

P r e f i x - C l o s e d B - t r e e

鐘 征 中村 貞吾 日高 達
Zhong Zheng Teigo NAKAMURA Toru HITAKA

九州大学 工学部
Faculty of Engineering, Kyushu University.

あらまし 任意に与えられた文字列におけるすべての最左部分語が効率よく一時に検索できる大容量の機械辞書向きデータ構造として、Prefix-Closed B-treeを提案し、その性質、操作手順及び能率について考察した。Prefix-Closed B-tree は B-tree 及び拡張 B-tree の長所を兼ね備えて設計したものである。

Abstract A modification of B-tree, Called Prefix-Closed B-tree is proposed. The new data structure can be used to retrieve all the leftmost words of a string in only one searching operation. Prefix-Closed B-tree is designed to combine the advantages of conventional B-tree and Extended B-tree. This data structure can be used in large scale oriental language processing systems.

1. ま え が き

日本語は、単語単位にわかち書きされる欧米語と異なり、べた書きされることから、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 trial and error の特殊な工程を必要とし、単語辞書の検索回数が非常に多くなる傾向がある。又、日本語では単語の正書法が確立していないため、漢字、漢字仮名まじり、かな書きによる単語表記が可能で、単語辞書の見出し語数は実質的な単語数よりもかなり多くなる。従って、日本語の機械処理では単語辞書の検索が処理全体の能率に及ぼす影響が極めて大きく、日本語の特殊性を勘案した能率的な単語機械辞書を作成する必要がある。

日本語文から単語を切り出す処理においては、任意に与えられた文字列の左端に位置する全ての単語（最左部分語）を検索する必要がある。多くの自然言語システムでは、'くるまだいそげ'という文字列の最左部分語を取り出すために、その最左部分列を探索キーとして、複数回検索を行っている。

なお、従来の自然言語処理システムにおいては、辞書を主記憶上に持っている、すなわち、常に in-core の状態で辞書を取り扱っていた。辞書の規模が大きくなった場合には、このような取扱は難しくなる。そこで、辞書を複数個ブロックに分割し、二次記憶上に分散配置して置き、検索要求が生じる毎に検索に必要なブロックを主記憶にデータ転送して検索を行うことになる。ところが、二次記憶から主記憶へのデータ転送は主記憶上の処理と比べて、非常に時間がかかるため、二次記憶からのデータ転送回数を少なくするのが最も重要な課題になっている。

以上の問題に対して、拡張 B-tree と言うものが提案された。拡張 B-tree は、外部記憶向

きの平衡木 B-tree に基づいて拡張したものである⁽⁴⁾。拡張 B-tree の問題点とは、文字列の最左部分語および関連情報（内容）を木の上位レベルにあるため、上位レベルの出せるポインタ数が減り、したがって、木の高さが増える傾向がある。

本稿は、この問題点に目を向けて、B-tree を変形させて、大容量単語辞書向き、かつ最左部分語を一時に検索できるような単語辞書のデータ構造 Prefix-Closed B-tree を提案し、その性質、操作手順及び能率について理論と実験両面から考察した。結論としては、挿入、削除の能率が拡張 B-tree より低下したが、単語の収容能力面では拡張 B-tree より改善した。更新操作 (update) が頻繁に行なわれない大容量辞書に対して、かなりよい構造と思われる。

2. Prefix-Closed B-tree の定義と性質

以下の定義で、 K はキーの全体集合 (key domain) を表す、木のファイルデータキーの集合を D で表す、勿論、 K, D は有限集合とし、 $K \supseteq D$ である。 K, D の要素の数はそれぞれ、 $|K|, |D|$ で表す。

【定義 1】(全順序と順序)⁽⁴⁾

任意の x, y, z ($x, y, z \in K$) に対して

$$x \models y \Rightarrow x \leq y, \quad \dots \dots \dots (1)$$

$$x \leq y \leq z, \text{ 且つ } x \models z \Rightarrow x \models y. \quad \dots \dots \dots (2)$$

日本語単語辞書の場合は、 K は記号列の全体集合であり、 D は単語となる記号列の集合である。キーは有限長の記号列だから、 $\leq$ は辞書式順序であり、 $x \models y$ は記号列 x が記号列 y の最左部分列である順序と考えると、(1) 及び (2) が成立する。

【定義 2】(D の d 次数 M_d)⁽⁴⁾

任意の z ($z \in D$) に対して、 $A(z) = \{y \mid y \in D, \text{ 且つ } y \models z\}$ とし、 D の d 次数というのは、 $M_d \triangleq \max |A(z)|$ 。

