論 文

準最適2分探索木の top-down 的構成アルゴリズム*

西川保幸** 吉田雄二*** 福村晃夫****

Abstract

A binary search tree is an important technique for organizing large files, because it is efficient for both random and sequential access of records in a file. In this paper we present some methods for constructing the nearly-optimal binary search tree. Under the realistic assumption that each name (or key) has different access frequencies, we examine techniques for producing binary search tree structures with minimal or nearly-minimal search time.

Our algorithms consist of heuristic top-down methods and Kunth's "Algorithm **K** (dynamic programming method)". Ours require a time proportional to $N^{3/2}$, and storage proportional to N for all names.

Experimental results are given which show the performances of our methods. The nearly-optimal trees can be expected to have an increase of average search length within 0.1% of that of the optimal tree.

1. まえがき

情報検索を容易に実現するデータ構造の一つに2分探索木 (binary search tree, 以下 b.s.t. と略記) がある. これは, 各キーが参照される頻度が一様でないようなとき, 平均検索時間を短縮できる点で, ファイリングにおける有効な手段であると考えられている.

b.s.t. の構成法についてはこれまでに, 最適構成を与えるアルゴリズムとして, (a) 動的計画法にもとづく方法[1),2)], (b) branch and bound 法による方法[2)]があり, また, 準最適構成を与えるアルゴリズムとして, (c) heuristics を用いた方法[2)~4)]が報告されている. N 個のキーの組に対して, (a)については使用メモリおよび計算時間が $O(N^2)$ であるため, 特に使用メモリの点で実用性に欠ける. 一方, (b)については, 使用メモリは $O(N \log_2 N)$ であるが, 計算時間が $O(N^4)$ となるために非現実的である. また, (c)については, 使用メモリおよび計算時間の点ではいずれも実用に足る方法であるが, 近似性の点で改善の余地が残されている.

本論文で示す構成法は, (a)と(c)との方法の中間に位置し, これらの長所をともに活かすように与えられていて, 文献1) の Algorithm **K** および文献2) のアルゴリズム4とを含むように構成されている. このアルゴリズムは, 初めにある基準に従って全体の木の根に対応するキーを定め, 次にそれによって分割されたキーの部分集合の各々に対して同じ手順を行うという形式になっていて b.s.t. が木の根から末端の節点に向かって順に決定されてゆく. アルゴリズムのもつこの特徴をここでは top-down 的と呼んでいる. **3.2** で与えられる具体的アルゴリズムでは, アルゴリズムを再帰的に記述することでこの特徴が表わされている.

このアルゴリズムに要する使用メモリは $O(N)$, 計算時間は $O(N^{3/2})$ である. また, 近似性については, 従来の方法と比較して, 1桁以上精度の高い準最適 b.s.t. が構成される.

本論文では, このアルゴリズムの定式化と, その有

---

* Top-down Algorithms for Constructnig Nearly Optimal Binary Search Trees, by Yasuyuki NISHIKAWA(TOYOTA MOTOR Co. Ltd.), Yuuji YOSHIDA (Computation Center,' Nagoya University) and Teruo FUKUMURA (Faculty of Engineering, Nagoya University)

** トヨタ自動車工業(株)開発企画室
*** 名古屋大学大型計算機センター
**** 名古屋大学工学部情報工学科

効さに関する実験結果について述べる．以下，**2.**ではアルゴリズムの定式化に必要な諸定義を与え，**3.**でアルゴリズムを構成する際基本となる定理，および，3つのアルゴリズムを与える．このうち，**アルゴリズム2,3**は，**アルゴリズム1**の計算時間の短縮化を計ったものであり，$O(N(\log_2 N)^2)$で実現される．**4.**ではこれらのアルゴリズムの使用メモリと計算時間についての評価を与え，最後に**5.**で，最適 b.s.t. との比較についての実験例を示し，これらにもとづいて，各アルゴリズムの比較考察を行う．

2. 諸 定 義

ここでは，次章で与えるアルゴリズムの記述に必要な用語と記法の定義を与える．

定義 1. 集合 $V=\{v_1, v_2, \cdots, v_n\}$ の上で**2分木** (binary tree, 以下 **b.t.** と略記) は，次のように定義される．

(i) ϕ (空) は b.t. である．

(ii) T_L, T_R を b.t. とするとき，$\forall v \in V$ に対し，(T_L, v, T_R) は b.t. である．

(iii) (i)と(ii)によって構成されるものだけが b.t. である．

T に属する V の要素を木の節点と呼ぶ．

定義 2. b.t. $T=(T_L, v, T_R)$ に対し，次のような関数を定義する．

$\mathrm{root}(T)=v$, $\mathrm{left}(T)=T_L$, $\mathrm{right}(T)=T_R$,

$\mathcal{N}(T)=\{v\}\cup\mathcal{N}(T_L)\cup\mathcal{N}(T_R)$, ただし $\mathcal{N}(\phi)=\phi$,

$|T|=$〔T の節点の総数〕.

