すると仮定する．x が含まれる節へのポインタを q，x のすぐ左のポインタを r とする．定義3 b），$D(q)\subseteq D(p_i)$，$x\le x_i$，$D(p)\subseteq D$ より，（2）式の証明の場合と同様にして，$|D(r)|\le M_d$ が得られる．これは，定義3 d）より得られる関係 $M_d<|D(r)|$ に矛盾する．したがって，（3）が成立する．

（2）（3）式より，本定理が成立する．　□

$T(p)$ を Prefix-Closed B-tree の任意の部分木とする．$N(p)$ の先祖の節 [… x q …] が存在し，$T(p)$ が $T(q)$ の最左の葉を含む $T(q)$ の部分木となっているとき，x を $T(p)$ の左分離キーと呼ぶ．また，$N(p)$ の先祖の節 [… q x …] が存在し，$T(p)$ が $T(q)$ の最右の葉を含む $T(q)$ の部分木となっているとき，x を $T(p)$ の右分離キーと呼ぶ．また，T_1 の右分離キーが x で，T_2 の左分離キーも x であるとき，T_1 と T_2 は隣接していると言い，x を T_1 と T_2 の分離キーと呼ぶ．また，葉に対してもこれらの用語（左分離キー，右分離キー，分離キー，隣接）を用いる．

定理 2 Prefix-Closed B-tree の任意の部分木 $T(p)$ に対して，$T(p)$ の左分離キーを x_L，右分離キーを x_R とすると，

$$D(p)=\{y \mid x_L \leq y < x_R \vee y \llcorner x_L,\ y \in D\}$$

が成立する．ただし，左分離キー，右分離キーが存在しない場合は，それぞれ，上式において x_L を含む条件，x_R を含む条件を除いた式が成立する．たとえば，左分離キーが存在しない場合は，

$$D(p)=\{y \mid y < x_R,\ y \in D\}$$

である．

証明 $N(p)$ の深さに関する帰納法により，定義 1，補題 1，定義 3 b)，定理 1 を用いて容易に証明できる．□

定理 2 は，Prefix-Closed B-tree の部分木 T は，その左分離キーを x_L，右分離キーを x_R とすると，$x_L \leq y < x_R \wedge y \in D$ なるデータキー y と $y \llcorner x_L \wedge y \in D$ なるデータキー y を含むことを示している．前者を T の固有キー，後者を T の補助キーと呼ぶ．

定理 3 Prefix-Closed B-tree に含まれる任意のポインタ p に対して，

$$\forall y \forall z[z \in (D(p) \cup I(p)) \wedge y \in D \wedge y \eqcirc z \rightarrow y \in D(p)] \quad (7)$$

が成立する．

証明 $N(p)$ の深さに関する帰納法で証明する．

i) basic step ($N(p)$ が深さ 0 の節，すなわち根のとき)：$D(p)=D$ であるから，自明．

ii) inductive step：深さ d の葉でない節

$[p_0\ x_1\ p_1 \cdots x_m\ p_m]$

を指すポインタ p に関して，(7)式が成立すると仮定する．

ii-1) $z \in (D(p_0) \cup I(p_0)) \wedge y \in D \wedge y \eqcirc z$ が成立すると仮定する．$z \in (D(p) \cup I(p))$ であるから，帰納法の仮定より，$y \in D(p)$ である．また，$z \in (D(p_0) \cup I(p_0))$，定義 3 b)，定理 1 より，$z < x_1$ であり，$y \eqcirc z$，定義 1 より $y \leq z$ であるから，$y < x_1$．したがって，$y < x_1 \wedge y \in D(p)$，定義 3 b) より，$y \in D(p_0)$．

ii-2) $z \in (D(p_i) \cup I(p_i)) \wedge y \in D \wedge y \eqcirc z$ が成立すると仮定する $(0<i<m)$．ii-1) と同様にして $y \in D(p)$，また，$z \in (D(p_i) \cup I(p_i))$，定義 3 b)，定理 1 より，$x_i \leq z < x_{i+1}$ または $z \llcorner x_i$ である．

$x_i \leq z < x_{i+1}$ の場合：$x_i \leq z \wedge y \eqcirc z$，補題 1 より $x_i \leq y \leq z \vee y \eqcirc x_i$ である．したがって，$(x_i \leq y < x_{i+1} \vee y \llcorner x_i) \wedge y \in D(p)$ であり，定義 3 b) より，$y \in D(p_i)$．

$z \llcorner x_i$ の場合：$y \eqcirc z$ と $\llcorner$ の推移性より $y \llcorner x_i$．これと，$y \in D(p)$，定義 3 b) より，$y \in D(p_i)$．

ii-3) $z \in (D(p_m) \cup I(p_m)) \wedge y \in D \wedge y \eqcirc z$ が成立すると仮定する．ii-2) の場合と同様にして $y \in D(p_m)$．

ii-1) ii-2) ii-3) より，深さ $d+1$ の節を指すポインタ p_i に対して(7)式が成立する．□

定理 2 より，データキーはいずれかの葉の固有キーになっていることがわかる．また，各葉には他の葉の固有キーになっているキー，すなわち，補助キー（図 1 の破線で囲まれたキー）が含まれる．各葉における補助キーをすべて取り除いたものが B-tree となることが，定義 3，定理 1，定理 2 より，容易に示せる．$\eqcirc$ を最左部分語の関係と考えた場合，葉における補助キーはその葉の左分離キーの最左部分語であるので，一つの葉に含まれる（他の葉に含まれるという意味で）冗長なキーの総数は，高々 10 程度である．これだけを重複させることにより，定理 3 で示したように，k に対する一回の検索で $y \eqcirc k \wedge y \in D$ なるすべてのキー y を検索することができる．

