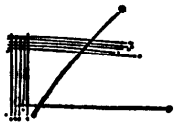
展　望



木構造を用いた見出し探索の技法†

弓場敏嗣††　星　守†††

1. 序　論

1.1 探索モデルと種々の見出し探索法

計算機内に格納される情報は，互いに相異なる情報要素の集合としてデータベースを構成する．この情報要素はレコードと呼ばれ，複数の属性の組からなる定形化された共通の表現形式を持つ．データベース内へのレコードの格納に際して，格納後に予想されるデータベースへの諸操作に対して効率的であるようにレコードの配置は構造化される．データベースに対する操作としては，格納されているレコードを使用しようとするとき必要な探索，既に存在するレコードの削除，さらに新しいレコードの挿入，等が基本的である．

探索を行うにあたってデータベースに対して質問を発する．このとき，データベース中のレコードの部分集合を指定する質問を与え，それを満足するレコード(複数でありうる)が格納されている場所を知り所要の情報をうる．与える質問にはいくつかの定型がある．最も基本的には，質問情報としてデータベース中の各レコードをユニークに参照しうる一つの属性(**主見出し**；primary key) を用いる型と，複数の属性 (**副見出し**；secondary keys) を用いる型に分類される．前者を主見出し探索，後者を副見出し探索と呼ぶ[Knuth 73]．

主見出し探索の質問には次のような種類がある．

(1)　一つの主見出しを与えそれと一致する主見出しを持つレコードを探し出す単純一致型，

(2)　主見出しの値を区間で示しその範囲内に存在するものを探し出す区間指定型[Driscoll 78]，

(3)　主見出しの定義されている空間内のある点を与え，その点に最も近接しているものを探し出す最近接型[Yuval 76]．

なかでも，最も一般的かつ基本的なのは単純一致型で，計算機内での探索の多くはこの型に属する．他の型の場合も単純一致型に依存あるいはそれに類したものに帰着させられることが多い．

副見出し探索における質問型には，

(1)　副見出しとして使用されるすべての属性に対してそれぞれの値が指定されている完全一致型，

(2)　副見出しのすべてではないいくつかの属性に対して値が指定され，無指定の属性の値は何であってもよいとする部分一致型[Rivest 76b]，

(3)　副見出しの各属性に対する値の指定を主見出し探索における区間指定型，最近接型等で与える複合区間指定型，複合最近接型，

(4)　副見出しの各属性を変数とする論理式で探索条件を示す条件指定型，

等がある[Bentley 75c][Dobkin 76][星 79]．副見出し探索での質問型は副見出しとして使用される各々の属性に対して，それを主見出しと見なして主見出し探索の質問型を組合せたものと考えることができ，上述の諸類型の混合型も存在する．

レコード集合が実際に格納される計算機内の物理的データ構造として，次のものが一般的である[Knuth 68]．

(1)　順配置で表現された線形リスト(表)，

(2)　つなぎ配置で表現された線形リスト，

(3)　順配置で表現された木構造，

(4)　つなぎ配置で表現された木構造．

データベースに格納されるレコードが各データ構造内にどのように配置されるかは，探索，挿入，削除等の操作アルゴリズムに依存する．例えば順配置表現の線形リスト（以下，表と呼ぶ）の場合，レコードを表上に任意の順序に並べておき**線形探索** (linear search) を行うこともできるし，主見出し（以下，誤解の恐れのない場合は単に見出しと略記）についてアルファベット順に配置しておき，探索にあたっては**2分探索** (binary search) を用いて効率を上げることもできる．さらに探索効率を高めるには，見出しに乱数を生成する関数を適用する直接計算によって表中のレコードの

† Tree Searching Techniques by Toshitsugu YUBA (Computer Division, Electrotechnical Laboratory) and Mamoru HOSHI (Faculty of Engineering, Chiba University).

†† 電子技術総合研究所　電子計算機部
††† 千葉大学　工学部

表-1.1 種々の主見出し探索法の効率

探索法	データ構造	平均探索路長	所要記憶領域	備考
線形探索法	表（見出しは不整列）	$\frac{1}{2}(N+1)$	rN	アルファベット順出力の計算量は $O(N^2)$.
	つなぎ配置線形リスト	$\frac{1}{2}(N+1)$	$(r+p)N$	レコードの挿入・削除が容易で計算量 $O(1)$.
2分探索法	表（見出しは整列）	$\log_2 N-1$	rN	アルファベット順出力の計算量は $O(N)$, 挿入・削除も $O(N)$.
分散記憶法	表	$\frac{1}{2}\left(1+\frac{1}{1-\alpha}\right)$	$\frac{rN}{\alpha}$	衝突の処理は線形法とする. α は表の占有率.
2分探索木法	順配置木構造	$1.39\log_2 N$	$\frac{rN}{\alpha}$	アルファベット順出力の計算量は $O(N)$, p が不要の代りに空き領域大きい.
	つなぎ配置木構造	$1.39\log_2 N$	$(r+2p)N$	アルファベット順出力の計算量は $O(N)$, 挿入・削除の計算量は $O(1)$.

N: レコード数　r: 1レコードの記憶領域
p: 1ポインタの記憶領域

配置場所（指数番地）をうる分散記憶法（hashing method, scatter storage technique）を用いればよい [Knuth 73][宮川 75b][弓場 80].

つなぎ配置による線形リストの場合は表のように指数番地計算を利用した高速の探索はできないが，レコードの挿入，削除操作がポインタのつなぎかえによって容易にできる長所がある.

表に予め決められた深さの木構造を埋め込み，木上での辿りは指数番地計算によって行う順配置表現の木構造の場合，ポインタ結合で表わされるつなぎ配置表現の木構造に比べて木の形に制約が強く，空き領域の発生による領域の無駄が生じ易い．つなぎ配置表現の木構造ではポインタ用領域が余分に必要であるが，木の形についての制約はなくレコードの挿入，削除に対しても比較的容易にできる．これら木構造を用いる場合は2分探索法と同程度の探索効率をうることができる.

これらのデータ構造上でのいくつかの主見出し探索法についての効率の目安を表-1.1 に示す．ここで，平均探索路長とは，あるレコードを探索するのに要する他のレコードの見出しとの平均比較回数，あるいは探索領域を狭めることに貢献する平均検査回数である.

本稿では木構造を用いた見出し探索法について，その技法の現状を展望する．計算機内に実現される物理的データ構造をポインタ結合によるつなぎ配置表現の木構造とし，各レコードは木構造を構成する節に（必ずしもすべての節に対してではない）配置されるものとする．また，一般にはデータベースは2次記憶を含む2レベル記憶系上に構築されるが，ここではデータ構造の全体が主記憶上に置かれる場合について考えることとする.

1.2 木構造探索の形式化

木構造を用いた各種の見出し探索法を一般的に取扱うことを考える．木構造は図-1.1(a)に示すように節とそれらを結ぶ枝（ポインタで実現される）から構成される．この m 分木の上に配置されたレコードは，各節上に置かれた探索情報（探索に際して目的とするレコードの存在領域を各節ごとに指示する情報）を用いて効率的に探索される．その平均探索路長は2分探索法と同じオーダの $O(\log N)$ である（Nはレコード数，以下同様）．探索情報を用いない場合は木の節をすべて辿って探索せざるをえないから，平均探索路長は線形探索法と同様に $O(N)$ となる.

簡単のためレコードを，それを一意に参照する固定長の主見出しで代表させる．レコードの集合を主見出しの集合として木に配置する．各見出しは図-1.1(b)で示すように見出しの全体が桁に分割されていると考える．例えば見出しが文字列であれば，各文字ごとに列の先頭から第1桁，第2桁，……というように考えることができる．また，見出しの文字列全体を第1桁と見なし，1桁のみからなる見出しと考えることもできる．見出しが数値である場合も同様に2進符号列表現の先頭から適当なビット数ごとに桁に分割して考えることも，全体を1桁からなる2進符号列表現の見出しと見なすことも可能である.

木の各節に置かれる探索情報は図-1.1(c)のような形式を持つ．すなわち，検査桁，検査桁値(域)とポインタの対の集合，それに見出し又は見出しへのポインタの各部分から構成される．検査桁 i（自然数）は探索

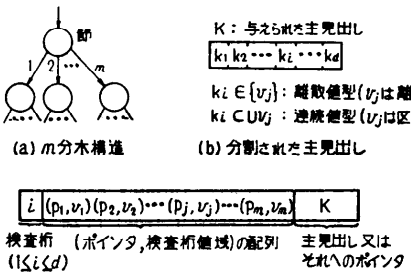


図-1.1 抽象木構造探索モデル

表-1.2 抽象木構造探索モデルの節構成要素の類別

検査桁	ポインタ	検査桁値(域)	分岐演算方式	見出し位置
・見出し全体 ・レベル対応の桁 ・選択的に任意	・$m=2$ (2分岐) ・$m>2$ (m分岐)	・離散値型 ・連続値型	・直接計算方式 ・2重連鎖方式	・すべての節(密モデル) ・端節(疎モデル)

の過程で，その節で検査される分割された見出しの桁が第 i 桁目であることを指示する．**検査桁値域**（又は**検査桁値**）は探索しようとする与えられた見出しKの第 i 桁目の桁値 k_i のとりうる値域を m 分割したものである．分割された m 個の検査桁値域のどれに k_i が属するかによって，木を根から下方向に辿りながら目的とする見出しの探索範囲を各節ごとに $1/m$ づつ狭めていくための辿り先が決定される．このとき，検査桁値域の重複はないものとする．見出しの検査される桁値 k_i が節に置かれた検査桁値域 v_j に含まれるとき，探索はポインタ p_j を選び次の節へと探索は進行する．

見出し又は見出しへのポインタ部分は端節以外には見出しを置かない**疎モデル**の場合は省略される．このときは木の端節に見出し又は見出しへのポインタ部分のみからなる節が配置される．各節に見出し部分が置かれる場合を**密モデル**と呼ぶが，そのとき各節あたりの見出し数は1個である場合と各検査桁値域に対応して m 個存在する場合がある．探索に際して木を根から下方向に辿るとき，密モデルでは与えられた見出しは途中出合う各節上に置かれた見出しとその都度比較され，一致すればその時点で探索は成功裏に終了する．一致しない場合更に下方向への辿りを進めるが，端節に到達しても一致しないとき探索は不成功裏に終了することになる．すなわち，この場合木構造に配置されたデータベース内に探索しようとして与えられた見出し（を持つレコード）は存在しなかったことになる．

検査桁の指定にあたって，それを各節ごとに選択させる場合を**桁選択型**と呼ぶ．ただし，木の根から端節に至る路上に同じ桁が検査桁として指定されることはない．検査桁の選び方として，どの節においても見出し全体を1桁からなる検査桁とする場合や，木の上での節のレベルに対応させその値を検査桁とする場合などがある．後述する2分探索木 (binary search tree) は前者，桁探索木 (digital search tree) は後者に属する例である．

検査桁値域の指定は線形順序集合のある区間として与えられる場合と離散的な値の集合として与えられる場合があり，前者を**連続値型**，後者を**離散値型**と呼ぶ．離散値型では各節で保持する離散値の集合は一般には共通したものとして，論理値，10進数値，アルファベット記号値，等がよく用いられる．各節中の検査桁値域の各々は完全に分離され，一つの見出しに対して探索範囲が二つ以上のポインタを辿る結果にはならないようになっている．

節内での検査桁値域の並べ方に関して，各値域ごとに節内の順配置空間（配列）上で左から右に値の大きい（又は小さい）順に配置する場合と順不同に配置する場合がある．この違いは与えられた見出しのその節で指定された検査桁の値がどの検査桁値域に属するかを決める分岐演算の方式に依存する．すなわち，各節での検査桁値域の集合が線形リストで与えられていると見なして，左から右に線形探索して包含しているか否かを調べる**2重連鎖方式** (doubly-chain；探索のための節間と節内の2種類の鎖の意味) をとる場合は，その線形リスト上にどのような順序で検査桁値域が配置されていてもよい．ただし，線形リスト上での探索効率を考えると全順序で配置されている方が2分探索が用いうる等の利点があるが，各検査桁値域への探索頻度を考慮すると必ずしもそうとはいえない．しかし，与えられた見出しの検査桁の値から直接計算によって該当する検査桁値域に到達し，値域の包含関係を調べる**直接計算方式** (direct calculation) をとる場合は，各節上で各検査桁値域の配置が予め固定されていなくてはならない．