T に対して，left, right を任意回施してえられる木を T の部分木と呼ぶ．

定義 3. b.t. T の節点 u から v への**パス**とは，T 上で u と v とを結ぶ最短の節点の系列である．パスの長さは，〔パスに属する節点数〕-1 で定義される．特に $u=\mathrm{root}(T)$ のとき，u から v へのパスの長さは，T における v の**レベル**と呼ばれ，$l_T(v)$ で表わされる．

定義 4. 要素の間に全順序関係 ($\prec$ で表わす) が定義されたキーの集合 $\boldsymbol{K}=\{K_1, K_2, \cdots, K_N\}$ に対する b.s.t. とは，次の条件をみたす b.t. T である．

(i) $\mathcal{N}(T)=\boldsymbol{K}$.

(ii) T の空でない任意の部分木 T' について，$\forall K_i \in \mathcal{N}(\mathrm{left}(T'))$, $\forall K_j \in \mathcal{N}(\mathrm{right}(T'))$ に対し，$K_i \prec \mathrm{root}(T') \prec K_j$ である．

以下ではキーの集合 $\{K_m \mid i \leqq m \leqq j\}$ を $\boldsymbol{K}(i, j)$, この集合に対する b.s.t. の集合を $\boldsymbol{T}(i, j)$, その要素を $T(i, j)$ のように表わす．単に $\boldsymbol{K}$ と書いたときは，$\boldsymbol{K}(1, N)$ を表わすものとする．

定義 5. $\boldsymbol{K}$ の各要素に対して，生起確率 $p_1, p_2, \cdots, p_N$ が与えられたとき，$\boldsymbol{K}(i, j)$ に対する b.t. T のコストを，

$$c(T)=\sum_{\theta=i}^{j} p_\theta (l_T(K_\theta)+1)$$

で定義する．

定義 6. $\boldsymbol{K}(i, j)$ に対する最適 b.s.t. のコスト $\boldsymbol{C}(i, j)$ を次式で定義する．

$$C(i, j)=\min_{T \in \boldsymbol{T}(i, j)} c(T)$$

定義 7. $T \in \boldsymbol{T}(i, j)$ かつ，$i \leqq k \leqq j$ なる k について $K_k=\mathrm{root}(T)$ であるとき，$\mathrm{left}(T), \mathrm{right}(T)$ をそれぞれ $T_L(k), T_R(k)$ と表わす．b.s.t. の性質から次式が成りたつ．

$$T_L(k) \in \boldsymbol{T}(i, k-1),$$
$$T_R(k) \in \boldsymbol{T}(k+1, j).$$

キーの組 $\{K_i, K_{i+1}, \cdots, K_j\}$ に対して最適 b.s.t. を構成するとき，あるキーが根に選ばれ易い条件として，次の3つを考える．

(1) 生起確率が大きい，

(2) 左右部分木の重さの差が小さい，

(3) 左右部分木の大きさ (深さ) の差が小さい．

条件(2), (3)のなりたつ度合いを表わすために，$T=(T_L(k), K_k, T_R(k))$ について，次のようなバランス量 σ, τ を定義する．

定義 8. (weight balance σ of T)

$$\omega_L(k)=\sum_{K_\theta \in \mathcal{N}(T_L(k))} p_\theta=\sum_{\theta=i}^{k-1} p_\theta,$$

$$\omega_R(k)=\sum_{K_\theta \in \mathcal{N}(T_R(k))} p_\theta=\sum_{\theta=k+1}^{j} p_\theta,$$

$$\omega=\sum_{K_\theta \in \mathcal{N}(T)} p_\theta=\sum_{\theta=i}^{j} p_\theta,$$

とするとき，

$$\sigma_k=\min\left(\frac{\omega_L(k)+p_k}{\omega+p_k}, \frac{\omega_k(k)+p_k}{\omega+p_R}\right).$$

定義 9. (depth balance τ of T)

$$d_L(k)=\log_2(|T_L(k)|+1)=\log_2(k-i+1),$$
$$d_R(k)=\log_2(|T_R(k)|+1)=\log_2(j-k+1),$$
$$d=\log_2(|T|+1)=\log_2(j-i+2),$$

とするとき，

$$\tau_k=\min\left(\frac{1+d_L(k)}{d}, \frac{1+d_R(k)}{d}\right).$$

条件(1)およびバランス量 σ, τ を用いて，根とし

ての選ばれ易さを表わす heuristic function $\mathcal{H}$ を次のように定義する.