また，$\eqcirc$ を最左部分語 (Prefix) の関係とすると，定理 3 より，Prefix-Closed B-tree の任意の部分木 $T(p)$ のデータキーの集合 $D(p)$ に対して，$D(p)$ の任意のキー y の最左部分語はすべて $D(p)$ の要素である（つまり，$D(p)$ は prefix に関して閉じている）ことがわかる．これが本論文で提案する B-tree を拡張した辞書データ構造を Prefix-Closed B-tree と呼んでいる理由である．

系 1 $T(p)$ を Prefix-Closed B-tree T の部分木，x をその右分離キーとすると，

$$\{y \mid y \llcorner x,\ y \in D\} \subseteq D(p), \qquad (8)$$

が成立する．

証明 定理2より，$\{y \mid y < x,\ y \in D\}$ の要素のキーのうち，$\leq$ に関して最大のキーが $D(p)$ に含まれることがわかる．このキーを y_0 とする．$y \llcorner x \wedge y \in D$ と仮定する．定義1より $y < x$ であるから，y_0 の定義より $y \leq y_0 < x$ である．したがって，定義1より，$y \eqsim y_0$ である．$y_0 \in D(p) \wedge y \in D \wedge y \eqsim y_0$ であるから，定理3より，$y \in D(p)$. したがって，(8)式が成立する． □

Prefix-Closed B-tree T の任意の部分木を $T(p)$，その左分離キーを x，x を右分離キーとする T の任意の部分木を $T(q)$ とすると，定理2，系1より，$y \llcorner x \wedge y \in D$ なるすべてのキー y が $T(p)$ に補助キーとして含まれ，かつ，$T(q)$ にデータキーとして含まれていることがわかる．

3. 検　索

定義 4 葉でない節 N

$[p_0\ x_1\ p_1 \cdots x_m\ p_m]$ に対して，

$$c(k) \overset{\text{def}}{=} \begin{cases} 0 \cdots k < x_1 \text{ のとき}, \\ i \cdots x_i \leq k < x_{i+1} \text{ のとき}, \\ m \cdots x_m \leq k \text{ のとき}. \end{cases}$$

で $c(k)$ を定義し，ポインタ $p_{c(k)}$ を k に対応する（節 N の）ポインタと呼ぶ． □

B-tree と同様にして，任意に与えられたキー k に対して，k を含む葉 N を求めることができる．定理3より，$y \eqsim k \wedge y \in D$ なるすべてのキー y が節 N 中に存在することがわかる．したがって，任意に与えられたキー k に対して，$y \eqsim k$ なるすべてのキー y の検索は，節 N を検索することによって求められる．

検索アルゴリズム 任意に与えられたキー k に対して，$y \eqsim k$ なるすべてのキーの検索は，根から出発して，次の操作：

> たどられた節 N が葉ならば，N の中から，$y \eqsim k$ なるキーをすべて出力する．N が葉でないならば，k に対応する N のポインタ（定義4参照）をたどる．

を再帰的に繰り返すことにより行われる． □

検索アルゴリズムの効率は木の高さで評価される．木の高さと領域の大きさ（木の節数に節の大きさを掛けたもの）に関する定理を次に与える．

定理 4 Prefix-Closed B-tree の木の高さ H，および，占有する領域の大きさ S に関して，以下の関係が成立する．

$$H \leq 1 + \log_{(1+\lfloor C/2 \rfloor)} \frac{|D|}{2\lfloor C/2 \rfloor - 2M_d}, \qquad (9)$$

$$S \leq C\left(1 + \frac{|D|(1+\lfloor C/2 \rfloor)}{\lfloor C/2 \rfloor(\lfloor C/2 \rfloor - M_d)} - \frac{2}{\lfloor C/2 \rfloor}\right). \qquad (10)$$

証明 $\lfloor C/2 \rfloor$ を a，木の高さを H とすると，根は1個以上，根以外の節は a 個以上のキーを持つので，深さ $i\ (i > 0)$ の節数は，$2(1+a)^{i-1}$ 以上である．したがって，葉の総数を n とすると，

$$2(1+a)^{H-1} \leq n. \qquad (11)$$

また，一つの葉に含まれる固有キーの個数は $a - M_d$ 以上である．したがって，

$$n(a - M_d) \leq |D|. \qquad (12)$$

(11)(12)式より，(9)式が成立する．

B-tree と同様，Prefix-Closed B-tree T 中の任意の検索キーに対して，それを分離キーとする隣接する葉 L_i L_{i+1} が一組決まり，逆に，隣接する任意の葉 L_i L_{i+1} の分離キーが T 中の検索キーの中から一つ決まる．したがって，検索キーの総数は $n-1$ である．根は1個以上，根と葉以外の節は a 個以上の検索キーを持つので，節の総数を Q とすると，検索キーの総数は，$a(Q-n-1)+1$ 以上である．したがって，