こうした抽象木構造探索モデルを物理的なデータ構造上に実際に構成しようとするとき，時間的・空間的な効率の観点からインプリメンテーション上の修正・変更の必要が生じる．例えば，桁選択型でない場合の検査桁の部分は節の構成要素から省略される．離散値型の検査桁値指定を行うとき，各検査桁値が各節の配列上の指数番地に対応する場合は検査桁値とポインタの対での検査桁値部分は不要となる．また，連続値型でも検査桁値域として指定される区間がその境界値として各節に置かれた見出しによって与えられる場合があり，そのときは見出しの値が兼用される．

上述のように実際の木構造を用いた見出し探索法を抽象木構造探索モデル上に位置づけることができるが，詳細は次章以降に譲る．見出しを複数の桁に分割し各節では検査桁を指定し離散値型の検査桁値を与える見出し探索法を**桁探索木法**と呼び第2章で詳述する，第3章では，見出し全体を一つの桁と見なし連続

値型の検査値域指定を行う探索木法に類別される具体的な見出し探索法の分類と相互関係について述べる．また，木構造を用いた副見出し探索法は，図-1.1(c)で示す節形式の先頭部分にその節で探索情報として用いられる副見出し（属性）の成分を選択的に指定する項を置いたものと考えられる．副見出し探索については第4章で述べる．

1.3 探索コストと最適構成問題

単純一致型の質問に対する見出し探索の効率の基準として表-1.1では平均探索路長と所要記憶領域を用いた．木構造探索を考えるとき，種々の見出し探索法の相互比較あるいは同じ探索法を用いる場合でもその木構造の形や見出しの配置に自由度があるときはそれ等相互の比較がなされねばならない．そうした場合の評価基準としては，探索に要する計算量に相当する評価関数として探索コスト（以下，単にコストと略記）を用いる．探索にあたって木を下方向に辿るとき途中で訪問する平均節数（平均探索路長）や，それを一般化して木上の各見出しに対する探索頻度を重みとして考慮し荷重平均をとったものがコストとしてよく用いられる[Hu 78]．

平均探索路長は探索に際して訪問する各節での検査の平均回数，すなわち，与えられた見出しの検査桁で指定された桁の値がどの検査桁値域に属するかを決める分岐演算の平均検査回数に等しい．このとき，分岐演算が直接計算方式の場合は各節での検査回数は1回であるが，2重連鎖方式の場合は最低1回，最高m回（mは分岐数）必要となる．2重連鎖方式として探索路長を考えるときを特に探索辿路長と呼ぶこととする．

コストの評価関数は先に述べた質問型によって異なる．基本となるコストの考え方としては探索に際して必要な計算量，すなわち，探索の過程で出会う各節上での分岐演算の検査回数の累積値である．区間指定型の質問や副見出し探索の場合は探索路が一般には複数個生じ，したがってコストの計算法も変ってくる[Driscoll 78][星 79]．さらに，探索コスト以外に挿入や削除等の操作に対しても，挿入コスト，削除コスト等のようにコストの考え方を定義できる．本稿ではデータベースに関係する主たるコストとして探索コストのみを考慮することとする．

見出し集合を木構造に写像するにあたってその配置に自由度がある場合，与えられた見出し集合に対してコストを最小にする木の構成を見つける問題，すなわち，最適構成問題が生じる．見出し探索に用いられる種々の木構造の中には見出しの配置や木の形に自由度のないものもあり，その自由度の程度はさまざまである．したがって，最適構成問題を考えるとき，各見出し探索法についてそれぞれの最適構成アルゴリズムの複雑さの程度[Knuth 71, 73][Hu 71][Miyakawa 77][弓場 77]，あるいは最適化の効果[Hoshi 78][弓場 79]はそれぞれ異なっている．

2. 桁探索木法による主見出し探索

2.1 桁探索木法の類別

桁探索木法では，探索に際して与えられた見出しはいくつかの桁に分割される．探索の過程で各節では節ごとに指定された検査桁の値について，検査桁値として与えられた離散値型の基準値との比較が行われる．今まで提案された見出し探索法で桁探索木法として分類される木構造探索法の相互関係を表-2.1に示す．前章で与えた見出し探索のための抽象木構造モデル上での相互の異なりを表示してある．以下では，それら各々について例を示しながら具体的な探索過程，見出しの挿入・削除操作などを説明する．

(1) 密な桁探索木 (sequence tree[Coffman 70][Miyakawa 77], 順序木[弓場 73, 74], digital search tree[Knuth 73])

桁探索木法として分類される探索に用いられる木構造として代表的なものであり，1970年にE. G. Coffman Jr. と J. Eve によって提案された[Coffman 70]．探索に際しては検査桁を木のレベルに対応させ，根（レベル0とする）では与えられた見出しの第1桁を，レベル1では第2桁をというふうに調べられる．したがって，インプリメンテーションでは各節に検査桁を置く部分は必要なく，探索プログラム中に探索過程の段階

表-2.1 桁探索木法の細分類

節構成要素 / 名称	検査桁	分岐演算方式	見出し位置	自由度†	他の呼称
密な桁探索木	木のレベル対応	直接計算方式	密モデル	有	順序木, sequence tree, digital search tree
疎な桁探索木	同上	同上	疎モデル	無	trie, prefix tree
d-c 木	同上	2重連鎖方式	同上	有	doubly-chained tree
patricia 木	スキップ桁数指定	直接計算方式	同上	無	
桁選択型桁探索木	選択的任意	同上	疎・密モデル	有	桁選択順序木, generalized tree

† 木の形および見出しの配置に関する自由度．

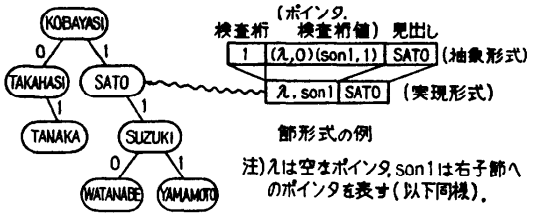


図-2.1 密な2分桁探索木 (アルファベット順挿入)

表-2.2 日本人の姓のうち多いものから7位まで

姓	推定人口(万人)	順位	2進符号列	重み(I)	重み(II)
KOBAYASI	95 75 535	7	1101 1111	9	5 4 31
SATO	190 75	2	1101 0000	22	11 4
SUZUKI	200 0	1	1101 1100	24	11 0
TAKAHASI	80 0	6	0111 0111	9	5 0
TANAKA	130 25	3	0101 1100	15	7 1
WATANABE	85 90	5	1101 0101	10	5 5
YAMAMOTO	90 80	4	1111 1101	11	5 5

(注) 佐久間英著:「日本人の姓」六芸出版 (1972) より作成,姓と姓の間の数字はその上下二つの姓にはさまれた30位までの姓を持つ人口の和を表わす.符号列はアルファベットの奇数番目を1,偶数番目を0(空白も0)で代表させたものである.また重みは推定人口の100分比をとったもので,重み(I)は最頻7位までの姓についてのみ,(II)は間の姓を含めて正規化している.

を示すレベル値が変数としてとられる.検査桁値とポインタの対は予め決められた各節での最大分岐数だけ本来必要である.しかし,この木の場合検査桁値は,分岐演算に際して直接計算によってえられる節の配列上の指数番地(ポインタが置かれる場所)と同じ値等が用いられる.したがって各節の検査桁値部分もインプリメンテーション上は省略される.

図-2.1に密な2分桁探索木の例を示す[宮川 75b].日本人の姓の最頻7位までのローマ字表現を見出しとし,表-2.2のようにそれぞれ分散関数等を用いて適当に2進符号化したものを探索に使用される見出しとして用いる*.例えば,与えられた見出し TANAKA は '01011100' と2進符号列に変換され,図-2.1の木で次のように根から下方向に探索される.まず,根でそこに置かれた見出し KOBAYASI と TANAKA を比べ等しくないので,レベル0での検査桁第1桁を調べ '0' なので左部分木へのポインタ (son) を辿る.

そこで再びその節に置かれた見出し TAKAHASI と比べ等しくないので,レベル1での検査桁第2桁を調べて右部分木に進む.今度はそこに置かれた見出し TANAKA と一致し探索は成功裏に終了する.符号化された見出しは探索に際して辿るべき路を指示していると見なすこともでき,探索路規定型の見出し探索法ということができる[星 75b].規定された探索路上の節で見出し部分が空でありさえすれば,その見出しを置くことが許される自由度がある.この自由度の存在によって後述する最適構成問題が生じる[弓場 71,73,74][杉藤 71][Miyakawa 77].

* 異なる見出しが同じ符号列に変換され衝突が生じても,密モデルの場合は充分長い符号長が用いられると問題は生じない.しかし,疎モデルの場合はすべての符号列は異なるものである必要がある.

新しい見出しの挿入は,符号化した見出しの指示に従って木を辿り空の部分木に到達したときそこに新たな節を創製し見出しを置く.既存の見出しを探索後に木から削除する場合は,削除しようとする見出しのみをそれが置かれた節から消去した後,その節を根とする部分木に属する任意の端節に置かれた見出しで消去された部分を補填すればよい.このとき,補填に用いられた見出しが置かれていた端節は木から切離されねばならない.

(2) 疎な桁探索木 (tree[Fredkin 60], prefix tree [Coffman 70])

密な桁探索木との違いは,見出しは端節にのみ置くとする疎モデルである点である.この木での探索は密な桁探索木と同様に進められるが,見出しの比較は端節においてのみでよい.図-2.2 に図-2.1 と同じ見出し集合に対する疎な桁探索木の例を示す.

挿入は新しい見出しの指示に従って木を辿り,空な端節位置に至るとそこに新たな端節を創製しその見出しを置く.空でない端節位置に到達した場合は,そこに置かれた見出しと新たな見出しの符号列の指示に従って,両者が分離するまで木を下りながら新しい節を

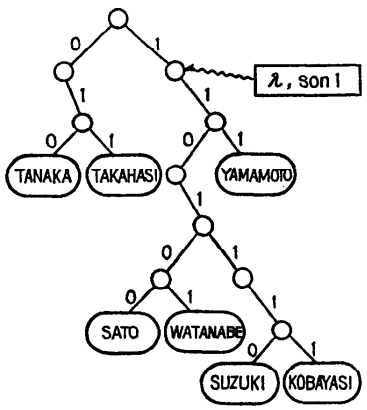


図-2.2 疎な2分桁探索木

創製する操作を繰返す．両者が分離する所で始めて端節を左右子節として創製し，それぞれ対応する見出しを置く．

削除の場合，削除しようとする見出しの置かれた端節を切離せばよい．このとき切離された節の兄弟節が端節でない場合はそのままでよいが，端節の場合はその親節を切離す操作を他の子節を持つ親節に到達するまで繰返す．この繰返しの過程は不可欠のものではないが，削除された節の兄弟節に対してその後に行われる探索の探索路長を短かくする意味を持つ．

この木は構成法から明らかなように見出し集合が与えられるとその形と見出しの配置は符号化された見出しの関数として一意に定まり，したがって最適構成問題は生じない．空間的な効率の悪さを改善するための試みとしては [Maly 76] がある．また，国語辞典等の辞書類の論理的な探索構造は，語彙の文字列の先頭から各文字ごとに桁と考え検査桁値として五十音を持つ疎な桁探索木と見なすことができる．

（3） d-c 木 (doubly-chained tree[Sussenguth 63])

この木の特徴は探索時における分岐演算に際して，2重連鎖方式を用いる点である．疎な桁探索木と同様に疎モデルであり，探索も節間の辿りは桁探索木として進行する．挿入，削除についても疎な桁探索木での操作と同様である．

兄弟節の配置の順序に自由度があり最適構成問題が生じる[Stanfel 69,70,72][Patt 69][Kennedy 72]．その場合探索辿路長によるコスト計算が行われる．木の構成としては疎な桁探索木も d-c 木の一種であり，図-2.2 の木では左右子節へのポインタが検査桁値の 0,1 に対応しているが，その対応関係に自由度を持たせたものが d-c 木であるといえる．

（4） patricia 木[Morrison 68]

直接計算方式の分岐演算を行う疎モデルの桁探索木であるが，検査桁が木のレベルと直接の対応関係を持たない．(1)～(3)の木と違って検査桁を指定する部分は省略できず，この部分には次の節で調べるべき桁と当該節で調べた桁の差の値が置かれる．すなわち，探索の次の段階で調べる検査桁をスキップ値でもって指定する．この機構によって端節を除いて探索領域を狭めることに貢献しない分岐しない節をなくすることができる．端節への平均探索路長が図-2.2 と図-2.3 を比べても明らかなように短縮され，空間的にも効率のよい構成がえられる．