定義 10. $T=(T_L(k), K_k, T_R(k))$ について,

$$\mathcal{H}_1(k)=p_k\times\sigma_k,$$
$$\mathcal{H}_2(k)=p_k\times\sigma_k\times\tau_k.$$

3. 準最適 b.s.t. の構成アルゴリズム

2. で与えた諸定義は, (node weighted) b.s.t. に関する定義である. ここでは, まず初めに, アルゴリズムの構成において必要となる node-leaf weighted b.s.t. (**N-L b.s.t.** と略記) に関する定義を与え, これから, アルゴリズムを構成するための基本となる定理を導く. 次に, この定理にもとづいて, 準最適 b.s.t. を構成するためのアルゴリズムを与える.

3.1 アルゴリズムの基本定理

定義 11. $\boldsymbol{K}$ に対して, p_i を $K_i\in\boldsymbol{K}$ が検索される確率, $\mathcal{E}_i$を, $\mathcal{E}_0=\{K\mid K\prec K_1\}$, $\mathcal{E}_N=\{K\mid K_N\prec K\}$. 一般に $\mathcal{E}_i=\{K\mid K_i\prec K\prec K_{i+1}\}$ $(i=1,2,\cdots,N-1)$ とするとき q_i を $\mathcal{E}_i$ に属するキーが検索される確率とする. $\boldsymbol{K}, \{p_i\}, \{q_i\}$ に対する N-L b.s.t. とは次の条件をみたす b.t. $\mathcal{T}$ である.

(i) $2\cdot N+1$ 個の節点をもち, どの節点も0個または2個の部分木をもつ.

(ii) $\mathcal{T}$ は集合 $\tilde{\boldsymbol{K}}=\{\mathcal{E}_0, \{K_1\}, \mathcal{E}_1, \{K_2\}, \cdots, \{K_N\}, \mathcal{E}_N\}$ に対する b.s.t. である. ここに $\tilde{\boldsymbol{K}}$ の各要素の間の順序関係は, $\tilde{\boldsymbol{K}}$ の2つの要素に属するキーをそれぞれ K, K' とするとき $K\prec K'$ ならば K が属する要素の方が K' の属する要素より小さいとして定める.

$\mathcal{T}$ の中間節点は $\{K_m\}$ に, 末端の節点は $\mathcal{E}_m$ に対応していることは容易に知られる. 以下では $\{K_m\}$ に対応する節点を Internal node と呼び $\mathcal{J}$m で表わす. $\mathcal{E}_m$ に対応する節点は External node と呼ばれる.

定義 12. $\{q_i, p_{i+1}, q_{i+1}, \cdots, p_j, q_j\}$ に対する$\mathcal{T}$のコスト $c(\mathcal{T})$ を次のように定義する.

$$c(\mathcal{T})=\sum_{\eta=i+1}^{j} p_\eta(l_{\mathcal{T}}(\eta)+1)+\sum_{\xi=i}^{j} p_\xi l_{\mathcal{T}}(\xi).$$

定義 13. キーの組 $\{K_i, K_{i+1}, \cdots, K_j\}$ から M 個のキー $\{\mathcal{J}_1, \mathcal{J}_2, \cdots, \mathcal{J}_M\}$, $(\mathcal{J}_m\prec\mathcal{J}_{m+1})$ を選び, これらにもとづいて M 個のキーに対する N-L b.s.t. の internal nodes, external nodes をそれぞれ次のように定める.

internal nodes の集合 $INT=\{\mathcal{J}_m\mid 1\le m\le M\}$,

external nodes の集合 $EXT=\{\mathcal{E}_m\mid 0\le m\le M\}$.

ここに, $\mathcal{J}_0=K_{i-1}, \mathcal{J}_{M+1}=K_{j+1}$ とし, $\mathcal{E}_m$ は $\{K_\theta\mid \mathcal{J}_m\prec K_\theta\prec\mathcal{J}_{m+1}\}$ とする. すなわち, $\mathcal{E}_m$ は空あるいは $\mathcal{J}_m$ と $\mathcal{J}_{m+1}$ にはさまれた1個以上のキーの集合に対応する節点である.