【定義 3】(D の u 次数 M_u)

任意の z ($z \in D$) に対して、 $B(z) = \{y \mid y \in D, \text{ 且つ } z \models y\}$ とし、 D の u 次数というのは、 $M_u \triangleq \max |B(z)|$ 。

任意の z ($z \in D$) に対して、 $B(z) = \{y \mid y \in D, \text{ 且つ } z \models y\}$ 、 $y_i \in B$ なら ($i = 1, 2, \dots, M_u$, $y_1 < y_2 < \dots < y_{M_u}$ とする)、すべての y_i は辞書式順序上に、必ず連続している。つまり、辞書式順序は、
 $\dots, z, y_1, y_2, y_3, \dots, y_{M_u}, \dots$ である。これは【定義 1】から、証明することができる。

B-tree と同じように、本論文で提案する Prefix-Closed B-tree の節はキーとポインタから構成される。

$$[p_0, x_1, p_1, x_2, \dots, x_i, p_i, \dots, x_{1m}, p_{1m}]$$

節が葉でない場合は、 x_i はインデックスキー (index key) と言う。節は葉である場合には、ポインタがないため、 p_i は空とする、この場合には、 x_i をファイルデータキー (file data key) と言う。以下の定義では、木の高さを H で表す、 $H > 0$ とする。

【定義 4】(Prefix-Closed B-tree)

空、または、次の性質を持つ木は Prefix-Closed B-tree ($H > 0$) と言う

i) T は以下のような節 (節 P とする) を根とする木である、

$$[p_0, x_1, p_1, x_2, \dots, x_i, p_i, \dots, x_{1m}, p_{1m}] \quad \dots \dots \dots (*)$$

ただし、 p_i は T の部分木を指すポインタである。この部分木は $T(p_i)$ で表す。 x_i はキーであって、 $x_i < x_{i+1}$ とする ($0 < i < |m|$)。 $D(P)$ で、節 P を根とする部分木の葉にあるデータキーの集合を表す。 P は木 T の根である時に、 $D(P) \triangleq D$ 。

ii) 節 P に対して、

a) 部分木 $T(p_i)$ にあるインデックスキー y_{bi} は以下の式を満足する

$$x_i \leq y_{bi} < x_{i+1}, \quad y_{bi} \in K;$$

b) 部分木 $T(p_i)$ の葉にあるデータキーの集合 $D(T(p_i))$ は

$$D(T(p_i)) = \{z \mid z \models x_i, \quad z \in D(P)\}$$

$$\cup \{z \mid x_i \leq z < x_{i+1}, \quad z \in D(P)\} \text{ である.}$$

ただし, $0 \leq i \leq |m|$, $x_0 \triangleq \varepsilon$, $x_{|m|+1} \triangleq \infty$.

iii) 木 T の全ての葉は同一の深さに現われる.

iv) 根以外の節は, $\lceil C/2 \rceil$ 個以上のキーを持つ ($C > 2(M_d + 1)$).

v) 根は (葉であるか, 又は,) 1 個以上のキーを持つ.

vi) 全ての節は C 個以下のキーを持つ.

vii) 節 P の任意の部分木も Prefix-Closed B-tree である.

以上の定義から, $M_d = 0$ の場合には, Prefix-Closed B-tree は B-tree に一致することを容易に証明する.

【定理 1】

Prefix-Closed B-tree ($H > 0$) では, 葉でない節 P にあるポインター p_i が指す部分木 $T(p_i)$ とすれば, 任意の $y \in K$, $x_i \leq y < x_{i+1}$, に対して, 式 $y' \models y$, $y' \in D$ を満足するすべての y' は, $T(p_i)$ のデータキーに含まれる.

(証明) 節 P は根である場合に対して, 背理法での証明を次のように与える.

仮定 $\exists y' \models y$ ($y' \in D$, $y \in K$), 且つ, $x_i \leq y < x_{i+1}$. 且つ, y' は $T(p_i)$ に含まれていない, つまり,

$$y' < x_i \leq y < x_{i+1},$$

$$\text{仮定から } y' \models y,$$

$$\text{だから } y' \models x_i,$$

P は根であるため, $D(P) = D$. 木の定義から,

$$D(T(p_i)) = \{z \mid z \models x_i, \quad z \in D\}$$

$$\cup \{z \mid x_i \leq z < x_{i+1}, \quad z \in D\} \text{ である.}$$

つまり, $y' \in D(T(p_i))$. これは仮定と矛盾する, だから, このような y' は存在しない. すなわち, 根に対して, 定理 1 が成り立つ.