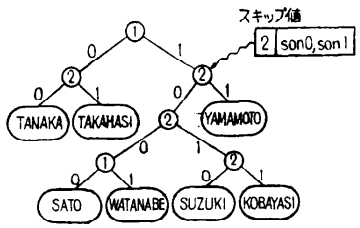


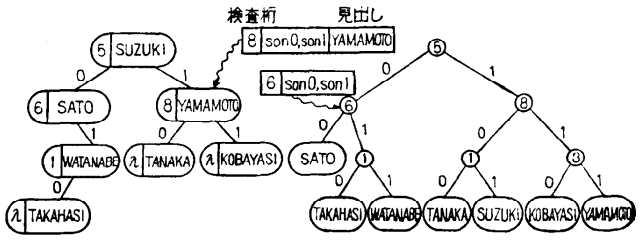
図-2.3 Patricia 木

挿入，削除については疎な桁探索木におけるそれ等のアルゴリズムから容易に類推される．検査桁のとり方から明らかなように，木の形および見出しの配置の自由度はなく最適構成問題は存在しない．

（5） 桁選択型桁探索木（桁選択順序木[弓場 76,77], generalized trie[Burkhard 76])

通常の桁探索木では検査桁が木のレベルに対応していたが，桁選択型のこの木の場合は各部に検査桁の部分を持たせ，探索に際してその節で調べる検査桁を指定する値を任意に与えうる．各節の見出し部分に関しては桁探索木と同様に疎，密両モデルが存在する（図-2.4 にそれぞれの例を示す）．検査桁の指定に自由度があることから密モデルは勿論，疎な桁探索木に対応する疎モデル桁選択型桁探索木の場合も最適構成問題が生じる．

挿入，削除は密モデルの場合は密な桁探索木に，疎モデルの場合は疎な桁探索木に準じる．ただし，挿入のとき検査桁の選択の自由度を生かして，各節での分岐ができるだけ探索領域を狭めることに貢献するような検査桁を選ぶことが必要である．この自由度によって，コストは桁探索木よりも一般に小さいものがえられる．



(a) 密モデル　　(b) 疎モデル

図-2.4 桁選択型桁探索木

2.2 桁探索木法の性質

（1） ランダム木

与えられた見出し集合を，ランダムな順序で挿入した結果構成される木を**ランダム木**と呼ぶ[Hibbard 62][Knuth 73]．先に挙げた五つの木について，疎な桁探索木と patricia 木は木の形，見出しの配置に自由度がなく，見出しの挿入順にそれ等は依存しない．d-c 木については節の兄弟関係の自由度は残るが木の位相的な形は変らない．見出しの挿入順によって，結果的に構成される木の形と見出しの配置が異なる密な桁探索木と桁選択型桁探索木では，ランダム木のコストが確率的に変動する．

これら自由度のある木のランダム木の平均コスト（$N!$ 通りの可能な順序の各々が等確率で生じるとしたときの平均探索路長の平均）は，N個の見出しからなる密な2分桁探索木について$\log_2 N-0.71665$と求まっている[Coffman 70][Knuth 73]．この値は重み一様の場合について求められたものであるが，重みの分布が見出しの挿入順に独立な場合の平均コストとも見なしうる．

桁選択型桁探索木で各検査桁の値を前もって固定してしまう場合も，同じ平均コストを持つランダム木がえられることに注意されたい．

（2） 最適構成問題

与えられた自由度の範囲内でコストを最小にする構成の木を**最適木**という．**最適構成問題**はこの最適木を見つける問題であり，自由度の程度によって最適構成アルゴリズムの複雑さ，計算量が異なる．先にも述べたように疎な桁探索木，patricia 木には最適構成問題は存在しない．

d-c 木は兄弟節間の順序にのみ自由度がある．その最適構成アルゴリズムは，疎な桁探索木から出発して木の底から根にむけて，各節を根とする部分木についてその子節を根とする部分木のコストの大きい順に左から右へ配置するというものである[弓場 75]．底から根にむけてアルゴリズムは進行するので，各節でのその子節を根とする部分木は最適部分木であるという最適木の性質が成立つ．このアルゴリズムの計算量は各節で子節の数だけの検査で充分だから $O(N)$ となる．また，重み一様でかつ見出し数を一定とする場合に，コストを最小とする子節の平均数（平均分岐数）は 3.6 でこのときのコストは $1.24\log_2 N$ であることが導びかれている[Sussenguth 63]．ただし，計算では端節のレベルはすべて等しいものと仮定している．

密な桁探索木の場合は与えられた木を最適構成するアルゴリズムが与えられており[弓場 73, 74][Miyakawa 77]その計算量の上限は2進符号長をLとして $O(LN^3)$，平均的には実験結果から $O(N(\log N)^2)$ によく合致することが知られている[弓場 79]．このアルゴリズムには，木の上に配置された見出しを置換操作によってコストが小さくなるように適宜移動させ，最終的に最適木を構成する方法と，最適木に対して新しい見出しを挿入するときにその結果の木が最適構成となるように置換操作を行う**最適挿入法**がある．

最も自由度の大きい桁選択型桁探索木の場合，疎な2分桁選択型桁探索木の最適構成問題は最適な決定木（decision tree）を構成する問題[Garey 72][Payne 77]とほぼ等価であり，後者については Hyafil 等が NP (Non-deterministic Polynomial) 完全な問題であることを示した[Hyafil 76][Comer 77]．密モデルの場合も同様に NP 完全な問題であることが知られている．

N個 の要素からなる見出し集合に対して $1\sim N-1$ 個の見出しの組合せからなる部分問題に分割し，それぞれを部分最適化した結果をより大きい部分問題に対して統合適用する動的計画法を用いた最適構成アルゴリズム[Payne 77][Ramamoorthy 79b]の計算量は $O(LN2^N)$ となる．この方法は現在の計算機の処理能力からして，$N=20\sim25$ 程度の問題の大きさに対してしか現実性がない[弓場 76]．

以上の桁探索木法における最適構成アルゴリズムの三つの例を比較するとアルゴリズムの複雑さと木の持つ自由度の関係が明らかとなる．すなわち，最も自由度の小さい d-c 木では計算量が $O(N)$，逆に最も自由度の大きい桁選択型桁探索木では $O(LN2^N)$，その中間の密モデル桁探索木は $O(N(\log N)^2)$ である．

最適構成することによってどの程度コストが小さくなるかに関して，最も現実的に有効と思われる密な桁探索木についての実験結果の報告がある[弓場 79]．それによると，各々の見出しに付加される重みの分布則が現実のファイル利用頻度等によく合致するといわれる Zipf 則，80-20 則の場合で，ランダム木に比べて平均的に 50% 程度の改良効果がえられる．

（3） 準最適構成法

与えられた見出し集合に対して木の形と見出しの配置に自由度を持つ木の場合，その木のコストが最適木コストに近い構成の木を**準最適木**という．当然のことながら，準最適木のコストは無作為に構成されるランダム木の平均コストより小さく，かつ準最適木を構成

するアルゴリズムの計算量は最適木のそれより小さくなければならない.

準最適木を構成するとき，桁探索木法の場合前提条件の異なる次の二つの方法がある．一つは，見出し集合の各要素は予め与えられた固定した重みを持つものとして，最適構成アルゴリズムの場合のように見出し集合全体あるいはその部分を配慮しながら，よりコストの小さい木を根から下方向に向って構成するトップダウン構成法である[Walker 72]．他の一つは，木に対して探索を繰返すことによって動的に探索頻度を計数し，それをそれぞれの探索時点での重みとしてその都度必要に応じて木を部分的に再構成する自己組織的構成法である．後者によって構成される木は自己組織木 (self-organizing tree) と呼ばれる[Baer 75][福島 76〜78][Allen 78][弓場 79]。

桁探索木法において準最適構成問題が重要であるのは密な桁探索木と桁選択型桁探索木である．後者では最適構成問題がNP完全であることから，準最適構成アルゴリズムの意義は大きい.

密モデル桁選択型桁探索木の場合，根から下方向に各節に置くべき見出しと検査桁を決めていく．決めていく各段階でその時点での根とその左，右部分木に属する見出しの部分集合が決まり，最終的に準最適木が構成される．置くべき見出しと検査桁を決定するとき，二つの戦術が考えられる．その一つは現在の見出しの部分集合の中から重みの最も重い見出しを選び，検査桁については残る部分集合に対して左右の重みの和が最も平衡するように選ぶ最大重み優先型である．他の一つは，まず各見出しについてそれを当該の根に置いたとき最も平衡する検査桁を求める．次にその検査桁で見出しの部分集合を分割したとしてその結果の左，右部分木に属する見出しの重みの和の大きい方の値が最も小さい見出しを選ぶ平衡優先型である．各々のアルゴリズムの計算量は $O(LN\log N)$, $O(LN^2\log N)$ である[弓場 77].

疎モデルの場合は端節以外には見出しを置かないから，左，右部分木に属する見出しの重みの和が最も平衡するように各検査桁を選べばよく，この計算量は $O(LN\log N)$ である.

密な桁探索木の場合も同様に根から下方向に各節に置くべき見出しを決めていく．各時点での根に置く見出しを決めるにあたっては，その部分木に属する見出しの部分集合から最も重みの重い見出しを選ぶ最大重み優先型と，左，右部分木に属する見出しの重みの和の大きい方の値が最も小さい見出しを選ぶ平衡優先型が考えられる．それぞれの計算量は $O(N\log N)$, $O(N^2\log N)$ で，桁選択型桁探索木のときの桁を各見出しについて調べる計算量分減少していることに注意されたい.

自己組織的準最適構成を必要とするのは，見出し集合の要素が動的に変動する環境や見出しに付加される重みを予め与えられない状況においてである．こうした状況は現実の見出し探索法を適用する局面において頻繁に存在する．最も簡単な自己組織的構成の例として，線形探索における線形リストの見出しの配置に関しての準最適構成アルゴリズムについていくつかの方法とその効果についての報告[Rivest 76a]がある．一般に自己組織的構成アルゴリズムの基本的考え方は，現在探索された見出しは他の見出しに比べて将来とも探索される確率が高いから，探索の開始地点（木の場合は根）に近い位置にその見出しを移動させ，以後の探索路長を短かくするというものである.

密な桁探索木の場合の自己組織的構成アルゴリズムとして，次のアルゴリズムが良い結果をもたらすことが実験的に知られている[弓場 79]．現在探索された見出しは根方向に1レベル移動させる．そのとき，探索回数を計数しておきそれを各見出しの重みとして移動の可否判定，すなわち，現在探索された見出しの重みとそれが置かれている節の親節に置かれている見出しの重みとを比較して前者が大きいときのみ移動するという判定を行う．このアルゴリズムによって木上で根から各端節に至る路上に置かれた見出しは重みの順に並んでいる準最適構成の下降木 (descending tree[Nievergelt 71]) あるいはそれに近い木がえられる．この性質を持つ木は重みの順に見出しを挿入することによっても一般に構成される.

実験によると，こうしてえられた準最適構成の自己組織木のある時点で定まる重みに対するコストは，その時点での最適木のコストに極めて近い値を持つ．また，自己組織的再構成に要する計算量を考慮しても，探索頻度が充分多くなるとそれは無視しうる程度に収束する[弓場 79].

桁選択型桁探索木の場合，各節の検査桁を固定するという前提に立てば密な桁探索木と同様のアルゴリズムで自己組織的構成が可能である．検査桁の値をも変更することを考える場合は，再構成の影響が当該部分木全域に及ぶものと見られるが具体的なアルゴリズムの提案はまだされていない.

3. 探索木法による主見出し探索

3.1 探索木法の類別

探索木法では，探索に際して与えられた見出し全体が，各節で指定された検査桁値域に属するか否かの判定が行われる．検査桁は常に見出し全体と見なされ，抽象木構造探索モデルでの検査桁部分は省略される．各節上の検査桁値域はその節を根とする部分木中に置きうる見出しのとりうる値の範囲を与えるが，実際のインプリメンテーションでは一般にそれら値域の境界値で代用される．表-3.1 に探索木法として分類される木構造探索法の細分類を示す．桁探索木法におけると同様に，以下では個々について例を示しながら具体的な探索過程，見出しの挿入・削除操作などを説明する．

（1） 2分探索木 (binary search tree[Knuth 73], 中順木[星 75b])

木構造を用いた見出し探索法の代表的なものであり，歴史的にも最も古くかつ最も多くの研究がなされている．順配置の線形リスト(表)上での2分探索法をつなぎ配置のデータ構造上で実現したものと考えてよい．各節での分岐数は最大2で，二つの検査桁値域の指定はそれら値域の境界値が用いられ，その境界値として各節に置かれた見出しの値が兼用される．したがって実際の節形式としては二つのポインタと一つの見出しによって構成される．

図-3.1 に表-2.2 の例に対する2分探索木を示す．この木はローマ字表現の見出しをイロハ順(すなわち，WATANABE, …, SUZUKI) に後述するアルゴリズムで挿入し構成した．因みにアルファベット順に挿入すると端節以外のすべての節が右子節のみを持つ線形リスト状の木が構成される．見出し TANAKA が与えられるとそれは次のように探索される．まず，根でそこに置かれた見出し（境界値でもある）WATANABE と与えられた見出し TANAKA を比べる．TANAKAの方が小さいので左部分木へのポインタを辿り，その節に置かれた見出し TAKAHASI と比較する．今度は TANAKA の方が大きいので右部分木に進み，そこに置かれた見出し TANAKA との比較の結果両者は一致するので探索は成功裏に終了する．

表-3.1 探索木法の細分類

名称＼節構成要素	分岐演算方式	見出し位置	自由度†	他の呼称
2分探索木	直接計算方式	密モデル	有	中順木, inorder tree, binary search tree
疎な2分探索木	同 上	疎モデル	有	alphabetic tree
前 順 木	同 上	密モデル	有	preorder tree
B-木	2重連鎖方式	同 上	有	B-tree
B*-木	同 上	疎モデル	有	B*-tree, B*-tree

† 木の形の自由度.