このような対応によるnode weigted b.s.t. T と, N-L b.s.t. $\mathcal{T}$ との間のコストについての関係は, 次のように表わされる.

internal node $\mathcal{J}_\eta$ の生起確率を $P_\eta=p_{\mathcal{J}_\eta}$,

external node $\mathcal{E}_\xi$ の生起確率を $Q_\xi=\sum_{K_\theta\in\mathcal{E}_\xi}p_\theta$,

$\mathcal{T}$ における $\mathcal{J}_\eta$ のレベルを $L_{\mathcal{T}}(\eta)$,

$\mathcal{T}$ における $\mathcal{E}_\eta$ のレベルを $L_{\mathcal{T}}(\xi)$,

とする. 2種類の b.s.t. T と $\mathcal{T}$ とにおいて, internal nodes として選ばれた全てのキーについて,

$$L_{\mathcal{T}}(\eta)=l_T(\mathcal{J}_\eta),\quad {}^\forall\mathcal{J}_\eta, \tag{1}$$

がみたされているならば,

$$\begin{aligned} c(T)&=\sum_{K_\theta\in INT} p_\theta(l_T(\theta)+1)+\sum_{K_\theta\in EXT} p_\theta(l_T(\theta)+1)\\ &=\sum_{\eta=1}^{M} P_\eta(L_{\mathcal{T}}(\eta)+1)+\sum_{\xi=0}^{M} Q_\xi L_{\mathcal{T}}(\xi)\\ &\quad+\sum_{\xi=0}^{M}\sum_{K_\theta\in\mathcal{E}_\xi} p_\theta(l_T(\theta)-L_{\mathcal{T}}(\xi)+1)\\ &=c(\mathcal{T})+\sum_{m=0}^{M} c(T\{\mathcal{E}_m\}). \end{aligned} \tag{2}$$

定　理

INT の各要素を T^* へ移したとき, 根からそれらの要素へのパス上に, EXT の要素が1つも含まれなければ, $\mathcal{T}^*$ と T^* との構成は INT に関して一致する.

証　明

INT が上の条件をみたしているとき,

$$c(T^*)=c(\mathcal{T}^*)+\sum_{m=0}^{M} c(T^*\{\mathcal{E}_m\}), \tag{3}$$

がなりたつ*. ここで, 第2項は, $\mathcal{T}^*$ の構成とは独立に求められ, これは T^* の部分と一致する.

従って, 次式が成立する.

$$L_{\mathcal{T}^*}(\eta)=l_{T^*}(\mathcal{J}_\eta),\quad {}^\forall\mathcal{J}_\eta. \tag{4}$$

Q.E.D.

3.2 アルゴリズム

ここで与えるアルゴリズムはすべて, $\boldsymbol{K}(i,j)$ に対する最適 b.s.t. $T^*(i,j)$ を求める再帰的アルゴリズム $A(i,j)$ の特別な場合として与えられる. **アルゴリズム** $\boldsymbol{A}(i,j)$ は次のように書かれる. なお, このアルゴリズム中で Algorithm **K** とあるのは Knuth の D.P. によるアルゴリズム[1]を表わす.

* 「最適 b.s.t. の部分木は, それに属するキーの集合について, また, 最適 b.s.t. である」[2] ことから明らかに(3)式が成立する.

〔アルゴリズム $A(i,j)$〕

step 1. $n=j-i+1$ とする．$n\leqq f(n)$ ならば step 2, $n>f(n)$ ならば step. 3 .

step 2.

(i) $n=0$ なら，$T^*(i,j)=\phi$，アルゴリズム終了．

(ii) $n=1$ なら，$T^*(i,j)=(\phi,K_i,\phi)$，アルゴリズム終了．

(iii) $n\geqq 2$ なら，Algorithm **K** により $T^*(i,j)$ を求めてアルゴリズム終了．

step 3. $K(i,j)$ より後述のアルゴリズム (**SELECT**, または **CHOICE**) により $f(n)$ 個のキーを選びこれらを Internal nodes (INT) とする．残りのキーを External nodes (EXT) に帰着させる．

step 4. step 3 で求めた INT, EXT の組に対する最適 N-L b.s.t. を Algorithm **K** により求め，このとき最適 b.s.t. の根を K_{k^*} とする．

step 5. $K(i,k^*-1)$, $K(k^*+1,j)$ に対する最適 b.s.t. をアルゴリズム $A(i,k^*-1)$, $A(k^*+1,j)$ により求め，その結果を $T^*(i,k^*-1)$, $T^*(k^*+1,j)$ とする．

step 6.

$$(T^*(i,k^*-1),K_{k^*},T^*(k^*+1,j))$$

を $K(i,j)$ に対する最適 b.s.t. とする．　⊠

このアルゴリズムで $f(n)=M$ (定数) としたときをアルズリズム 1，$f(n)=F(n)$ (F は単調増加関数で $1\leqq F(n)\leqq n$ であるような関数) としたときをアルゴリズム 2，$f(n)=\max(F(n),M_L)$ (step 3 の INT の要素数に下限 M_L を設ける) としたときをアルゴリズム 3 と呼ぶ．さらに，**step 3** で INT の構成に用いるアルゴリズム SELECT, CHOICE はそれぞれ次のように与えられる．(キーの集合 $K(i,j)$ から m 個のキーを選ぶ場合に対するアルゴリズムとして示す．)