$$n - 1 \geq a(Q - n - 1) + 1. \qquad (13)$$

(12)(13)式より，(10)式が成立する． □

日本語単語辞書では，$C \gg 2M_d$, $C \gg 1$ が成立する．この場合，木の高さと領域の大きさに関して以下の系が成立する．

系 2 $C \gg 2M_d$, $C \gg 1$ の場合，

$$H \leq 1 + \log_{(1+\lfloor C/2 \rfloor)} \frac{|D|}{C}, \quad S \leq 2|D|$$

である． □

Prefix-Closed B-tree では同一のキーが異なる葉に重複して含まれるので，占有する領域は B-tree よりも大きい．しかし，拡張 B-tree でも，同一の C（節の容量）に対して，一つの節から出せるポインタが B-tree, Prefix-Closed B-tree に比べて少ないため，葉以外の節の数が増え，全体として占有する領域は B-tree に比べ大きい．実際に，ワーストケースを比較

すると，拡張 B-tree の占有する領域 S' に関しては，

$$S'<C\cdot\left(\frac{\lfloor C/2\rfloor-M_d+1}{(\lfloor C/2\rfloor-M_d)^2}\cdot|D|+1\right)$$

が成立する[2] ので，(10)式と上式を比較すると，$S\leq S'$ がいえる．

拡張 B-tree の木の高さ H' に関しては，

$$H'<\log_{(1+a')}\left(\frac{a'}{2}\cdot\frac{\lfloor C/2\rfloor-M_d+1}{(\lfloor C/2\rfloor-M_d)^2}\cdot|D|+1\right),$$

が成立する[2]．ただし，$a'=\lfloor C(2M_d+2)\rfloor$ である．H と H' に関しては一概に大きさの比較はできないが，$C\gg 2M_d$, $C\gg 1$ の場合，

$$H\leq 1+\log_{(1+a')}\frac{|D|}{2(M_d+1)},$$

であるので，上記の系と比較すると，高さは Prefix-Close B-tree の方が拡張 B-tree より低いことがわかる．

4. 挿入・削除

キー k を Prefix-Closed B-tree に挿入する手順を与えるために，まず，*Split* と *InsertAux* の二つの手続きを与える．B-tree と同様，挿入によって一つの節に含まれるキーの数が C を越える場合があり，この場合，節の分割を行う必要がある．*Split*(p) は，節 $N(p)$ に対して分割を行う手続きであり，分割により新しくできた節へのポインタを q，分割後の $N(p)$ と $N(q)$ の分離キーを x とすると，列 xq を返す．*InsertAux*(p, k) は部分木 $T(p)$ にキー k を再帰的に挿入する手続きである．挿入により節 $N(p)$ のキーの数が C を越える場合は，*Split*(p) が実行され，*InsertAux*(p) は *Split*(p) が返す値を返す．$N(p)$ のキーの数が C を越えない場合は，*InsertAux*(p, k) は ϵ (空列) を返す．

手続き *Split*(p)

$N(p)$ が葉の場合：$N(p)$ を

$[y_1\ \cdots\ y_t\ y_{t+1}\ \cdots\ y_m]$

とする．ただし，$t=\lfloor m/2\rfloor$ である．y_t と y_{t+1} の間で分割し，$N(p)$ を

$[y_1\ \cdots\ y_t]$

に変更する．さらに，$y_i \llcorner y_{t+1}$ なるすべての $y_i(0\leq i\leq t)$ を $\delta_1, \cdots, \delta_u$ とすると（ただし，$j<k$ ならば $\delta_j<\delta_k$），新たにポインタ q で指される節

$[\delta_1\ \cdots\ \delta_u\ y_{t+1}\ \cdots\ y_m]$

を作り，列 $y_{t+1}\ q$ を返す．

$N(p)$ が葉でない場合：$N(p)$ を

$[p_0\ x_1\ \cdots\ x_t\ p_t\ x_{t+1}\ p_{t+1}\ x_{t+2}\ \cdots\ x_m\ p_m]$

とする．ただし，$t=\lfloor m/2\rfloor$ である．x_{t+1} の前後で分割し，$N(p)$ を

$[p_0\ x_1\ \cdots\ x_t\ p_t]$

に変更し，新たに q で指される節

$[p_{t+1}\ x_{t+2}\ \cdots\ x_m\ p_m]$

を作り，列 $x_{t+1}\ q$ を返す． □

手続き *InsertAux*(p, k)

Step 1. $N(p)$ が葉ならば，k を（$N(p)$ 内のキーが $\leq$ に関して昇順になるような）$N(p)$ の適切な位置に挿入する．$N(p)$ が葉でないならば，

- r は k に対応する $N(p)$ のポインタ（定義4参照）である．
- r は $k \llcorner x$ なるキー x のすぐ右のポインタである．

のいずれかが成立するすべてのポインタ r に対して，*InsertAux*(r, k) を行い（これが返す値を α とする），$N(p)$ の r の右に α を挿入する（**図2** 参照）．

Step 2. $N(p)$ のキーの個数が C を越える場合は，手続き *Split*(p) により分割を行い，この手続きの返す値を返す．$N(p)$ のキーの個数が C を越えない場合は，ϵ を返す． □