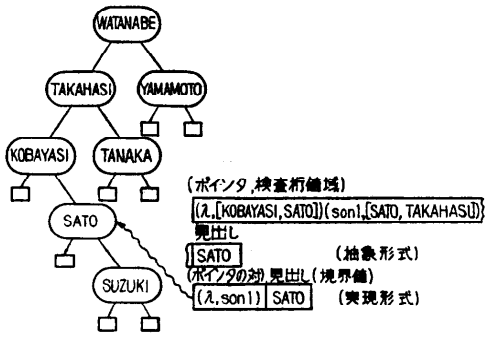


図-3.1 2分探索木（イロハ順挿入，□印は外箱）

新しい見出しの挿入は，根から各節に置かれた見出しとの比較によって左または右の部分木を辿り，空な部分木に到達した時点でそこに新たな節を創製し見出しを置く．図-3.1 では空な部分木の位置を□印(外箱と呼ぶ) で示し，不成功裏に終る探索の終了位置を表わす．表-2.2 の重み(II)の間の重みはこれら外箱に対応させることができる．既存の見出しの削除は，それが置かれている節が端節の場合はその節を切離す．端節でない場合はその見出しを消去後，その節を根とする部分木に属し全順序関係が消去した見出しの直前又は直後の見出し（少なくとも一つは存在する）で補塡する．このとき補塡に用いられた見出しは同様の手続きで削除されねばならない．削除は端節方向に進むのでいずれ操作は終了する．

2分探索木は任意の節においてそこに置かれる見出しは，その節の左(右)部分木中にあるすべての見出しより大きい（小さい）という性質を持ち，この性質を満足する限りどのような配置も可能である．木の形に関する制約はなく，節の総数が見出し数に等しいどのような木の形に対しても上述の性質を保存する見出しの配置は一意に定まる．

（2） 疎な2分探索木 (alphabetic tree[Hu 71])

見出しが端節のみに置かれる疎モデルの2分探索木で，内節には検査桁値域がそれぞれ指定されている．実際のインプリメンテーションでは密モデルにおけると同様に二つの検査桁値域の境界値が各内節に置かれ

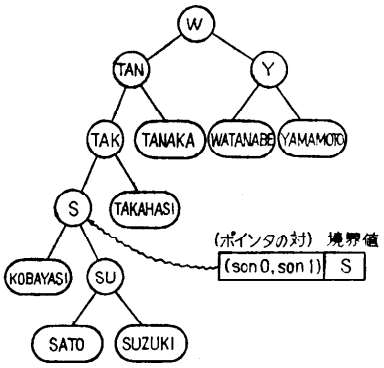


図-3.2 疎な2分探索木（イロハ順挿入）

る．この境界値は必ずしも見出しそのものである必要はない．**図-3.2** に同じく表-2.2 の例に対する疎な2分探索木を示す．この木での探索は密な2分探索木と同様に進められ，端節に至るまでは与えられた見出しと上述の境界値との比較が行われる．

見出しの挿入，削除については疎な桁探索木に類似の操作が必要である．すなわち，挿入では端節位置に到達後，その位置に新しい内節を創製し端節に置かれた見出しと新しい見出しとを分離する適当な境界値を置く．新しい内節の両子節として，元の端節と新しい見出しを置くために創製した端節をつなぐ．削除の場合，切離された端節の兄弟節が端節であるときは疎な桁探索木におけると同様の枝を刈込む操作が必要である．

この木では端節を左から右に辿ると見出しの全順序系列（見出しがアルファベット文字で表現されている場合はアルファベット順の系列）がえられる．どの内節も二つの子節を持ち，見出しの数だけの端節を持つ木であればどのような形の木に対しても見出しの配置は可能でかつ一意に定まる．密モデルと同じく木の形に自由度があるので最適構成問題が存在する．

（3） 前順木[星 75a] (preorder tree[Hoshi 76])

どの部分木をとってみてもその根には部分木に属する見出しの中で最も小さいものが置かれ，さらにその部分木の左部分木中の見出しは右部分木中の見出しより小さいという性質がすべての節において成立している2分木である．2分探索木と同じくインプリメンテーションでは二つの検査桁値域はそれらの境界値（密モデルではその節に置かれた見出し）でもって与えられる．

図-3.3 に図-3.1 と同じく見出しをイロハ順に挿入して構成した前順木の例を示す．この木で与えられた

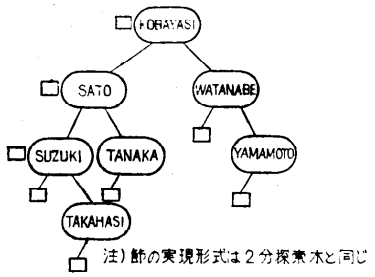


図-3.3 前順木（イロハ順挿入）

見出し TANAKA を探索するとき，まず根の見出し（境界値でもある）KOBAYASI と比較し一致しないから右子節を辿る．右子節に置かれた見出し WATANABE と比べると，TANAKA の方が小さいのでその部分木には属さないことが判明し，左子節の SATO と比べる．TANAKA は SATO より大きいから，SATOの右子節に進みそこで所望のTANAKA を見つけ探索は成功裏に終了する．2分探索木と違って探索路は一直線でなく，兄弟節をジグザグに辿る場合が生じる．したがって，コストを考えるとき探索路長として辿路長を考えなくてはならない．図で8個の外箱は2分探索木の場合と同様に不成功探索の生じる位置を表わし，表-2.2 の間の重みに対応する．

新しい見出しの挿入は，不成功探索の結果位置づけられた外箱の位置に応じて操作が異なる．その外箱が見出しの置かれた節に付随しない場合は，外箱の位置に端節を創製し挿入された見出しを置く．外箱の位置が見出しの置かれた節に付随しかつその節が兄弟節を持つ場合は，まずその節に新たな見出しを格納し，ついでその節を根とする部分木に根を置かれていた見出しを再帰的に挿入する．同じ場合で外箱が付随している節が兄弟節を持たないときはその節を空いている兄弟節の位置につなぎ替え，元の位置に端節を創製してそこに挿入された見出しを置けばよい．削除については2分探索木における削除操作に準じる．

この木では前順序(preorder)で木を辿ると見出しの全順序系列がえられる．木の形に2分探索木と同じ自由度があり，最適構成問題が存在する．そのときコストの計算法が2分探索木の場合と異なる．

（4） B-木 (B-tree[Bayer 72])

2分探索木をm分木 ($m \geq 3$) に拡張したもので，最大分岐数mの場合をm次の B-木と呼ぶ．各節での分岐数は根と端節以外は $\lceil m/2 \rceil \sim m$ であり，根はそれ自身が端節の場合を除いて二つ以上の子節を持ち，さ

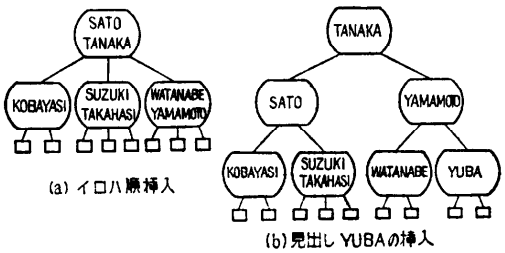


図-3.4 3-2 木(3次のB-木)

らにすべての端節のレベルは同一であるという木の形に関する制限を持つ．検査桁値域は見出しで兼用される境界値で与えられることは2分探索木と同様である．一つの節内に最大 $m-1$ 個の置きうる見出しは，順配置の線形リスト（見出しは整列）によって表現されている．すなわち，分岐演算は2重連鎖方式が一般に用いられる．

探索は2分探索木からの類推で明らかであるが，挿入，削除に関しては木の形に上述の制約があることから特別な技巧を必要とする．図-3.4(a)に示す3次のB-木（3-2木あるいは2-3木と呼ばれる[Knuth 73][Aho 74]）の例を用いて説明する．この木に見出し HOSHI を挿入する場合は不成功探索は最左端の外箱で生じる．それが付随する端節には見出しが一つしか置かれていないのでその節内での整列した位置（左端）に HOSHI は置かれる．見出し YUBA を挿入することを考える．この場合は最右端の外箱で不成功探索が検出されるがそれが付随する端節には既に最大許容数2個の見出しが置かれている．このようにオーバフローが生じると3個の見出しの内，中央に位置する見出し YAMAMOTO を親節に渡し，それ自身の節を分割しそれぞれ残った見出しを置く．YAMAMOTO を受取った節（今の場合，根）でもオーバフローが生じ，ここでもこの節を分割し中央の見出し TANAKA を新しい根に渡しそこに置く（図-3.4(b)）．この結果，木のレベルは1だけ増すことになる．オーバフローは端節から根方向に向って伝播して行くが，根でそれが生じたときのみ木のレベルの増加は生じる．

削除の場合，内節の見出しに対しては2分探索木と同様にその見出しを消去後，全順序関係がその見出しの直前又は直後である見出しでその位置を補塡し，補塡に用いられた見出しは再帰的に削除される．端節に対しては一般には $\lfloor m/2 \rfloor$ 個以上の見出しが置かれているが，当該の見出しの消去によってアンダフローが生じる場合がある．そのときは兄弟節から見出しの移動を行う必要がある．それでもアンダフローが補償できないとき（図-3.4(b)で見出し YUBA を削除の場合等），兄弟節を合体した上で両節の親節に置かれた境界値を与えていた見出しを吸収する．この結果，親節でさらにアンダフローが生じうるが同様の操作を繰返すと例の場合では図-3.4(b)は図-3.4(a)となる．

この木にも上述の制約付きながら木の形に関する自由度がある他，端節に置きうる見出し数に幅があるので木の形が固定されている場合でも見出しの配置に自由度がある．木のレベルが一定であることから最悪の探索路長がおさえられること，アンダフローの閾値が存在することから空間効率がよいこと，節の最大量を固定長（例えば磁気ディスクの1トラック分など）としうることなどの長所から，2次記憶を用いるファイル管理に使用されるようになっている[Comer 79]．

（5） 疎な B-木 (B*-tree[Bayer 72], B+-tree[Comer 79])

疎モデルの B-木でおり，見出しは端節にのみ置かれる．内節に置かれる境界値は疎な2分探索木と同様に見出しそのものである必要はなく，所要領域の小さい分離値を選ぶことで空間効率を上げることができる（例えば，prefix B-tree[Bayer 77]）．なお，[Knuth 73] では B*-木の定義で疎モデルであることの他に，各節の分岐数に関するアンダフローの閾値を $\lfloor 1/3(2m-1) \rfloor$ としている．この閾値は一般に自由に設定しうるが，空間効率やアンダフローの発生頻度，探索効率などに影響を及ぼす因子である．また，探索，挿入，削除の手続きは B-木に準じる．

この木も B-木と同じくすべての端節に至る路長が等しく，探索時間の上限が保障されていることから実時間システムの見出し探索に好適である．木の形あるいは見出しの配置の自由度は B-木と同様である．

3.2 探索木法の性質

（1） ランダム木

2分探索木のランダム木の平均コストは，見出しの重み w の期待値を $E(w)$ とすると $1.386E(w)\log_2 N$ で与えられることが解析的に求まっている[Hibbard 62][Nievergelt 71]．また，2分探索木のランダム木から先に示したアルゴリズムを用いてある任意の見出しを削除しても，その結果の木はランダム木であることが知られている[Hibbard 62]．前順木の重み一様な場合のランダム木コストは，簡単な解析モデルで計算された漸近値として $1.9\log_2 N$ がえられている[Hoshi 78]．

m次の B-木のランダム木については N 個の見出し

を配置するときの平均節数の上，下限，記憶利用率（空間効率）などがmとNの関数として求められている[Knuth 73][Yao 76]．例えば，3-2木の場合の平均節数は大きいNに対して $0.70N$ より大きく $0.79N$ より小さいこと，またmが大きい場合の空間効率は漸近的に $\ln 2 \fallingdotseq 69\%$ であること，などが知られている．