〔SELECT〕

step 1. $k=i,i+1,\cdots,j$ に対して $\mathcal{H}(k)$ を求める．

step 2. $\mathcal{H}(k)$ の値の大きいものから m 個のキーを選びそれらの集合を INT とする．　⊠

〔CHOICE〕

step 1. INT$=\phi$, $\mathcal{K}=\{K(i,j)\}$ (以下では $\mathcal{K}$ の要素を $K_1,K_2,\cdots$ のように表わす)．$P=1$ とおく．

step 2. $p=1$, $\tilde{\mathcal{K}}=\phi$ とおく．

step 3. K_t に属するキーのなかから，$\mathcal{H}$ の値を最大にするキーを選びこれを K_t とする．INT←INT$\cup\{K_t\}$ とする．K_t を K_t より小さいキーの集合と K_t より大きいキーの集合とに分ける．(どちらも K_t は含まない)．これら2つの集合のうち，空でないものを $\tilde{\mathcal{K}}$ の要素とする．

step 4. INT の要素数が m ならばアルゴリズム終了．そうでなければ，

(i) $p<P$ ならば，$p\leftarrow p+1$ として step 3 へ．

(ii) $p=P$ ならば，$\mathcal{K}\leftarrow\tilde{\mathcal{K}}$, $P\leftarrow\tilde{\mathcal{K}}$ の要素数として，step 2 へ．　⊠

CHOICE は，**アルゴリズム 1** において，$f(n)=1$ として，かつ，INT の決定に **SELECT** を使用したときのアルゴリズムにほぼ対応する．(**アルゴリズム 1** では，根の決定が depth first に従って行われるが，**CHOICE** ではこれを breadth first に従って行った形になっている．)

上で述べたアルゴリズム 1 は，文献 2) で示した2つの構成法，すなわち，動的計画法 (D.P.) にもとづく最適構成アルゴリズムと，$\mathcal{H}$ を用いて根を決定し top-down 的に準最適 b.s.t. を構成するアルゴリズムとの中間に位置するものである．これは次のような理由による．**アルゴリズム 1** において，$M=N$ とすれば，1回の **step 2** の操作により $T^*(1,N)$ が構成される．これはまさに N 個のキーについて直接 D.P. を適用して最適 b.s.t. を構成する方法 (文献 1) の Algorithm **K** および文献 2) のアルゴリズム 1) に他ならない．一方，$M=1$ とすれば，**step 3** の操作は各集合について $\mathcal{H}$ を最大とするキーを選ぶことであり，このとき，**step 2** および **step 4, 5** の操作は無効となるから，結局，**CHOICE** と同じ要領で直接準最適 b.s.t. を構成する方法 (文献 2) のアルゴリズム 4) に帰着する．

このアルゴリズムでは，M の値をある程度大きくすることによって，最適な根が INT の要素として含まれる可能性を大きくし，かつ，定理の条件がみたされ易くなり，準最適構成における近似性が高められる．

4. 使用メモリと計算時間の評価

ここでは，**3.** で与えられた各アルゴリズムの使用メモリと計算時間についての評価を示す．

〔使用メモリ〕

Algorithm **K** は，M 個のキーについて $O(M^2)$ のメモリを必要とする[1),*]．$M\leqq\sqrt{2N}$ とすれば，各アルゴリズムの使用メモリは，$O(N)$ である．

* 実際には，$(M+1)(M+2)/2$ の大きさの配列を必要とする．この値が N を越えないとするならば，M としては，$M\leqq(-3+\sqrt{8N+1})\div 2$ まで許される．

〔計算時間〕

SELECT と **CHOICE** に要する計算時間は，それぞれ次のようである．各段階で対象となる集合の大きさを $n=j-i+1$ とする．

SELECT は n 個のキーの中から $\mathcal{H}$ の大きい順に M 個のキーを選ぶ方法であるから，その計算時間は $O(n\times M)$ である．

一方，**CHOICE** は，M 個からなる準最適 b.s.t. を構成する方法であり，このとき，準最適 b.s.t. の各レベル l で計算の対象となる節点の数は，

$$n-2^l+1,\ l=0,1,\cdots,\log_2(M+1)-1,$$

程度である．よって，この計算時間は，

$$\sum_{l=0}^{\log_2(M+1)-1}(n-2^l+1)\simeq n\log_2 M,$$

に比例する．すなわち，$O(n\log_2 M)$.