そのほかの場合には, 同じように, 証明することが簡単にできる. ここでは, 省略する. □

定理 1 は次のことを意味する. すなわち, Prefix-Closed B-tree に対する一回の検索操作で, 任意に与えられた文字列 z ($z \in K$) のすべての最左部分語を取り出すことができる.

【定義 5】 (正常キー y_s , 変形キー y_H)

Prefix-Closed B-tree では, 葉以外の任意の節 P に対して, 正常データキー集合 $D_s(T(p_i))$ と変形データキー集合 $D_H(T(p_i))$ を以下のように定義する.

i) 節 P が木の根である場合には,

$$D_s(P) \triangleq D, \quad D_H(P) \triangleq \emptyset;$$

ii) 根以外の節 P には,

$$D_s(T(p_i)) \triangleq \{z \mid x_i \leq z < x_{i+1}, \quad z \in D_s(P)\};$$

$$D_H(T(p_i)) \triangleq \{z \mid z \models x_i, \quad z \in D(P)\}$$

$$\cup \{z \mid z > x_i, \quad z \in D_H(P)\}.$$

ただし, $0 \leq i \leq |m|$, $x_0 \triangleq \varepsilon$, $x_{|m|+1} \triangleq \infty$

$D_s(T(p_i))$ の要素 y_s は $T(p_i)$ の正常キーと呼び, $D_H(T(p_i))$ の要素 y_H は $T(p_i)$ の変形キーと呼ぶ.

これから説明の簡単のために, 任意の木 T に対して, T の最も右にある葉 (rightmost-leaf)

を含む任意の部分木 (T を含む) は T の最右部分木と言い, T の最も左にある葉 (leftmost-leaf) を含む任意の部分木 (T を含む) は T の最左部分木と言う.

【定義 6】 (二つの部分木の隣接)

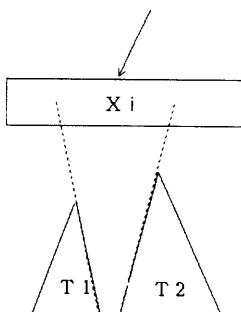
Prefix-Closed B-tree では, 葉でない任意の節 P にあるキー x_{i+1} ($0 \leq i < |m|$) の左右にあるポインター p_i, p_{i+1} の指す部分木 $T(p_i), T(p_{i+1})$ に対して, $T(p_i)$ の任意の最右部分木と, $T(p_{i+1})$ の任意の最左部分木と, お互いに隣接していると言う.

x_{i+1} はこれらのお互いに隣接している木の分離キーという.

明らかに, Prefix-Closed B-tree では, 隣接している任意の二つ部分木には, 分離キーが必ず存在し, そして, 一個しかない.

【定理 2】

Prefix-Closed B-tree ($H > 0$) では, 任意の二つの隣接している部分木 T_1, T_2 (T_1 は T_2 の左にあるとする) に対して, T_2 のすべての変形データキーは, 必ず T_1 のデータキーである.



この定理の証明は, 省略する.

3. Prefix-Closed B-tree の操作アルゴリズム

[編集手順]

単語辞書システムの構築には, ソートされた原データに対して, 編集アルゴリズムを用いて行う. これは B-tree の編集方法とほぼ同様で, 各節の分割キーはこの節の最大のキーであり, キーを挿入されるのは, 各レベルにある最も右にある節に限る. ただし, B-tree と違って, Prefix-Closed B-tree では, 新しい葉を作る時に, 直前の葉から, 分割キーの全ての最左部分語をコピーして, この葉に加える必要がある.