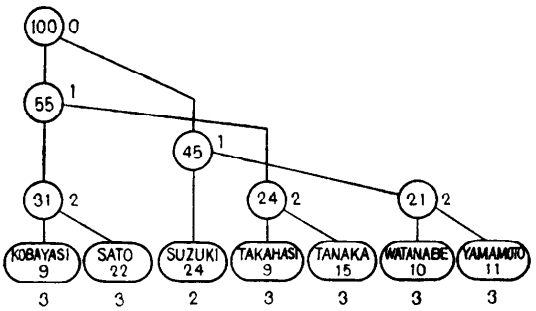
（2） 最適構成問題

探索木法に類別される木では，木の形，見出しの配置に自由度があることからすべての木について最適構成問題が存在する．木によってコストの定義，最適構成アルゴリズムなどの違いがあるので以下に個別にそれらの概要を示す．

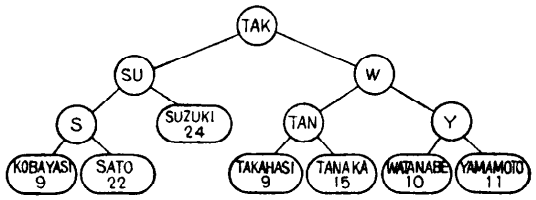
2分探索木（密モデル）では，重み一様の場合は明らかに完全木あるいは完全木の最大レベルの端節が幾つか欠けた形の木が最適木である．重みが一様でない場合，最適木ではその任意の部分木が最適部分木になっているという性質を用い，動的計画法に依って最適構成される．すなわち，全順序付けられた見出しの任意の部分列に対応する部分木が，この見出しの列に対する最適木になっているので，まず見出し数の小さい列に対する最適木を作る．次にその結果を用いてより大きい列に対する最適木を作るという操作を系統的に繰返して最終的に全体の見出し集合に対する最適木をうることができる[Knuth 71]．このアルゴリズムの計算量は $O(N^2)$ の記憶量を用いて $O(N^3)$ であるが，最適部分木の根の選択範囲を狭めうる性質により計算量を $O(N^2)$ としうることが知られている[Knuth 71, 73]．木の最大レベルがLに制限されたときの最適構成アルゴリズムも同じく動的計画法を用いた方法があり，その計算量は $O(LN^2)$ である[Wessner 76]．

疎な2分探索木においても重み一様の場合の最適木は自明である．重み一様でない場合，密モデルと同様に動的計画法を用いて同じ計算量で最適木を構成することができる．木の最大レベルが制限されているときも同様である[Itai 76]．疎な2分探索木の最適構成問題は見出しの全順序関係を保存する最適符号化問題[Gilbelt 59]と等価であり，そうした制約のない場合の最適符号化法として Huffman 法[Huffman 52] がある．Huffman 法を基礎とした計算量 $O(N \log N)$，記憶量 $O(N)$ の最適構成アルゴリズムが存在する[Hu 71, 73]．その概要は次の通りである．

図-3.5(a)に示すように，まず与えられた見出しをそれぞれ重みと共に全順序関係に従って左から右へ並べる．節と節との結合によって新たに創製される節を



(a) レベル値指定（節の枠外の数字はレベル値を示す）



(b) 最適木（図3.2と比較されたい）

図-3.5 疎な2分探索木の最適構成

便宜的に結合節（図の○印）と呼ぶ．結合節を創製するにあたって，二つの節が隣り合っているかあるいはそれらの間にあるのは結合節のみであるような節の対で，それぞれに置かれた見出しの重みの和が最小なもの(複数個あるときは左方優先)から順に結合節を創製していく．この操作を繰返すと最終的に図-3.5(a)で示す様な2分木が構成されるから，この木の上で各節にレベル指定を行う．次に，各端節が指定されたレベル値を持つように，同一レベル値を持ち，隣り合っている節の対でレベル値が大きいものから順次結合していくと（複数個あるときは左方優先）最適木がえられる．図-3.5(b)を図-3.2と比べると約20%コストが減少している．

前順木の最適構成アルゴリズムは，2分探索木の場合と類似した動的計画法を用いた $O(N^2)$ の記憶量で $O(N^3)$ の計算量を要するものが提案されている[Ikeda 75a]．前順木の場合2分探索木とコストのとり方が異なり，重み一様のときの最適木はフィボナッチ木[Knuth 73]となる[Hoshi 78]．

B-木については重み一様な場合に二つのコストの考え方を導入し，それぞれについて最適構成アルゴリズムが提案されている．コストの一つは平均訪問節数で節内での探索時間を無視するものであり，他の一つ

は他の見出しとの平均比較回数で2重連鎖の分岐演算方式における辿路長コストである．どちらの場合も $O(N)$ の計算量で構成される[Rosenberg 78][Miller 79]．重み一様でない場合については未だ解決されていない．一方，B*-木ではすべての見出しは同じ訪問節数を持つので，木の高さを最小にする構成が最適木である．

最適構成にすることの効果について，2分探索木と前順木の場合の改良効果はそれぞれ約 27，33% で，2分桁探索木の 50% 前後に比べると小さいことが報告されている[Hoshi 78]．

3.3 準最適構成法

探索木法に類別される木で準最適構成問題が重要であるのは2分探索木である．B-木では木の形がどの部分木をとっても高さ平衡しており，その意味で常に準最適構成にあるといえる．しかし，見出しの重みを考慮して各節への見出しの再配置を行うなどの準最適化の余地があるが，B-木の準最適構成問題については前順木の場合と同様にまだ未解決の分野と考えられる．

2分探索木の準最適構成法は，桁探索木法におけるトップダウン構成法と自己組織的構成法の他に準最適挿入法がある．これは既にある基準の下で準最適である木に新しい見出しを挿入した結果の木を再び同じ基準の下で準最適に保つよう再構成する方法である．以下では上述の各方法について詳述する．

（1） トップダウン構成法

最適構成アルゴリズムと同じ前提で，より計算量，記憶量の小さいアルゴリズムで最適木コストに近いコストを持つ準最適木が構成できる．最も簡単な方法は木の形は無視して重みの順に見出しを挿入し，準最適木として下降木を構成する方法である．また，重みを無視して左右の部分木に含まれる節数（見出し数）の差が1以内である完全木に近い形の木に構成する方法もある．

少し複雑な発見的方法として，左右の部分木に属する見出しの重みの和を平衡させながら根から下方向に構成していく方法がある．このとき，各部分木の根に置く見出しを決めるにあたって，左右部分木に属する見出しの重みの和が最も平衡するように選ぶ単純平衡型と，左右それぞれの重みの和の大きい方の値が最も小さい見出しを選ぶミニマックス型がある．それぞれのアルゴリズムの計算量は $O(N\log N)$ である．また，得られる準最適木のコストは w_i を各見出しの重みとしてそれぞれ $-\sum_{i=1}^{N} w_i \log w_i + 2$ と $-\sum_{i=1}^{N} w_i \log w_i + 1$ で押えられることが知られている[Melhorn 75] [Unterauer 79]．

その他，単純平衡型として，選ばれた見出しの前後数個の見出しの中で最も重みの大きいものを選ぶという発見的方法を導入すると，最適木の 2～3% 増しのコストの木が構成されることが実験的に確かめられている[Walker 72]．このアルゴリズムも計算量，記憶量はそれぞれ $O(N\log N)$，$O(N)$ であり，最適構成アルゴリズムが $O(N^2)$ の記憶量を用いて $O(N^2)$ の計算量を必要とすることを考えると準最適構成アルゴリズムの有効性が理解できよう．

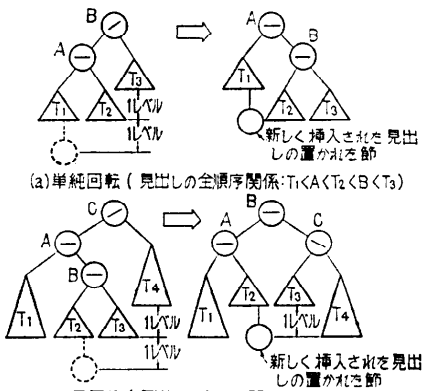
（2） 自己組織的構成法

この方法が用いられる前提は，見出し集合の要素が動的に変動する環境や見出しに付加される重みを予め与えられない状況においてである．2分探索木における自己組織的構成アルゴリズムとしては，桁探索木におけると同様に探索された見出しを根方向に1レベル移動させる方法と，根まで連続して移動させる方法がある．2分探索木の性質を保存する移動操作として，図-3.6 で示す回転操作のうち(a)の単純回転が用いられる（図で節Aの見出しが探索されたとしてAのレベルを根方向に1レベル移動する）．

これらアルゴリズムによって構成される自己組織木は，重み一様とするとそれぞれ $\sqrt{\pi N}$，$1.386\log_2 N$ に収束することが知られている[Allen 78]．なお，桁探索木におけると同様に，探索回数を計数しておきそれを移動の可否判定に用いるアルゴリズムの場合，下降木あるいはそれに近いコストの木に収束する[福島 78a]．

（3） 準最適挿入法

桁探索木（密モデル）においては最適挿入法が存在



注) 節中の一は左右の木の高さが平衡していること，／は右，＼は左部分木の高さがレベル1だけ低いことを示す．

図-3.6 2分探索木における回転操作

し，新しい見出しが最適木に挿入された後にその木を最適に保つことができた．したがって，準最適挿入法を特には必要としなかったが，2分探索木ではこの方法は重要な位置を占める．2分探索木の最適構成法では新しい見出しの挿入に対しては，改めて最初からすべての見出しに対する最適木を構成する必要がある．しかし，予め準最適性の基準を与えそれを保存する木の再構成をともなう新しい見出しの挿入を，$O(\log N)$の計算量と各節に左右部分木の平衡状態などを記憶する若干の領域を付加することによって行うことが可能である．準最適性の基準としては次の様なものがある．

① 高さ平衡木 (height-balanced tree, HB[α] 木)

② 節平衡木 (node-balanced tree, BB[β] 木)

③ 重み平衡木 (weight-balanced tree, WB[γ] 木)

ここで，α, β, γ はそれぞれ平衡条件の閾値を与える．

HB[α] 木は各節での左右部分木の高さの差が α 以内である木[Foster 73]であり，$\alpha=1$ のときは AVL 木[Adel'son-Vel'skii 62]と呼ばれる．AVL 木は2分探索木の平衡木として最も古い歴史を持ち，高さ平衡を保持するための再構成を行う回転操作を初めて用いた．新しく挿入される見出しが図-3.6 左側の木の◌印の位置に置かれるとき，それぞれ単純回転，2重回転を行い右側の木に再構成する．この1回の回転操作によって，結果の木では再び AVL 木の性質が保持されている．なお，節 A, B, C での ──, ／, ＼ 印はそれぞれ左右部分木の高さが等しい，左部分木，右部分木の高さがレベル1だけ大きいことを表わす．実際のインプリメンテーションにおいてはこれら三つの平衡状態を記憶する2ビットの領域が各節に必要となる．高さの不平衡を常に一方の部分木に特定することによって平衡状態を二つにし，所要領域を1ビットにする試みもある[Hirschberg 76][Zweben 78]．

BB[β] 木は各節において，その左部分木の節数+1とその節を根とする部分木の節数+1 との比の値が β と $1-\beta$ の間である木である[Nievergelt 73b]．各節にその節を根とする部分木の節数を記憶させておくことによって，新しい見出しを挿入するとき平衡状態を判定して HB[α] 木と同様に回転操作によって再構成する．HB[α] 木と類似の性質を持つが，両者の関係は一方が一方を包含する関係にはない．

WB[γ] 木は，前の二つが木の高さと節数という木の形に依存する準最適性の基準に基づくものであったのに対し，重みの平衡を考慮に入れる．新しい見出しが挿入される路上の各節において，根から端節方向に重み条件の判定を行う．判定式には挿入路上の各節に置かれた見出しの重みと，その節を根とする部分木に属する見出しの重みの和（各節に記憶しておく）が用いられる．単純回転によってよりよい平衡がえられる状況と2重回転による方がよい状況があり，それぞれに応じた判定式が用いられる[Baer 75, 77]．