また，Algorithm **K** の計算時間は，M 個のキーについて，$O(M^2)$ である[1]．

アルゴリズム 1～3 の **step 3** と **5** で要する計算時間が上記のようであるとして，全体としての計算時間を各アルゴリズムで **SELECT** と **CHOICE** を用いたそれぞれの場合について検討する．なお，以下の評価は，得られる木の形に極端な偏りがないことを想定して，構成される b.s.t. $T\{1,N\}$ が完全木となる場合について行われている．$T\{1,N\}$ のレベル l で対象となる集合の大きさ n は，次式で表わされる．

$$n=(N-2^l+1)/2^l,\quad l=0,1,\cdots,L. \qquad (5)$$

以下では**アルゴリズム 1** についての計算時間の評価について述べる*.

アルゴリズム 1 の場合，(5)式において，

$$L=\log_2(N+1)-\log_2(M+1)-1 \qquad (6)$$

となり，$n>M$ のとき **step 3～6**，$n\leq M$ のとき **step 2** が計算の対象となる．

(I) **アルゴリズム 1** で **SELECT** を用いた場合

$$\begin{aligned}&\sum_{n>M}(n\times M+M^2)+\sum_{n\leq M}n^2\\&\lesssim\sum_{l=0}^{L}2^l\left(\frac{N-2^{l+1}}{2^l}\times M+M^2\right)+M^2\times 2^{L+1}\\&=M(N+1)(L+1)+(M^2-M)(2^{L+1}-1)\\&\quad+M^2\times 2^{L+1}\\&=M(N+1)\left(\log_2\frac{N+1}{M+1}+\frac{2M-1}{M+1}\right)-M^2+M\\&\simeq M\times N\log_2(N/M) \qquad (7)\end{aligned}$$

$M\simeq\sqrt{N}$ とすれば計算時間は $O(N^{3/2}\log_2 N)$

(II) **アルゴリズム 1** で **CHOICE** を用いた場合

$$\begin{aligned}&\sum_{n>M}(n\log_2 M+M^2)+\sum_{n\leq M}n^2\\&\lesssim\sum_{l=0}^{L}2^l\left(\frac{N-2^l+1}{2^l}\log_2 M+M^2\right)+M^2\times 2^{L+1}\\&\simeq N\{\log_2 M\cdot\log_2(N/M)+2M\} \qquad (8)\end{aligned}$$

$M\simeq\sqrt{N}$ とすれば計算時間は $O(N^{3/2})$

アルゴリズム 2, 3 については以下のようになる．

(III) **アルゴリズム 2** で **SELECT** を用いた場合，計算時間は $O(N^{3/2})$.

(IV) **アルゴリズム 2** で **CHOICE** を用いた場合には $O(N(\log_2 N)^2)$.

(V) **アルゴリズム 3** で **SELECT** を用いた場合，$O(N^{3/2})$.

(VI) **アルゴリズム 3** で **CHOICE** を用いた場合，

$N\gtrsim 10^6$ のとき，$O(N^{4/3})$,

$N\lesssim 10^6$ のとき，$O(N(\log_2 N)^2)$.

5. 実験例および考察

〔実験データ〕

キーの頻度分布 $\phi_1,\phi_2,\cdots,\phi_N$ を，単語の頻度分布に関する Zipf の法則 $\phi_r=N/r$ に従って発生させる．ここに，r は出現頻度の大きさの順位である．実際には，$[1,N]$ で一様に分布する整乱数 N 個を与え，これらを ϕ_r に従って変換して得られる値の組を1データとした．$N=50$, 100 について 100 データずつ，$N=200$, 400, 800 について 50 データずつ生成する．

〔実験方法〕

上記の実験データに対し，3つのアルゴリズムを，次の2つの場合について適用する．

Case 1: $M=\sqrt{N}$, $f(n)=\sqrt{n}$, $M_L=N^{1/3}$.

Case 2: $M=\sqrt{2N}-2$, $f(n)=\sqrt{2n}-2$, $M_L=\sqrt{N/2}$.

最適 b.s.t. との比較は，$N\leq 200$ (最適解が求まる範囲) で，文献 2) のアルゴリズム1について行う．

〔実験結果〕

実験結果を **Table 1** (次頁参照) に示す．ここで，準最適 b.s.t. の最適 b.s.t. に対するコストの劣化は，

$$\Delta C=\frac{c(T)-c(T^*)}{c(T^*)}\times 100\%,$$

として定義される．

計算には，名古屋大学大型計算機センター FACOM 230-60 FORTRAN D を用いた．

〔考 察〕

各アルゴリズムにおいて，**CHOICE** で H_2 を用い

* **アルゴリズム 2, 3** についての評価については文献 5) に述べられている．

Table 1 Comparisons between Nearly-Optimal b.s.t. and the Optimal b.s.t.