[検索手順]

任意に与えられた変数 z ($z \in K$) に対する検索も, B-tree とほぼ同様である, ただし, 葉のところで, この葉から, 以下の式を満足するすべての z' (最左部分語) を取り出す.

$$z' \preceq z, \quad z' \in D.$$

[挿入手順]

新しい単語の登録は, 挿入手順を用いて行う. 定義から分かるように, Prefix-Closed B-tree のインデックス部分は B-tree であるため, この部分については述べない. 以下のように, 葉のレベルで, z を登録する手順を与える. x_s は current-leaf とその右にある葉との分離キーを表す. 挿入手順:

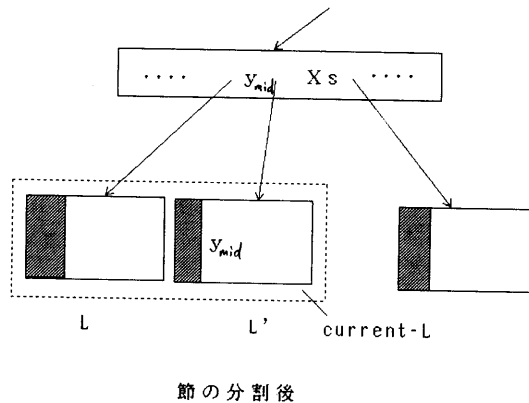
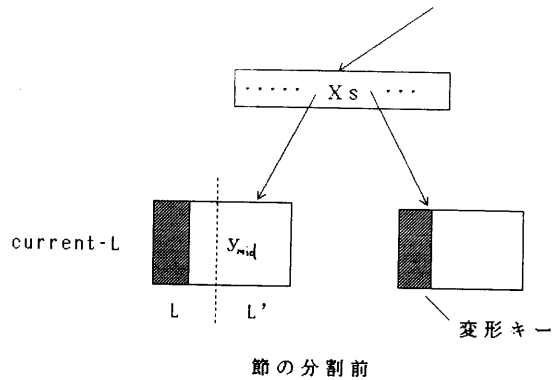
- (1) 検索アルゴリズムで, z を z に対応している葉 L に加える.
- (2) 葉の分割が必要なら, step(5)へ.
- (3) current- L では, $z \not\preceq x_s$ だったら, 終了する.

(4) 右隣接している葉に z を挿入する。この葉を current-L に、step (2) へ。

(5) /* 節の分割, 図を参照 */

- 葉の真中の所から、分割キー y_{mid} を選び、節を二つの部分 L と L' に分ける (L は L' の左にあるとする、 $y_{mid} \in L'$ とする) ;
- L で、式 $y' \neq y_{mid}$ を満足するすべての y' をコピーして、 L' の変形キーとして、 L' に加える ;
- 葉の分割キー y_{mid} をコピーして、上のレベルに挿入する、 L と L' を二つの独立の葉に分かれる。

(6) L' を current-L に、step(3)へ。



削除手順は、大ざっぱに言えば、挿入の逆操作である。ここでは省略する。
挿入、削除アルゴリズムから、以下の定理は成立する。

【定理 3】 Prefix-Closed B-tree の挿入と削除アルゴリズムはこの木の性質 (木の定義を参照) を保存する。

この定理の証明は省略する。

4. Prefix-Closed B-tree に対する操作の能率

- 木の高さ H と占有領域 S

木の高さの上限 $H_{\max}$ (upper bound) 及び占有する領域の大きさ S と見出し語数 $|D|$ の関係は次のように与える.

すべての節がフルの状態 (C 個キーある) である場合には, Best case とする. そして, 根が一つのキーを持って, 根でない全ての節の半分がデータにつまっている場合は Worst case とする.

レベル 0, 1, 2, 3, ... の節の個数は最小で, 1, 2, $2([C/2] + 1)$, $2([C/2] + 1)^2$, $2([C/2] + 1)^3, \dots$; 葉には $|D| / ([C/2] - M_d)$ 個ある. だから,

$$|D| / ([C/2] - M_d) = 2(1 + [C/2])^{(H-1)}$$

$$H_{\max} = 1 + \log_{1 + [C/2]} (|D| / (2([C/2] - M_d)))$$

C) M_d とすれば, 近似で,

$$H < \log_{1+a} |D| / C, \quad a = [C/2].$$

拡張 B-tree では

$$H'_{\max} = 1 + \log_{1+b} |D| / (2(M_d + 1)), \quad b = C / (2(M_d + 1)).$$

この二つの式から, 同じ D 集合に対して, Prefix-Closed B-tree の高さ H は拡張 B-tree より小さくなったことが分かる.