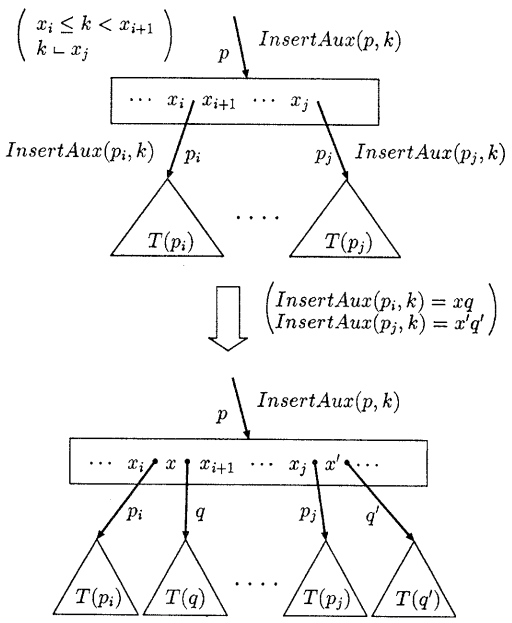

図 2 *InsertAux*(p, k)
Fig. 2 *InsertAux*(p, k).

Prefix-Closed B-tree $T(root)$ にキー k を挿入する手順 $Insert(root, k)$ を以下に与える．返す値は，k 挿入後の Prefix-Closed B-tree へのポインタである．

挿入アルゴリズム *Insert*(*root*, *k*)

手続き $InsertAux(root, k)$ を行い，これが返す値を α とする．α が ϵ でないならば，新しく節

$[root\ x\ q]$； $\alpha=xq$

を作り，この節へのポインタを返す．α が ϵ ならば $root$ を返す． □

$InsertAux(p, k)$ の実行により，$T(p)$ が変化する．混乱を避けるために，$InsertAux(p, k)$ を行う前の $T(p)$ $D(p)$ $N(p)$ をそれぞれ $T_0(p)$ $D_0(p)$ $N_0(p)$ で，$InsertAux(p, k)$ の Step 1. が終了した時点での（すなわち，$Split(p)$ を行う前の）$T(p)$ $D(p)$ $N(p)$ をそれぞれ $T_1(p)$ $D_1(p)$ $N_1(p)$，$InsertAux(p, k)$ が終了した時点での（すなわち，$Split(p)$ を行った後の）$T(p)$ $D(p)$ $N(p)$ をそれぞれ $T_2(p)$ $D_2(p)$ $N_2(p)$ で表す．また，$InsertAux(p', k)$ が実行されない $T(p)$ の部分木 $T(p')$ は $InsertAux(p, k)$ の実行の前後で変化がないので，$T_0(p')=T_1(p')=T_2(p')=T(p')$，$D_0(p')=D_1(p')=D_2(p')=D(p')$，$N_0(p')=N_1(p')=N_2(p')=N(p')$ とする．

補題 2 $T_1(p)$ は，$N_1(p)$ に含まれるキーの個数が，$N_1(p)$ が葉の場合は $C+1$，葉でない場合は $C+1$ 以上 $2C+1$ 以下であり，$N_1(p)$ に対する定義 3 d) の性質以外，Prefix-Closed B-tree のすべての性質を満足しているとする．$Split(p)$ を実行し，これが返す値を xq とすると，$T_2(p)$ $T_2(q)$ は $T_1(p)$ と同じ高さの Prefix-Closed B-tree であり，

$D_2(p)=\{y \mid y<x,\ y\in D_1(p)\}$,

$D_2(q)=\{y \mid x\leq y \vee y \frown x,\ y\in D_1(p)\}$

が成立する．

証明 手続き $Split$ の定義，定義 1，定義 3 より明らか． □

補題 3 $T_0(p)$ は Prefix-Closed B-tree であり，$InsertAux(p, k)$ を行った結果，これが返す値を α とする．$\alpha=\epsilon$ ならば，$T_2(p)$ は $T_0(p)$ と同じ高さの Prefix-Closed B-tree であり，

$$D_2(p)=D_0(p)\cup\{k\} \tag{14}$$

が成立し，$\alpha=xq$ ならば，$T_2(p)$ $T_2(q)$ は $T_0(p)$ と同じ高さの Prefix-Closed B-tree であり，

$$D_2(p)=\{y \mid y<x,\ y\in D_0(p)\cup\{k\}\}, \tag{15}$$

$$D_2(q)=\{y \mid x\leq y \vee y \frown x,\ y\in D_0(p)\cup\{k\}\} \tag{16}$$

が成立する．また，$N_0(p)$ が葉でないとき，この節に含まれるキーのうち $\leq$ に関して最小（左端）および最大（右端）のキーをそれぞれ $x_{\min}$ $x_{\max}$ とすると，

$$x_{\min}\leq x\leq x_{\max}, \tag{17}$$

が成立する．

証明 $T_0(p)$ の高さ H に関する帰納法で証明できるが，ここでは省略する（付録参照）． □

定理 5 Prefix-Closed B-tree $T(root)$ に対して，挿入アルゴリズム $Insert(root, k)$ を行った結果，これが返す値を $root'$ とすると，$T(root')$ は Prefix-Closed B-tree であり，

$D(root')=D(root)\cup\{k\}$

である．

証明 定義 3，挿入アルゴリズムの定義，補題 3 より自明． □

挿入アルゴリズムの効率は，たどる節の個数で評価される．

定理 6 Prefix-Closed B-tree $T(root)$ に対する挿入操作では，最大，

$$\frac{(\lfloor C/2\rfloor+1)(M_u-1)}{\lfloor C/2\rfloor(\lfloor C/2\rfloor-M_d)}+2H+1 \tag{18}$$