INT	SELECT				CHOICE			
H	$\mathcal{H}_1$		$\mathcal{H}_2$		$\mathcal{H}_1$		$\mathcal{H}_2$	
Case	1	2	1	2	1	2	1	2
	0.18±0.60	0.13±0.35	0.23±0.62	0.12±0.32	0.11±0.26	0.09±0.22	0.08±0.23	0.07±0.21
N=50	0.50±0.73	0.46±0.53	0.62±0.88	0.49±0.68	0.38±0.45	0.47±0.45	0.19±0.32	0.20±0.32
	0.41±0.72	0.18±0.38	0.52±0.79	0.26±0.46	0.32±0.43	0.17±0.31	0.17±0.31	0.12±0.27
	0.06±0.18	0.03±0.15	0.07±0.20	0.06±0.16	0.11±0.24	0.10±0.25	0.11±0.33	0.10±0.32
N=100	0.38±0.41	0.33±0.36	0.49±0.46	0.36±0.37	0.35±0.35	0.40±0.33	0.23±0.38	0.23±0.37
	0.28±0.38	0.09±0.23	0.36±0.40	0.14±0.31	0.27±0.32	0.15±0.27	0.21±0.38	0.13±0.35
	0.05±0.14	0.02±0.07	0.07±0.17	0.05±0.14	0.08±0.17	0.05±0.13	0.03±0.08	0.03±0.08
N=200	0.34±0.31	0.30±0.20	0.43±0.33	0.34±0.28	0.29±0.24	0.30±0.20	0.14±0.12	0.14±0.11
	0.21±0.28	0.05±0.12	0.27±0.29	0.11±0.21	0.18±0.23	0.09±0.16	0.10±0.11	0.06±0.10

(注) 各Nについて，ΔC±標準偏差%を表わし，上段がアルゴリズム1，中段がアルゴリズム2，下段がアルゴリズム3に対応する．

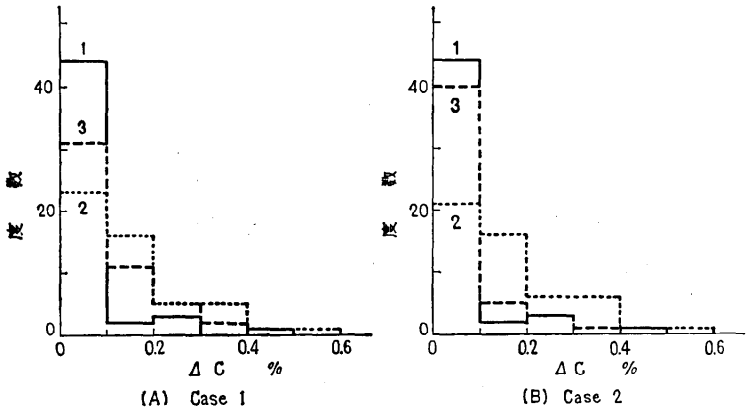

Fig. 1
Performances of Algorithm 1, 2 and 3 (CHOICE/H_2 : N=200)

たときの ΔC の度数分布を **Fig. 1** に示す．Case 1 については (A)，Case 2 については (B) に示されている．これらの結果は，ここに述べた方法が，これまでに報告されている方法[2)-4)]に比べて，最適性が1桁以上改善されていることを示している．

Fig. 2 に **CHOICE** で $\mathcal{H}_2$ を用いたときの計算時間が示されている．**SELECT** で $\mathcal{H}_1$, $\mathcal{H}_2$; **CHOICE** を $\mathcal{H}_1$ 用いた場合についても同じような傾向がみられる．なお，図上で D.P. とあるのは，文献 2) のアルゴリズム1を適用して，最適 b.s.t. を構成するときに要する計算時間である．実験結果からも，**4.** で述べた計算時間について評価が正しいことが実証される．またアルゴリズム1に比べて，極めて計算時間が短縮されたことが知られる．

$\mathcal{H}$としては，近似性および計算時間の点から，**SELECT** においては $\mathcal{H}_1$, とすべきであることが知られる．一方，**CHOICE** においては，計算時間に関しては $\mathcal{H}_1$ の方が有利であるが，近似性の点で $\mathcal{H}_2$ の方が優れている．$\mathcal{H}_1$ と $\mathcal{H}_2$ を用いたときの計算時間につ