占有領域 S は, 以下のように与える. 木の節の総数 (根を含む) N は, 最大で,

$$N_{\max} = 1 + 2((1 + [C/2])^{H-1} / [C/2])$$

占有領域 $S \leq N_{\max} * C$ は

$$\begin{aligned} S &\leq \{(4/C) \{ (1 + C/2) [|D| / (2(C/2 - M_d))] - 1 \} + 1\} * C \\ &= (C-4) + (2(2+C) |D| / (C-2M_d)) \end{aligned}$$

ii) 編集の能率

編集の能率は, 単語辞書を編集するのに, 主記憶と二次記憶との間のデータ転送の回数で表わす. 各節を二次記憶から読み込む回数 R_{edit} と各節を二次記憶に書き出す回数 W_{edit} は, 次の式で与える. N は木にある節の総数を表す.

$$R_{edit} = 0.$$

$$W_{edit} = N \leq ((C+1) / (C-M_d)) * (|D| / C) + 1.$$

この式の証明は省略する.

iii) 検索の能率

単語のデータは葉のところにしかないため, 文字列の検索に対して, 節を読み込む回数 R_{search} は, 最大, 最小とも木の高さ H 回である. 明らかに, 書き込む回数 $W_{search} = 0$.

iv) 挿入の能率

挿入の時に, 最悪の場合というのは, 根を含む木のすべての関連している節が分割する場合である. このときに, 二次記憶から読み込む回数 $R_{insetion}$ と二次記憶へ書き込む回数 $W_{insetion}$ は, 以下の式で与える. N' は関連する節の個数の最大値である.

$$R_{insetion} = N' \leq 2(1+1/C)*H + (C+1)/(C-M_d) * M_u / C.$$

$$W_{insetion} = 2N' + 1 \leq 4(1+1/C)*H + 2(C+1)/(C-M_d) * M_u / C + 1.$$

この二つの式の証明は省略する.

以上の式から, Prefix-Closed B-tree の挿入の能率は拡張 B-tree の $R_{insetion} \leq H$, $W_{insetion} \leq 2H + 1$ より悪くなったことが分かる.

5. あとがき

我々は, データを使って, 拡張 B-tree と Prefix-Closed B-tree の単語の収容能力を比較した. Best Case で, 木の高さ $H = 1$ の時に, 拡張 B-tree の収容できる語数は 2 万語に足らずに対して, 同じ環境 (ブロックサイズ $C = 250$ 語, $M_d = 20$ 語) で, Prefix-Closed B-tree 十万語位,

H = 2 の時に 150 万語に対して、4 千万語に近いである。単語収容の能力面ではかなり改善されたことが分かる。しかし、この構造の弱点と言うのは、単語の挿入、削除の効率は、最悪の場合には、拡張 B-tree より、かなり悪くなったことである。

本稿で提案した Prefix-Closed B-tree のインデックス部分は B-tree であるため、メモリの使用率を高めるために、すでに提案されている B*-tree に変えることが考えられる。さらに、インデックスキーに対する圧縮を考慮する Prefix B-tree にすることもできる⁽²⁾⁽³⁾。

これからの課題としては、Prefix-Closed B-tree の更新操作の能率を高めることであると考えている。

謝辞 日頃ご協力を頂いた日高研究室の皆様深く感謝の意を表します。本稿での実験は、九州大学大型計算機センター公用データベース日本語単語辞書を使用した。原データの作成者である九州芸術工科大学の稲永紘之先生に感謝します。

References

1. Bayer, R., and McCreight, C. "Organization and maintenance of large ordered indexes," Acta Inf. 1, 3(1972), 173-189.
2. Bayer, R., and Unterauer, K. "Prefix B-tree," ACM Trans. Database Syst. 2, 1 (March 1977), 11-26.
3. Comer, D., "The ubiquitous B-tree", Comput. Surv., 11(2), (1979) 121-137.
4. 日高, 稲永, 吉田: "拡張 B-tree と日本語単語辞書への応用", 電子通信学会論文誌, 1984/4 Vol. J67-D No. 4.
5. Knuth, D. The art of computer programming, Vol. 3: sorting and searching. Addison-Wesley Pub. Co., Reading, Mass., 1973.
6. Leung, C. H. C., "Approximate storage utilisation of B-tree: A simple derivation and generalisations", Info. Proc. Lett., 19(1984) 199-201.
7. McCreight, E., "Pagination of B*-trees with variable-length records," Commun. of ACM 20, 9(Sept. 1977), 670-674.
8. 横山 晶一, 元吉 文男, 井左 原均: 二次記憶上の大規模語彙を用いる自然言語処理システム, 情報処理学会論文誌, Vol. 29, No. 6
9. Ellis, H., Sartaj, S. Fundamentals of Data Structures, Chapter 10. Computer science press, INC. 478-540.
10. 鐘 征: 変形 B-tree と機械辞書への応用, 九州大学修士論文.