個の節をたどる．ただし，H は $T(root)$ の高さである．

証明 $a=\lfloor C/2\rfloor$，挿入するキーを k とする．また，$L_0, L_1, \cdots, L_s$ のみが，k を挿入すべき葉であるとする．定理 5 より，$Insert(root, k)$ を実行した場合，これらの節はすべてたどられる．また，補題 3 より，$InsertAux(root, k)$ が節 $N(p)$ をたどるならば，$T(p)$ の葉の少なくとも一つには k が挿入される（無駄な節はたどらない）ことがわかる．したがって，$T(root)$ に対する k の挿入操作でたどられる節は，$L_0, \cdots, L_s$，とこれらの先祖の節のみである．$k \frown y \wedge y\in D$ なるすべての y を $\leq$ に関して昇順に並べた列を，$y_1 \cdots y_t$ とする．定義 2 より，

$$t\leq M_u \tag{19}$$

である．これらのキーが，固有キーとして，L_0 には 0 個以上，L_s には 1 個以上，L_0 と L_s の間の各葉には $a-M_d$ 個以上含まれる．したがって，

$$(a-M_d)(s-1)+1\leq t \tag{20}$$

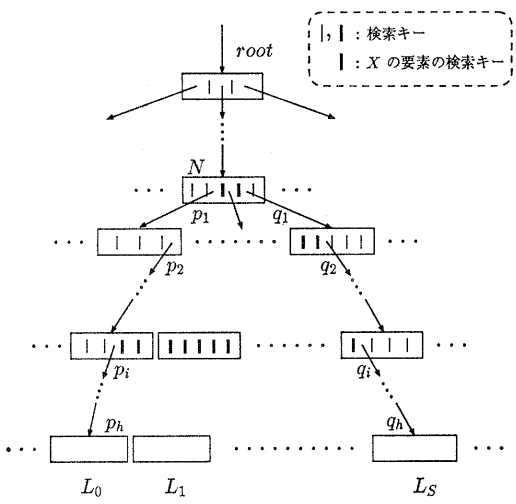

図 3 挿入アルゴリズムでたどられる節
Fig. 3 Nodes traced in the insertion algorithm.

である．また，L_0 と L_s の共通の先祖の節のうち，最も低い節を N とし，その高さを h とする．また，N から L_0, L_s をたどるためのポインタをそれぞれ，$p_1, p_2, \cdots, p_h$，および $q_1, q_2, \cdots, q_h$ とする．p_i と $q_i(i=1, \cdots, h)$ の間にあるキーから成る集合を X とする．Xの任意のキーに対して，これを分離キーとする隣接する葉 L_j と L_{j+1} が一組決まり，逆に任意の隣接する葉 L_j と L_{j+1} の分離キーが X 中より一つ決まる．したがって，X の要素数は s である．Xの要素のキーが，$N(p_i)$, $N(q_i)$ $(i=1, \cdots, h-1)$では0個以上含まれ（なぜならば，たとえば，p_{i+1} が $N(p_i)$ の最右のポインタであることがあるので），$N(p_i)$ と $N(q_i)$ $(i<h)$ の間の節では a 個以上含まれ，N では1個以上含まれる．*InsertAux*$(root, k)$ を実行したときにたどられる節の総数を Q とすると，

$$(Q-(s+1)-2(h-1)-1-(H-h))a+1 \leq s \quad (21)$$

である．(19) (20) (21)式より

$$Q \leq \frac{(a+1)(M_u-1)}{a(a-M_d)}+H+h+1 \quad (22)$$

が成立する．$0 \leq h \leq H$ であるから，(22)式より，本定理が証明できる． □

拡張 B-tree における挿入操作でたどる節の数は，木の高さに等しい．これと(18)式を一概に比較することはできないが，それぞれのデータ構造を現実的な大きさ（見出し語数 10 万程度）の日本語単語辞書に適用した場合で比較してみる．M_d は 10 程度，M_u は1万程度である．仮に，

見出し語数：100000
M_d ：10
M_u ：8000
C ：200

とすると，Prefix-Closed B-tree の場合は，高さが2，挿入でたどる節数は最悪 94，拡張 B-tree の場合は，高さが3，したがって挿入でたどる節数も最悪3である．このように挿入操作の効率に関しては，Prefix-Closed B-tree は拡張 B-tree に比べかなり劣ることがわかる．

アクセスする節の個数は，たどられたすべての節で分割が起こる場合，最悪となる．一つの節に対して，*Split* 操作で二つの節の write 操作を行う．根が分割された場合は，さらに新しい根を作成する．したがって，たどる節の個数を Q とすると，アクセスする節の個数は $3Q+1$ 以下である．

キー $k(\in D(root))$ を $T(root)$ から削除する操作は，基本的には挿入の逆の操作であり，$T(p)$ から k を削除し，削除後の $N(p)$ のキーの個数を返す再帰的な手続き *DeleteAux*(p, k) を用いて行われる．*DeleteAux*(p, k) は，$N(p)$ が葉ならば $N(p)$ から k を削除し，$N(p)$ が葉でないならば，$k \in D(r)$ であるようなすべての $T(p)$ の子の木 $T(r)$ に対して以下の操作：