AVL 木の高さの上，下限は $1.44\log_2(N+2)-0.328$ と $\log_2(N+1)$ であり[Knuth 73]，BB[β] 木の上限は $\{\log_2(N+1)-1\}/\log_2(1-\beta)^{-1}$ で与えられる[Nievergelt 73b]ことが解析的に示されている．重み一様な場合の各平衡木の比較実験結果がある[Baer 75, 77]．それによるとコストおよび回転操作数については三者とも大きな差はないが HB[α] 木がややよいこと，$\alpha=1\sim5$, $\beta=1-(1/\sqrt{2})\sim0.15$, $\gamma=0\sim4$ と平衡閾値を変化させ条件を緩めても，それによるコストの増分は約3%であり回転操作数の方は半減するが全計算時間はそれ程減らないこと，新たに挿入された見出しがそれ以後数回以上探索されるならば回転操作の計算量を充分補うこと，などが報告されている．

以上の準最適挿入法に対応して準最適削除法もそれぞれ存在し，それぞれの繰返しによって木は準最適な状態に保ちうる．

4. 副見出し探索

4.1 副見出し探索モデル

複数の属性を用いてデータベース中のレコードを探索する副見出し探索法の抽象木構造探索モデルとして，図-1.1(c)で示す節形式の先頭部分にその節で検査する属性の成分を選択的に指定する項を付加したものを考える．この指定する項を検査属性と呼び，主見出し探索における検査桁に対応させることができる．探索に際して各節では，与えられた属性の内一つを用いて探索範囲を狭めるための判定が行われる．このとき，複数の属性値を複合した判定条件を節ごとに置くことも可能である．

主見出し探索におけると同様にコストは探索に要する計算量であり，具体的には探索にあたって木を辿るとき途中で訪問する平均節数である．しかし，主見出し探索での平均探索路長のようにレコードが配置された木が与えられると定まるという訳ではなく，質問型と各質問の出現頻度が与えられて始めてコストは定まる．一つの質問によって複数のレコードが探索され，探索はすべて端節まで進められている．すなわち，探索路は複数存在し，成功，不成功にかかわらず端節に

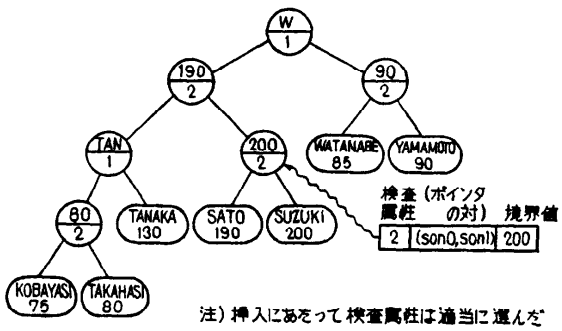


図-4.1 疎な2次元2分探索木（イロハ順挿入）

到達する．したがって，副見出し探索では内節にレコードを置く密モデルは途中でのレコード比較が頻繁となるので効率を悪くする（現実的には疎モデルの木構造を考えればよいが，理論的には密モデルも存在する）．疎モデルでレコードをすべて2次記憶に置くような場合，訪問される平均端節数がコストと考えられる．

4.2 副見出し探索法の類別

副見出し探索における質問型として最も典型的であり，他の質問型の原型をなす部分一致型の質問について考える．探索に際しては一般にレコードを構成する k 個の属性の内数個が指定される．探索の過程である節での検査属性が質問中に指定されていないときは，その節のすべての子節に向けて探索は進行する．質問中に指定されている場合は，指定された属性値について主見出し探索で行ったと同様の比較判定操作によって下方向への辿りを進める．

指定された属性値を主見出しと見なして，先に示した探索法の分類を行いうる．すなわち，指定された属性値を複数の桁に分割し，その節での検査桁の指定に従って離散値型の検査桁値との比較操作を行う桁探索木法，指定された属性値全体を一つの桁と見なし連続値型の検査桁値域指定に基づく比較操作を行う探索木法を考えることができる．以下では副見出し探索法として提案されている具体的な木構造に関して，抽象木構造探索モデル上での位置づけ，および探索過程，レコードの挿入・削除操作などについて述べる．

（1） 属性選択型2値副見出し探索木

副見出しとして用いられる k 個の属性値が $\{0,1\}$ の2値に限定され，質問は無指定（空欄）を表わす＊印を含めて属性数 k 桁の符号列で表現される．例えば，すべての属性に対して肯定，否定，不問の三つの値を持つレコード集合をデータベースとする場合などに用いられる[Knuth 73]．

各節にはその節で検査される検査属性を選択的に指定する値の他は，属性値0,1に対応する分岐先を示すポインタが置かれる．抽象木構造探索モデルの節形式での検査桁は指定された属性の全体が1ビットなので，インプリメンテーション上は不要である．したがって，構成される木は検査属性を検査桁と見なせば桁選択型桁探索木と同じものとなる．異なる点は質問およびレコード自体が属性値(桁値)として＊印を持ちうることである．

探索は根から下方向に指定された検査属性の値を判定しながら進行する．その値が＊印の場合，探索はその節で左，右両子節に分岐する．＊印を含まない新しいレコードの挿入，既存のレコードの削除操作は桁選択型桁探索木におけると同様である．＊印を含む新しいレコードの挿入は，不成功に終った探索の結果位置づけられる端節の内の任意のもの，あるいはコストを考慮してレベル最小の端節に対して挿入操作を行う．

検査属性の指定の自由度およびレコード中に＊印が存在することによる任意性から最適構成問題が存在する．ただし，コストの計算法が主見出し探索におけるものと異なることに注意されたい．

（2） k 次元2分探索木

副見出しとして用いられる k 個の属性のいずれかを検査属性として指定する．抽象木構造探索モデルの節形式で検査桁以下の主見出し探索の場合に対応する部分は，第3章の探索木法におけると同じである．したがって，2分探索木では常に主見出しを唯一つの検査属性としていたが，それを k 次元の属性に拡張したものと見なすことができる．又，疎，密両モデルも存在しうる．

桁探索木では検査桁を木のレベルに対応させたが，この木の場合も検査属性を木のレベルに対応させることができる．属性数が木のレベルより少ない場合は再び最初の属性に戻るという様に巡回的に割当てられる．密モデルの木は k-dimensional tree(略して k-d tree)と呼ばれ，1975年 J. L. Bentley によって提案された[Bentley 75c]．すべての属性値が $\{0,1\}$ を値域とするとき，この木も検査属性を検査桁と見なすことによって密モデル2分桁探索木と考えることができる．

探索は根から下方向に指定された検査属性の値を判定しながら進行する．レコードについては空欄の属性を特別に考慮する必要はないが，部分一致型の質問に

関しては質問の＊印を検査属性として指定する節では探索路が両子節に分岐する．新しいレコードの挿入は疎モデルでは不成功探索が生じた端節の位置に新しい内節を創製し，端節に置かれていたレコードと新しいレコードを分離する属性を検査属性として指定する情報を創製した内節に置けばよい．その内節の両子節として上記の二つのレコードが配置される．密モデルでも不成功探索が生じた節の検査属性部分に新しいレコードを分離する属性を指定して，その子節として新しいレコードを置く端節を創製するとよい．

削除に関しては疎モデルの場合は疎な2分探索木に準じた手順でよい．密モデルでは削除されるレコードの置かれた節の検査属性について，その左部分木内で属性値の最も大きいレコードあるいはその右部分木内で同じく最も小さいものによって，削除されるレコードの位置は補填される．この過程は削除されるレコードが端節に至るまで再帰的に繰返される．

検査属性の指定の自由度の他，2分探索木の有する見出し配置の自由度によって多様性のある最適構成問題が生じる．上述の検査属性がレベルによって固定されている k-d 木の場合も最適構成問題は存在する[Bentley 75c]．

4.3 副見出し探索法の性質

（1） ランダム木

質問型および質問の出現頻度によってコストの計算法は異なるが，ここでは部分一致型の質問で出現頻度一様な場合を考える．属性選択型2値副見出し探索木では，k 個の属性の内 t 個が指定され残りは無指定のときの平均コストは近似的に $N^{\log_2(2-t/k)}$ であることが示されている[Rivest 76b]．またこのような質問に対する平均コストの下限の仮説として $N^{1-t/k}$ なる量がある．

k 次元2分探索木の場合，k-d 木についてのコストとして $O(N^{1-t/k})$ が解析的に求められている[Bentley 75c]．複合区間指定型および t 個の属性についてのみ区間指定された型の場合，最悪のコストはそれぞれ $O(kN^{1-1/k})$，$O(tN^{1-1/k})$ である[Lee 77]．

（2） 最適構成問題

副見出し探索における最適構成問題は，以下の前提条件の違いによって異なったものとなる．

① 質問型と各質問の出現頻度およびレコードの集合が与えられる．

② 各レコードごとの探索頻度（重み）の付加されたレコード集合が与えられる．

③ 質問型と各質問の出現頻度および重みの付加されたレコード集合が与えられる．

すなわち，データベースを使用する環境の特性を与えそれを考慮する場合と，使用された結果としての特性を与えそれに対するコスト最小の木を構成する場合，さらにそれら両者を考慮する場合の三つがある．このとき，レコードの重みとして不成功探索をも考慮すると更に問題は難しくなる．

質問の出現頻度やレコードの重みが一様でない場合の最適構成問題は，主見出し探索からの類推による上述の②の状況に対して考察されているにすぎない．最適決定木の構成アルゴリズム[Payne 77]と同様の，動的計画法による k 次元2分探索木の最適構成アルゴリズムが $O(N^{2k+1})$ の計算量を必要とすることの他，$k=1$ のとき成立した計算量を減らす性質が $k=2$ では成立しないことが知られている[Lueker 79b]．

（3） 準最適構成法

準最適構成問題の前提条件も最適構成問題における①～③の状況が存在する．レコードに付加される重みのみを考慮する②の状況の場合は，主見出し探索における準最適木のトップダウン構成法をそのまま適用することができる．自己組織的構成法については，属性選択型2値副見出し探索木は桁選択型桁探索木と同じ状況にある．ただし，一つの質問に対して探索路が複数存在することから，再構成にあたって移動操作が複雑となる．k 次元2分探索木の場合，2分探索木における回転操作のような木の性質を保存する動的な再構成の方法が知られていない．準最適挿入法に関しても同様のことがいえる．

質問の出現頻度を考慮した発見的方法として，部分一致型質問に対して＊印で指定される頻度が最も小さい属性を検査属性とするというように，根から下方向に検査属性の指定を行っていく方法がある[Alagar 79]．質問の出現頻度が一様なときは，レコードの重みのみを考慮したトップダウン構成法でほとんど常に最適またはそれに近いコストの木を構成できる．一様でないときは上述の発見的構成法によって同様の結果がえられることが実験的に知られている[Alagar 79]．

5. 歴史と幾つかの課題

5.1 歴　史

プログラミング技法として整列 (sorting) の問題が計算機が誕生して間もない頃から関心を集めてきたのに比べて，見出し探索の問題はそれ程注意を払われて

こなかった．その理由は D. E. Knuth によると初期においては主記憶が極く小容量であり，大容量の記憶媒体としては磁気テープしか存在しなかったことに依る[Knuth 73]．そうしたハードウェア環境での見出し探索の問題は意味がない位に簡単であるか，さもないとほとんど不可能なものとなってしまうからである．その後，主記憶の大容量化や磁気ディスクの出現によるハードウェア環境の変化にともなって，見出し探索法の重要性が認知されプログラミング技法の重要な一分野を占めるに至っている．

木構造を用いた見出し探索法は，1950年代に考案された2分探索法に始まり1960年代に入ってその研究は盛んになった．この間の事情は文献 [Knuth 73] に詳しい．以下ではその後の木構造探索に関する研究の足取りを概観する．

新しい見出し探索法の提案については本文中でも触れたように，疎な桁探索木[Fredkin 60]，d-c 木[Sussenguth 63]，patricia 木[Morrison 68]などが1960年代に発表されている．1970年代に入ってからのものとしては，密な桁探索木 sequence tree[Coffman 70] の他，B-木[Bayer 72]，副見出し探索に用いられる k-d 木[Bentley 75c] 等が著名なものといえよう．

1962年のソ連の G. M. Adel'son-Vel'skii と Y. M. Landis による高さ平衡の2分探索木[Adel'son-Vel'skii 62]は準最適構成法の端初をなすものであり，その後の平衡木の研究の基礎をなしている[Foster 65]．この年には2分探索木の平均ランダム木コストも解析的に求められている[Hibbard 62]．桁探索木の平均ランダム木コストは1970年になって始めて計算されている[Coffman 70]．平衡木については，平衡要因に重さを加える等の一般化したモデルについてそのコストの上，下限が1971年に解析的に求められた[Nievergelt 71][Bruno 71]．B-木，3-2木の平均ランダム木コストの解析結果は1978年 A. C. Yao によって与えられた[Yao 78]．