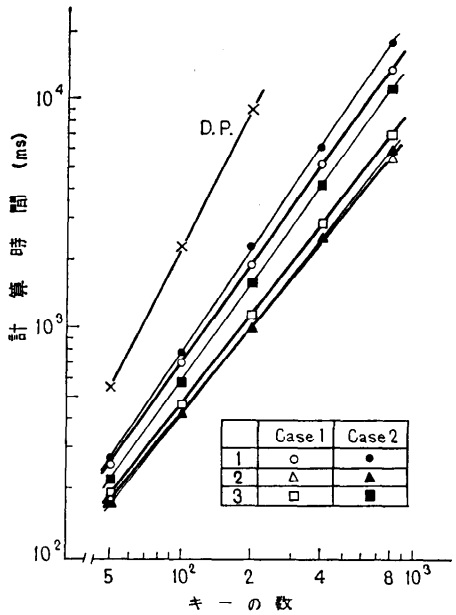

Fig. 2 CPU-time of Algorithm 1, 2 and 3 (CHOICE/H_2)

いての差異が10～20%程度であることを考慮すると，$\mathcal{H}_2$ を用いた方が効果的であると考えられる．

また，INT の決定方法として，**CHOICE** は近似性の点で **SELECT** より優れていることが，実験結果から知られる．**アルゴリズム1** においては，**CHOICE** で $\mathcal{H}_2$ を用いたときの計算時間が，**SELECT** で $\mathcal{H}_1$ を用いたときよりも短縮されることが確かめられている[5]．

最後に，**3.** で述べた定理について検討する．Case 1 と Case 2 との比較，および，文献 5) の実験結果によれば，M を増すにつれ ΔC は小さくなる．これは，このアルゴリズムの近似性に対する保証を示すものであると同時に，**SELECT** および **CHOICE** によって選ばれる INT は，M を増すにつれ，定理の条件をみたし易くなる傾向を有することを示している．また，全ての場合について，定理の条件を満足するように INT を選ぶことは不可能であるが，これを満足しないような若干のキーが INT に含まれているとしても，root(T^*)=root($\mathcal{T}^*$) が多くの場合についてなりたつことが知られている．これらのことから，この定理にもとづく構成法が極めて有効であることが知られる．

6. むすび

本論文では，準最適 b.s.t. の構成法に関して，極めて近似性の高い b.s.t. を構成するアルゴリズムを示し各種のデータに対して実験を行った．

N 個のキーの組に対して，動的計画法にもとづく方法で最適 b.s.t. を構成するアルゴリズム[1),2)]では，計算時間および使用メモリが $O(N^2)$ となるため，大きなNについては実現不可能である．一方，**アルゴリズム1**では，計算時間が $O(N^{3/2})$，使用メモリが $O(N)$ であり，極めて実用的な方法である．

また，さらに計算時間を短縮するために，**アルゴリズム2,3** を示した．これらは，**アルゴリズム1** に比べて近似性の点で若干劣るといえ，充分実用に足る範囲にある．

ここで述べたアルゴリズムが，検索項目の生起頻度の種々の分布に対して同じように有効であるか否かについての実験的検討は残された問題である．なお，一様な生起頻度の場合にはどのアルゴリズムも最適解を与える．また，**アルゴリズム1** で $M=1$ としたとき，頻度分布が $\phi_k=2^{k-1}$ $(1\leq k\leq N)$；$\phi_k=2^{k-3}$ $(3\leq k\leq N)$，$\phi_1=\phi_2=1$；$\phi_k=\phi_{k-1}+\phi_{k-2}$ $(3\leq k\leq N)$，$\phi_1=\phi_2=1$ として与えられるような極端な偏りをもつ場合についても，ほとんどの N について最適解が得られることが実験的に確かめられている．これらのことから，いくつかのバランス量を用いて定義された heuristic function が分布の種々のひずみに適応する能力を十分備えていることが期待される．

謝 辞 熱心に御指導，御討論を賜わる東北大学本多波雄教授に深謝するとともに，日頃有意義な御意見を提供していただく研究室の諸賢に感謝の意を表します．

参考文献

1) D.E. Knuth: The Art of Computer Programming, Vol. 3, pp. 433～439, Addison-Wesley, Reading, Mass. (1973).
2) 西川，吉田，福村: Binary Search Tree のいくつかの構成法について，信学会研資 AL 74-31 (1974).
3) J. Broun and E.G. Coffman: Neary Optimal Binary Search Trees, IFIP Congress 71, pp. 99～103, North-Holland Publishing Company (1972).
4) W.A. Walker and C.C. Gotlieb: A Top-down Algorithm for Constructing Nearly Optimal Lexicographic Trees, Graph Theory and Computing (ed. by R.C. Read), pp. 303～323, Academic Press (1972).
5) 西川，吉田，福村: heuristics および D.P. の特長を活かした準最適 binary search trees の構成法，信学会研資 AL 74-47 (1975).

(昭和50年3月3日受付)
(昭和51年1月19日再受付)