DeleteAux(r, k) を適用し，これが返す $N(r)$ のキーの個数が $\lfloor C/2 \rfloor$ 未満であるならば，$N(r)$ の隣の節との併合・再分割を行う．

を行い，$N(p)$ のキーの個数を返す．削除アルゴリズムの詳細，およびその妥当性，効率（挿入アルゴリズムとほぼ同じ）に関しては，紙面の都合上本論文では省略する．

5. 編　　集

ソート済みの原データに対して，各節に含まれるキーの個数が C である（ただし，各高さの右端の節を除く）ような Prefix-Closed B-tree を構築する手順（編集アルゴリズム）を与える．これは厳密には，Prefix-Closed B-tree ではない，つまり，右端の節が定義 3 d）の性質を満足しないが，この性質は，領域の有効利用のために設定されたもので，いくつかの節に関して，この性質が成り立たなくても，定義3で定義した Prefix-Closed B-tree との本質的な違いはなく，定理1，定理2，定理3が成立し，検索アルゴリ

ズムもそのまま適用できることが容易にわかる．挿入アルゴリズムを用いて Prefix-Closed B-tree の構築はできるが，編集アルゴリズムを用いて構築した方が，木の高さ・領域に関してよりよい Prefix-Closed B-tree が構築できる．

編集アルゴリズムを後述の手続き *EditAux* を用いて以下のように与える．*EditAux*(h, α) は，α を高さ h の右端の節に挿入する手続きである．

編集アルゴリズム 原データから計算される高さ（定理7参照）を H_e とすると，$p_{rm}(0)=nil, \cdots, p_{rm}(H_e)=nil$ と初期化する．$p_{rm}(h)$ は高さ h の右端の節へのポインタである．原データからキー k を一つずつ昇順に取り出し，*EditAux*$(0, k)$ を実行する．すべてのキーに対してこの操作が終了した時点で，$p_{rm}(h) \neq nil$ なる最大の h を $h_{\max}$ とすると，$T(p(h_{\max}))$ が構築された Prefix-Closed B-tree である． □

手続き *EditAux*(h, α) $p_{rm}(h)=nil$ ならば，新たに節を作り，この節へのポインタを $p_{rm}(h)$ とする．

$N(p_{rm}(h))$ のキーの個数が C 未満の場合

$N(p_{rm}(h))$ の右端に α を挿入する．

$N(p_{rm}(h))$ のキーの個数が C の場合

新たに節 $N(q)$ を作成する．$h=0$ ならば，$y \llcorner \alpha \vee y \in D(p_{rm}(0))$ なるすべての y を $\delta_1, \cdots, \delta_t$ とすると（ただし，$i<j$ ならば $\delta_i<\delta_j$），$N(q)=[\delta_1 \cdots \delta_t \ \alpha]$ とし，*EditAux*$(1, \alpha q)$ を実行する．$h>0$ ならば，$\alpha=kr$ とすると，$N(q)=[r]$ とし，*EditAux*$(h+1, kq)$ を実行する．最後にいずれの場合も $p_{rm}(h)$ を q とする． □

定理 7 編集アルゴリズムで構築された Prefix-Closed B-tree の木の高さ H_e，および，占有する領域の大きさに関して，以下の関係が成立する．

$$H_e \leq 1+\log_{(1+C)} \frac{|D|-1}{C-M_d}, \tag{23}$$

領域の大きさ

$$\leq (C+1)\frac{|D|-1}{C-M_d}+C(H_e+1)-1. \tag{24}$$

証明 深さ0の節（根）は1個以上，深さ $i(0<i<H_e)$ の右端の節は0個以上，深さ $i(0<i<H_e)$ で右端でない節は C 個の検索キーを持つので，深さ $i(0<i\leq H_e)$ の節数は，$(1+C)^{i-1}+1$ 以上である．したがって，葉の総数を n とすると，

$$(1+C)^{H_e-1}+1 \leq n. \tag{25}$$

また，右端の葉以外の葉は $C-M_d$ 個以上，右端の葉は1個以上の固有キーを含むので，

$$(n-1)(C-M_d)+1 \leq |D|. \tag{26}$$

(25) (26)式より (23)式成立．

定理4の証明と同様にして，検索キーの総数は $n-1$ である．したがって，節の総数を Q とすると，

$$(Q-n-H_e)C+1 \leq n-1. \tag{27}$$

(26) (27)式より (24)式が成立する． □

6. あとがき

任意に与えられた文字列におけるすべての最左部分語を一回の検索で検索できる単語辞書データ構造として，Prefix-Closed B-tree を提案した．このデータ構造が，すでに提案されている，最左部分語検索に関して同様の機能を持つ拡張 B-tree に比べ，検索効率の点で優れていること，挿入・削除の点では劣っていることを述べた．したがって，本論文で提案した Prefix-Closed B-tree は，挿入・削除を頻繁に行う必要がなく，ある程度挿入・削除の要求を保留しておき，一度期に変更しても支障のないような辞書に向くデータ構造である（この変更は，挿入・削除操作により行うのではなく，編集操作によって単語辞書を再構築することにより行う）．たとえば，共通の単語辞書と利用者登録の個別の辞書とが分離されているような辞書システムでは，共通の単語辞書のデータ構造として Prefix-Closed B-tree が適している．一方，挿入・削除を頻繁に行わなければならないような辞書システムには，拡張 B-tree が適している．

われわれは，日本語の一般用語約 11 万語（約 20 万見出し）に対して，Prefix-Closed B-tree の形式の単語機械辞書システムを作成しており，現在当研究室で稼働中である．

参考文献

1) Knuth, D.: *The Art of Computer Programming*, Vol. 3, Addison-Wesley Publishing Company (1973).
2) 日高，稲永，吉田：拡張 B-tree と日本語単語辞書への応用，信学論 (D), Vol. J67-D, No. 4, pp. 399-405 (1984).
3) 鐘，中村，日高：Prefix-Closed B-tree, 情報処理学会研究会報告, NL 73-15, pp. 115-121 (1989).