見出し探索に用いられる木の最適構成アルゴリズムの起源は，情報理論における最適符号化法として有名な Huffman アルゴリズム[Huffman 52]に求めることができる．Huffman の符号化法は与えられた見出し（重みを持つ）に2進符号を割当てる場合におけるコスト最小の疎な2分木構成法である．この木を見出し探索に用いることはできないが，同じ最適符号化法でも見出しの木構造上の配置条件として，各見出しは端節にアルファベット順に左から右に並ぶという制約が与えられると疎な2分探索木の最適構成モデルと同じになる．この場合の最適符号化法は1959年に E. N. Gilbert と E. F. Moore によって提案された[Gilbert 59]．これらのアルゴリズムの計算量は各々 $O(N \log N)$, $O(N^3)$ である．後者については約10年後に T. C. Hu と A. C. Tucker によって $O(N \log N)$ の計算量のアルゴリズムが発見された[Hu 71]．同じ年，D. E. Knuth は密な2分探索木の最適構成法について，$O(N^2)$ の計算量の動的計画法によるアルゴリズムを発表した[Knuth 71]．密な桁探索木の最適構成アルゴリズムは，重み一様な場合[杉藤 71][弓場 73, 74]，1977年には重み一様でない場合[Miyakawa 77]がそれぞれ与えられた．B-木の特別な場合である3-2木についても最近 IBM の研究者達によって最適構成アルゴリズムが発表された[Rosenberg78][Miller 79]．

その他の見出し探索と関連した問題として，パタン類別問題[Meisel 73][Garey 77]，決定表プログラミング問題[Reinwald 66, 67]等における事象同定のための決定木の最適構成問題がある．この問題は NP 完全な問題であることが1976，1977年に示された[Hyafil 76][Comer 77]．桁選択型の桁探索木の最適構成問題もこの範ちゅうに含まれるものである[弓場 77]．

5.2 いくつかの課題

木構造を用いた見出し探索法の研究は約20年の歴史を持つがまだ今後に残された課題は多い．本稿を終えるにあたって，その幾つかについて簡単に紹介する．

（1） 新しい見出し探索法

主見出しについての探索法は出尽した観もあるが，連想記憶，磁気バブル等の新しい記憶媒体，マイクロプロセッサ，多重プロセッサ技術等の進歩に依るハードウェア環境の変化にともなう新しい前提条件に基づく探索法の研究は，同じくハードウェアの影響する所の大きい並列探索法の研究と共に今後への期待は大きい．

（2） 種々の質問型に適応した探索法

本稿での見出し探索法は単純一致型の質問形式を前提としたが，第1章で述べたように実際のデータベースへの質問形式は多様である．各質問形式に応じてコストの定義も異なり，種々の最適構成問題が生じると考えられる．

（3） 副見出し探索法

副見出し探索に関して今までにえられた結果は余り豊かなものではない．例えば，k-d 木の平均ランダム木コストの解析結果やその平衡木，自己組織木の構成

法について知られていない．副見出し探索法の現実のデータベースへの適用と合せて，その性質の理論的解明が待たれる．

（4） 2レベル記憶系を用いた探索法

2次記憶を使用する大規模なデータベースを木構造探索法によって構築する方法としてはB-木がある．その他の探索法でも，例えば仮想記憶上に構成することによって理論上は容易に適用しうるが効率の観点からの検討が必要であろう．

（5） Hu-Tucker アルゴリズムの疎な3分探索木への拡張

この問題は文献 [Knuth 73][Hu 78] 等で未解決問題として掲げられているものである．m分探索木の$m=2$の場合にエレガントに解けるものが，$m=3$と次元が一つ上がると非常に難しくなる問題の一例である．

最後に木構造を用いた見出し探索に関する解説，展望論文を紹介する．本文中，幾度も参照した [Knuth 73] の第6章は最も内容が豊かで含蓄に富んだものである．ACM Computing Survey 誌に掲載されたものとして [Nievergelt 74] [Severance 74] の二つがある．前者は平衡2分探索木の解析結果を中心に実際への応用局面への課題等について展望している．後者は著者の提案する探索モデルに沿って見出し探索法の諸問題を扱ったものである．

国内のものとしては [宮川 75b] は分散記憶法を含めて見出し探索全般について平易にその技法を紹介している．また，[星 75b] は著者等の分類法に基づく木構造探索法の類別と関連文献を50件掲載している．

謝辞　電子技術総合研究所・数理基礎研究室の宮川正弘氏には，本稿をまとめる上で有益な御意見をいただいたことを感謝致します．

参考文献

[星 75a] 星　守，弓場敏嗣：前順木における探索と探索コスト最小構成，信学論 (D)，Vol. 58-D, No. 4, 217-219 (1975, 4).

[星 75b] 星　守，弓場敏嗣：木構造と見出し探索—サーベイ，信学技報，EC 75-54, 21-30 (1975, 12).

[星 79a] 星　守，弓場敏嗣：2次元桁選択探索木の準最適構成，信学技報，EC 78-78, 71-81 (1979, 3).

[星 79b] 星　守，弓場敏嗣：最適k次元桁選択探索木の単調性について，昭54・信学全大，＃1491 (1979, 3).

[福島 76] 福島　甫，木村正行：自己組織的な2分探索木 (sobst) について，信学技報，AL 76-48, 83-92 (1976, 10).

[福島 78a] 福島　甫，木村正行：カウントフィールドを持つ自己組織的2分探索木，信学論 (D)，Vol. J 61-D, No. 7, 473-480 (1978, 7).

[福島 78b] 福島　甫，木村正行：ある種の2分探索木間のコストの意味での階層性，信学技報，AL 78-16, 33-40 (1978, 6).

[福島 79] 福島　甫，木村正行：各種の自己組織的2分探索木の期待コストについて，信学技報，AL 78-76, 57-64 (1979, 1).

[牧野 78] 牧野武則，律雲　淳：B-tree における性能評価，昭53・情処全大，＃61-7 (1978, 8).

[宮川 75a] 宮川正弘，弓場敏嗣，杉藤芳雄，星　守：不成功探索を含む最適順序木，昭50・信学全大，＃1116 (1975, 3).

[宮川 75b] 宮川正弘，弓場敏嗣：計算機における見出し探索の技術，電学誌，Vol. 95, No. 8, 11-18 (1975, 8).

[溝口 79] 溝口徹夫：ファイル構成方式の計算量論的評価，昭54・信学全大，＃S 11-5 (1979, 3).

[西川 74] 西川保幸，吉田雄二，福村晃夫：Binary search trees のいくつかの構成法について，信学技報，AL 74-31, 23-34 (1974, 11).

[西川 75] 西川保幸，吉田雄二，福村晃夫：Heuristics および D. P. の特長を活かした準最適 binary search trees の構成法，信学技報，AL 74-47, 87-96 (1975, 1).

[西川 76] 西川保幸，吉田雄二，福村晃夫：準最適2分探索木の top-down 的構成アルゴリズム，情報処理，Vol. 17, No. 8, 736-742 (1976, 8).

[西条 78] 西条経治，丸岡　章，木村正行：部分一致質問に関する最適ファイル構成法，信学技報，AL 78-51, 9-16 (1978, 10).

[杉藤 71] 杉藤芳雄，弓場敏嗣，鳥居宏次：Prefix hash tree を用いた sequence hash tree の最適化の一方法，昭46・信学全大，＃1063 (1971, 3).

[杉藤 73] 杉藤芳雄，星　守，宮川正弘，弓場敏嗣：端節路長和の最小化問題について，信学技報，EC 73-44 (1973, 11).

[弓場 71] 弓場敏嗣，宮川正弘，鳥居宏次：Sequence hash tree の最適化に関する幾つかの問題，昭46・信学全大，＃1062 (1971, 3).

[弓場 73] 弓場敏嗣，宮川正弘：最適な sequence hash tree に関する考察，信学論 (D)，Vol. 56-D, No. 1, 9-16 (1973, 1).

[弓場 74] 弓場敏嗣，宮川正弘，星　守：順序木の最適化問題，信学論 (D)，Vol. 57-D, No. 4, 238-239 (1974, 4).

[弓場 75] 弓場敏嗣，星　守：Doubly-chained tree のコスト最小構成，昭50・信学全大，＃1114 (1975, 3).

[弓場 76] 弓場敏嗣，星　守：桁選択順序木による探索，昭51・信学全大，＃1104 (1976, 3).

[弓場 77] 弓場敏嗣，星　守：桁選択順序木の準最適構成，昭52・信学全大，＃1115 (1977, 3).

[弓場 79] 弓場敏嗣, 宮川正弘: 最適順序木の評価, 信学技報, EC 78-77, 61-70 (1979, 3).

[弓場 80] 弓場敏嗣: ハッシングを用いた見出し探索の技法, 信学誌, 解説 (1980, 1掲載予定).

[Adel'son-Vel'skii 62] Adel'son-Vel'skii, G. M. and Landis, E. M. :An Algorithm for the Organization of Information. Dokl. Akad. Nauk SSSR, Vol. 146, 263-266 (1962).

[Aho 74] Aho, A. V., Hopcroft, J. E. and Ullman, J. D.: The Design and Analysis of Computer Algorithms. Addison-Wesley (1974).

[Alager 79] Alager, V. S. and Soochan, C.: Partial Match Retrieval for Non-Uniform Query Distributions. Proc. AFIPS 1979 NCC, Vol. 48, 775-780 (1979).

[Allen 78] Allen, B. and Munro, I.: Self-Organizing Binary Search Trees. J. ACM, Vol. 25, No. 4, 526-535 (Oct. 1978).

[Arora 69] Arora, S. R. and Dent, W. T.: Randamized Binary Search Technique. Commun. ACM, Vol. 12, No. 2, 77-80 (Feb. 1969).

[Bear 75] Bear, J. L.: Weight-Balanced Trees. Proc. AFIPS 1975 NCC, Vol. 44, 467-472 (1975).

[Bear 77] Bear, J. L. and Schwab, B.: A Comparison of Tree-Balancing Algorithms. Commun. ACM, Vol. 20, No. 5, 322-330 (May 1977).

[Bayer 72] Bayer, R. and McGreight, E.: Organization and Maintenance of Large Ordered Indexes. Acta Informatica, Vol. 1, 173-189 (1972).

[Bayer 77a] Bayer, R. and Unterauer, K.: Prefix B-Trees. ACM Trans. on Database Systems, Vol. 2, No. 1, 11-26 (March 1977).

[Bayer 77b] Bayer, R. and Schkolnick, M.: Concurrency of Operations on B-Trees. Acta Informatica, Vol. 9, No. 1, 1-21 (1977).

[Bentley 75a] Bentley, J. L. and Stanat, D. F.: Analysis of Range Searches in Quad Trees. Information Processing Letters, Vol. 3, No. 6, 170-173 (July 1975).

[Bentley 75b] Bentley, J. L.: A Survey of Techniques for Fixed Radius Near Neighbor Searching. Stanford CS Rep. 75-513, Stanford Univ.(1975).

[Bentley 75c] Bentley, J. L.: Multidimensional Binary Search Trees Used for Associative Searching. Commun. ACM, Vol. 18, No. 9, 509-517 (Sept. 1975).

[Bentley 76a] Bentley, J. L. and Burkhard, W. A.: Heuristics for Partial-Match Retrieval Data Base Design. Information Processing Letters, Vol. 4, No. 5, 132-135 (Feb. 1976).

[Bentley 76b] Bentley, J. L. and Yao, A. C.: An Almost Optimal Algorithm for Unbounded Searching. Information Processing Letters, Vol. 5, No. 3, 82-87 (Aug. 1976).

[Bentley 78] Bentley, J. L.: Multidimensional Binary Search Trees in Database Applications. CMU-CS-78-139, Carnegie-Mellon University (Sept. 1978).

[Bitner 79] Bitner, J. R.: Heuristics That Dynamically Organize Data Structures. SIAM J. Comput., Vol. 8, No. 1, 82-110 (Feb. 1979).

[Brown 79a] Brown, M. R.: A Partial Analysis of Random Height-Balanced Trees. SIAM J. Comput., Vol. 8, No. 1, 33-41 (Feb. 1979).

[Brown 79b] Brown, M. R.: Addendum to "A Storage Scheme for Height-Balanced Trees." Information Processing Letters, Vol. 8, No. 3, 154-156 (March 1979).

[Bruno 71] Bruno, J. and Coffman, E. G., Jr.: Nearly Optimal Binary Search Trees. Proc. IFIP Congress 71, North-Holland Pub. 99-103 (1972).