付録

補題3の証明を示す．$T_0(p)$ の高さ H に関する帰納法で証明する．

ⅰ) basic step ($H=0$ の場合) : $\alpha=\epsilon$ の場合．明らかに，$T(p)$ は Prefix-Closed B-tree であり，$D_2(p)=D_0(p)\cup\{k\}$．$\alpha=xq$，すなわち分割が起きた場合．明らかに $T_1(p)$ は定義 3 a) b) c) e) を満足し，$N_1(p)$ のキーの数は $C+1$ であり，$T_1(p)$ の高さは $T_0(p)$ と等しい．また，明らかに，$D_1(p)=D_0(p)\cup\{k\}$ である．したがって，補題 2 より，$T_2(p)$, $T_2(q)$ は $T_0(p)$ と同じ高さの Prefix-Closed B-tree であり，(15) (16)式が成立する．

ⅱ) inductive step : 高さ H の Prefix-Closed B-tree に関して本補題が成立すると仮定する．$T_0(p)$ が高さ $H+1$ の Prefix-Closed B-tree であり，$N_0(p)$ が

$[p_0\ x_1\ \cdots\ x_i\ p_i\ x_{i+1}\ \cdots\ x_m\ p_m]$

であるとする．*InsertAux*(p, k) が実行され，Step. 1 が終了した時点を考える．$N_1(p)$ を

$[p_0\ x_0'\ p_0'\ x_1\ \cdots\ x_i\ p_i\ x_i'\ p_i'\ x_{i+1}\ \cdots\ x_m\ p_m\ x_m'\ p_m']$

とする．$x_i'\ p_i'$ は *InsertAux*(p_i, k) が返す値であり，*InsertAux*(p_i, k) が実行されなかったか，実行されても返される値が ϵ ならば，$x_i'\ p_i'$ は空列である．

明らかに，$T_1(p)$ は定義 3 e) を満たす．帰納法の仮定より，$T_2(p_i)$ $T_2(p_i')$ は高さ H の Prefix-Closed B-tree であるから，$T_1(p)$ は根から葉へのパス長はすべて $H+1$ であり（すなわち定義 3 c) を満たす），$N_1(p)$ 以外のすべての節に関して，定義 3 a) b) d) を満たす．$N_1(p)$ に対して定義 3 b) が成立することを以下のようにして証明する．

1) *InsertAux*(p_i, k) $(0\neq i,\ i\neq m)$ が実行され，ϵ が返された場合．定義 3 b) より，

$$D_0(p_i)=\{y\mid x_i\leq y<x_{i+1}\vee y\llcorner x_i,\ y\in D_0(p)\}$$

である．$x_i\leq k<x_{i+1}\vee k\llcorner x_i$ であるから，

$$D_0(p_i)\cup\{k\}=\{y\mid x_i\leq y<x_{i+1}\vee y\llcorner x_i,\ y\in D_0(p)\cup\{k\}\}. \tag{28}$$

帰納法の仮定より，$D_2(p_i)=D_0(p_i)\cup\{k\}$ であるから，

$$D_2(p_i)=\{y\mid x_i\leq y<x_{i+1}\vee y\llcorner x_i,\ y\in D_0(p)\cup\{k\}\}.$$

2) *InsertAux*(p_i, k) $(0\neq i,\ i\neq m)$ が実行され，$x_i'p_i'$ が返された場合．帰納法の仮定より，

$$D_2(p_i)=\{y\mid y<x_i',\ y\in D_0(p_i)\cup\{k\}\}, \tag{29}$$

$$D_2(p_i')=\{y\mid x_i'\leq y\vee y\llcorner x_i',\ y\in D_0(p_i)\cup\{k\}\} \tag{30}$$

である．1) の場合と同様にして，(28)式が成立する．

$N_0(p_i)$ が葉

$[\delta_1\ \cdots\ \delta_n\ y_{n+1}\ \cdots\ y_m]$

であるならば（ただし，$\delta_1\llcorner\cdots\llcorner\delta_n\llcorner x_i$, $x_i\leq y_{n+1}$），定義 2，定義 3 より，$n\leq M_d$, $M_d<\lfloor C/2\rfloor$ であり，*InsertAux* の定義より，$m=C+1$ であるから，$n<\lfloor m/2\rfloor$ である．したがって，*InsertAux* の定義より x_i' は $\lfloor m/2\rfloor+1$ 番目のキーであるから，$y_{n+1}<x\leq y_m$ が成立し，$x_i\leq y_{n+1}$, $y_m<x_{i+1}$ より，

$$x_i<x_i'<x_{i+1} \tag{31}$$

である．$N_0(p_i)$ が葉でないならば，定理 1 と帰納法の仮定より，(31)式が成立する．(28) (29) (30) (31)式および定義 1 より，

$$D_2(p_i)=\{y\mid x_i\leq y<x_i'\vee y\llcorner x_i,\ y\in D_0(p)\cup\{k\}\},$$

$$D_2(p_i')=\{y\mid x_i'\leq y<x_{i+1}\vee y\llcorner x_i',\ y\in D_0(p)\cup\{k\}\}.$$

3) *InsertAux*(p_i, k) $(0<i<m)$ が実行されなかった場合．定義 3 b) より，

$$D_0(p_i)=\{y\mid x_i\leq y<x_{i+1}\vee y\llcorner x_i,\ y\in D_0(p)\}$$

である．$D_2(p_i)=D_0(p_i)$ であり，$x_i\leq k<x_{i+1}\vee k\llcorner x_i$ が成立しないので，

$$D_2(p_i)=\{y\mid x_i\leq y<x_{i+1}\vee y\llcorner x_i,\ y\in D_0(p)\cup\{k\}\}.$$

4) $p_0\ p_0'$, $p_m\ p_m'$ に関しても，1)～3) と同様にして，定義 3 b) を満たすことが示せる．

$D_1(p)=\bigcup_{i=0}^{i=m}D_2(p_i)\cup\bigcup_{i=0}^{i=m}D_2(p_i')$ であるから（ただし，$p_i'=\epsilon$ ならば $D_2(p_i')=\emptyset$），1)～4) より，

$$D_1(p)=D_0(p)\cup\{k\} \tag{32}$$

である．1)～4) および(32)式より，$N_1(p)$ に対して定義 3 b) が成立する．また，2) の場合と同様にして，$x_0'<x_1$, $x_m<x_m'$ であることが示せる．これと(31)式より，根 $N(p)$ に対して定義 3 a) が成立する．したがって，$T_1(p)$ はその根に対する定義 3 d) の性質以外，Prefix-Closed B-tree のすべての性質を満足する．

InsertAux(p, k) が ϵ を返す場合．この場合，$T_2(p)=T_1(p)$ であり，$N_1(p)$ に含まれるキーの個数は C 以下である．したがって，$T_2(p)$ は $T_0(p)$ と同じ高さの Prefix-Closed B-tree である．さらに，(32)式より，

$$D_2(p)=D_0(p)\cup\{k\}$$

である．

InsertAux(p, k) が $x\ q$ を返す場合，すなわち，*Split*(p) が実行される場合．$N_1(p)$ に含まれるキーの

個数は $C+1$ 以上 $2C+1$ 以下であり（なぜならば，この節に含まれるキーの個数は $2m+1$ 以下であり，$m \leq C$ であるから，$2C+1$ 以下．また，$Split(p)$ が実行されるのであるから，この節に含まれるキーの個数は $C+1$ 以上である．），$T_1(p)$ は $N_1(p)$ に対する定義 3 d）の性質以外，Prefix-Closed B-tree のすべての性質を満足するので，(32)式と補題2より，$T_2(p)$ $T_2(q)$ は $T_1(p)$ と同じ高さ，すなわち $T_0(p)$ と同じ高さの Prefix-Closed B-tree であり，

$D_2(p)=\{y \mid y<x,\ y \in D_0(p) \cup \{k\}\}$,

$D_2(p)=\{y \mid x \leq y \vee y \frown x,\ y \in D_0(p) \cup \{k\}\}$

である．また，$N_1(p)$ の根のキーはソートされており，昇順に，

$x_0'\ x_1\ x_1' \cdots x_m\ x_m'$

である．$\lfloor C/2 \rfloor > 0$ であり，分割が起こるためには，$N_1(p)$ のキーの総数は $C+1$ 以上であるから，$N_1(p)$ のキーの総数は3以上．したがって，x は x_0'，x_m' ではない．つまり，$x_1 \leq x \leq x_m$ である．$x_{min}=x_1$，$x_{max}=x_m$ であるから，(17)式が成立する．

（平成5年8月 12 日受付）
（平成6年1月 13 日採録）

冨浦 洋一（正会員）

昭和36年生．昭和59年九州大学工学部電子工学科卒業．昭和61年同大大学院工学研究科電子工学専攻修士課程修了．平成元年同大学院工学研究科電子工学専攻課程単位取得退学．同年九州大学工学部助手，現在に至る．工学博士．自然言語処理，計算言語学，人工知能に関する研究に従事．人工知能学会会員．

中村 貞吾（正会員）

昭和34年生．昭和57年九州大学工学部電子工学科卒業．昭和59年同大大学院工学研究科電子工学専攻修士課程修了．昭和62年同大学院工学研究科電子工学専攻博士課程単位取得退学．同年九州大学工学部助手，平成4年九州工業大学情報工学部講師，現在に至る．工学博士．自然言語処理，計算言語学，人工知能に関する研究に従事．電子情報通信学会，人工知能学会各会員．

日高 達（正会員）

昭和14年生．昭和40年九州大学工学部電子工学科卒業．昭和42年同大大学院工学研究科電子工学専攻修士課程修了．昭和44年同大学院工学研究科電子工学専攻博士課程中退．同年九州大学工学部助手，昭和48年同講師，昭和55年同助教授，昭和63年同教授，現在に至る．工学博士．形式言語の方程式論，自然言語処理，手書き文字認識の研究に従事．電子情報通信学会，人工知能学会各会員．