[Burge 76] Burge, W. H. : An Analysis of Binary Search Trees Formed from Sequences of Nondistinct Keys. J. ACM, Vol. 23, No. 3, 451-454 (July 1976).

[Burkhard 73] Burkhard, W. A. and Keller, R. M.: Some Approaches to Best-Match File Searching. Commun. ACM, Vol. 16, No. 4, 230-236 (Apr. 1973).

[Bukhard 76] Burkhard, W. A.: Hashing and Trie Algorithms for Partial Match Retrieval. ACM Trans. on Database Systems, Vol. 1, No. 2, 175-187 (June 1976).

[Cardenas 77] Cardenas, A. F. and Sagamang, J. P.: Doubly-Chained Tree Data Base Organization —Analysis and Design Strategies. Computer Journal, Vol. 20, No. 1, 15-26 (Feb. 1977).

[Casey 73] Casey, R. G.: Design of Tree Structures for Efficient Querying. Commun. ACM, Vol. 16, No. 9, 549-556 (Sept. 1973).

[Coffman 70] Coffman, E. G., Jr. and Eve, J.: File Structures Using Hashing Functions. Commun ACM, Vol. 13, No. 7, 427-436 (July 1970).

[Comer 77] Comer, D. and Sethi, R.: The Complexity of Trie Index Construction. J. ACM, Vol. 24, No. 3, 428-440 (July 1977).

[Comer 79] Comer, D.: The Ubiquitous B-Tree. Comput. Surv., Vol. 11, No. 2, 121-137 (June 1979).

[Dobkin 76] Dobkin, D. and Lipton, R.: Multidimensional Searching Problems. SIAM J. Comput., Vol. 5, No. 2, 181-186 (June 1976).

[Driscoll 78] Driscoll, J. R. and Lien, Y. E.: A Selective Traversal Algorithm for Binary Search Trees. Commun. ACM, Vol. 21, No. 6, 445-447 (April 1978).

[Finkel 74] Finkel, R. A. and Bentley, J. L.: Quad Trees a Data Structure for Retrieval on Composite Keys. Acta Informatica, Vol. 4, 1-9 (1974).

[Foster 65] Foster, C. C.: Information Storage

and Retrieval Using AVL Trees. Proc. ACM, 20 th Natl. Conf., 192-205 (1965).

[Foster 73] Foster, C. C.: A Generalization of AVL Trees. Commun. ACM, Vol. 16, No. 8, 513-517 (Aug. 1973).

[Fredkin 60] Fredkin, E.: Trie Memory. Commun. ACM, Vol. 3, No. 9, 490-499 (Sept. 1960).

[Friedman 75] Friedman, J. H., Bentley, J. L. and Finkel, R. A.: An Algorithm for Finding Best Match in Logarithmic Time. STAN-CS-75-482, Stanford Univ. (Feb. 1975).

[Garey 72] Garey, M. R.: Optimal Binary Identification Procedures. SIAM J. Appl. Math., Vol. 23, No. 2, 173-186 (Sept. 1972).

[Garey 74a] Garey, M. R.: Optimal Binary Search Trees with Restricted Maximal Depth. SIAM J. Comput., Vol. 3, No. 2, 101-110 (June 1974).

[Garey 74b] Garey, M. R. and Graham, R. L.: Performance Bounds on the Splitting Algorithm for Binary Testing. Acta Informatica, Vol. 3, 347-355 (Dec. 1974).

[Garsia 77] Garsia, A. M. and Michelle L. W.: A New Algorithm for Minimum Cost Binary Trees. SIAM J. Comput., Vol. 6, No. 4, 622-642 (Dec. 1977).

[Gilbert 59] Gilbert, E. N. and Moore, E. F.: Variable-Length Binary Encodings. Bell Sys. Tech. J., 933-967 (July 1959).

[Hibbard 62] Hibbard, T. N.: Some Combinatorial Properties of Certain Trees with Applications to Searching and Sorting. J. ACM, Vol. 9, No. 1, 13-28 (Jan. 1962).

[Hill 78] Hill, E., Jr.: A Comparative Study of Very Large Data Bases. Springer-Verlag, (1978).

[Hirschberg 76] Hirschberg, D. S.: An Insertion Technique for One-Sided Height-Balanced Trees. Commun. ACM, Vol. 19, No. 8, 471-473 (Aug. 1976).

[Horibe 77] Horibe, Y.: An Improved Bound for Weighted-Balanced Trees. Information and Control, Vol. 34, No. 2, 148-151 (June 1977).

[Hoshi 78] Hoshi, M. and Yuba, T.: Searching in Preorder Trees and Its Evaluation. Trans. of IECE of Japan, Vol. E-61, No. 1, 8-14 (Jan. 1978).

[Hu 71] Hu, T. C. and Tucker, A. C.: Optimal Computer Search Trees and Variable-Length Alphabetic Codes, SIAM J. Appl. Math., Vol. 21, No. 4, 514-532 (Dec. 1971).

[Hu 72a] Hu, T. C. and Tan, K. C.: Path Length of Binary Search Trees. SIAM J. Appl. Math., Vol. 22, No. 2, 225-233 (Mar. 1972).

[Hu 72b] Hu, T. C.: A Comment on the Double-Chained Tree, Commun. ACM, Vol. 15, No. 4, p. 276 (Apr. 1972).

[Hu 73] Hu, T. C.: A New Proof of the T-C Algorithm. SIAM J. Appl. Math., Vol. 25, No. 1, 83-94 (July 1973).

[Hu 78] Hu, T. C., Kleitman, D. J. and Tamaki, J. K.: Binary Trees Optimum under Various Criteria. 78-CS-016, Dept. of Applied Physics and Information Science, Univ. of California at San Diego (1978).

[Huffman 52] Huffman, D. A.: A Method for the Construction of Minimum Redundancy Codes. Proc. IRE, Vol. 40, 1098-1101 (1952).

[Hyafil 76] Hyafil, L. and Rivest, R.: Constructing Optimal Binary Decision Trees is NP-Complete. Information Processing Letters, Vol. 1, No. 1, 15-17 (May 1976).

[Itai 76] Itai, A.: Optimal Alphabetic Trees. SIAM J. Comput., Vol. 5, No. 1, 9-18 (March 1976).

[Karlton 76] Karlton, P. L., Fuller, S. H., Scroggs, R. E. and Kaehler, E. B.: Performance of Height-Balanced Trees. Commun. ACM, Vol. 19, No. 1, 23-28 (Jan. 1976).

[Kashyap 77] Kashyap, R. L., Subas, S. K. C. and Yao, S. B.: Analysis of the Multiple-Attribute-Tree Data-Base Organization. IEEE Trans. on Software Engineering, Vol. SE-3, No. 6, 451-467 (Nov. 1977).

[Kennedy 72a] Kennedy, S. R.: A Note on Optimal Doubly-Chained Trees. Commun. ACM, Vol. 15. No. 11, 997-998 (Nov. 1972).

[Kennedy 72b] Kennedy, S. R.: Optimal Weighted Doubly-Chained Trees. Information Science TR. No. 1, Calitech. (May 1972).

[Knuth 68] Knuth, D. E.: The Art of Computer Programming, Vol. 1, Fundamental Algorithms. Addison-Wesley, 1968 (Revised 1973).

[Knuth 71] Knuth, D. E.: Optimum Binary Search Trees. Acta Informatica, Vol. 1, No. 1, 14-25 (1971).

[Knuth 73] Knuth, D. E.: The Art of Computer Programming, Vol. 3, Sorting and Searching. Addison-Wesley (1973).

[Knuth 76] Knuth, D. E.: The State of the Art of Computer Programming. STAN-CS-76-551, Stanford Univ. (1976).

[Knuth 77] Knuth, D. E.: Deletions That Preserve Randomness. IEEE Trans. on Software Engineering, Vol. SE-3, No. 5, 351-359 (Sept. 1977).

[Kosaraju 78] Kosaraju, S. Rao: Insertions and Deletions in One-Sided Height-Balanced Trees. Commun. ACM, Vol. 21, No. 3, 226-227 (March 1978).

[Landauer 63] Landauer, W. I.: The Balanced Tree and Its Utilization in Information Retrieval. IEEE Trans. on Electronic Computers, Vol. EC-12, No. 5, 863-871 (Dec. 1963).

[Lee 76] Lee, R. C. T., Chin, Y. H. and Chang. S. C.: Application of Principal Component Analysis to Multikey Searching. IEEE Trans. on Software

Engineering, Vol. SE-2, No. 3, 185-193 (Sept. 1976).

[Lee 77] Lee, D. T. and Wong, C. K.: Worst-Case Analysis for Region and Partial Region Searches in Multidimensional Binary Search Trees and Balanced Quad Trees. Acta Informatica, Vol. 9, 23-29 (1977).

[Luccio 76] Luccio, F. and Pagli, L.: On the Height of Height-Balanced Trees. IEEE Trans. on Computers. Vol. C-25, No. 1, 87-90 (Jan. 1976).

[Luccio 78a] Luccio, F. and Pagli, L.: Rebalancing Height Balanced Trees. IEEE Trans. on Computers, Vol. C-27, No. 5, 386-396 (May 1978).

[Luccio 78b] Luccio, F. and Pagli, L.: Power Trees. Commun. ACM, Vol. 21, No. 11, 941-947 (Nov. 1978).

[Lum 70] Lum, V. Y.: Multi-attribute Retrieval with Combined Indexes. Commun. ACM, Vol. 13, No. 11, 660-665 (Nov. 1970).

[Maly 76] Maly, K.: Compressed Tries. Commun. ACM, Vol. 19, No. 7, 409-415 (July 1976).

[Martin 72] Martin, W. A. and Ness, D. N.: Optimizing Binary Trees Grown with a Sorting Algorithm. Commun. ACM, Vol. 15, No. 2, 88-93 (Feb. 1972).

[Maurer 76] Maurer, H. A.: Implementing Dictionaries Using Binary Trees of Very Small Height. Information Processing Letters, Vol. 5, No. 1, 11-14.

[McCreight 77] McCreight, E. M.: Pagination of B*-Trees with Variable-Length Records. Commun. ACM, Vol. 20, No. 9, 670-674.

[Mehlhorn 75] Mehlhorn, K.: Nearly Optimal Binary Search Trees. Acta Informatica, Vol. 5, 287-295.

[Miller 79] Miller, R. E., Pippenger, N., Rosenberg, A. L. and Snyder, L.: Optimal 2, 3-Trees. SIAM J. Comput., Vol. 8, No. 1, 42-59 (Feb. 1979).

[Miyakawa 77] Miyakawa, M., Yuba, T., Sugito, Y. and Hoshi, M.: Optimum Sequence Trees. SIAM J. Comput., Vol. 6, No. 2, 201-234 (June 1977).

[Morrison 68] Morrison, D. R.: PATRICIA—Practical Algorithm to Retrieve Information Coded in Alphanumeric. J. ACM, Vol. 15, No. 4, 514-534 (Oct. 1968).

[Nicklas 76] Nicklas, B. M. and Schlageter, G.: Index Structuring in Inverted Data Bases by TRIES. The Computer Journal, Vol. 20, No. 4, 321-324 (1976).

[Nievergelt 71] Nievergelt, J. and Wong, C. K.: On Binary Search Trees. Proc. IFIP Congress 71, North-Holland Pub., 91-98 (1972).

[Nievergelt 72] Nievergelt, J., Paradels. J., Wong, C. K. and Yue, P.C.: Bounds on the Weighted Path Length of Binary Trees. Information Processing Letters 1, 220-225 (1972).

[Nievergelt 73a] Nievergerlt, J. and Wong, C. K.: Upper Bounds of the Total Path Length of Binary Trees. J. ACM, Vol. 20, No. 1, 1-6 (Jan. 1973).

[Nievergelt 73b] Nievergelt, J. and Reingold, E. M.: Binary Search Trees of Bounded Balance. SIAM J. Comput., Vol. 2, No. 1, 33-43 (Mar. 1973).

[Nievergelt 74] Nievergelt, J.: Binary Search Trees and File Organization. Comput. Surv., Vol. 6, No. 3. 195-207 (Sept. 1974).

[Patt 69] Patt, Y. N.: Variable Length Tree Structures Having Minimum Average Search Time. Commun. ACM, Vol. 12, No. 2, 72-76 (Feb. 1969).

[Payne 77] Payne, H. J. and Meisel, W. S.: An Algorithm for Constructing Optimal Binary Decision Trees. IEEE Trans. on Computers, Vol. C-26, No. 9, 905-916 (Sep. 1977).

[Pollack 65] Pollack, S. L.: Convertion of Limited-Entry Decision Tables to Computer Programs, Commun. ACM, Vol. 8, No. 11, 677-682 (Nov. 1965).

[Raghavan 77] Raghavan, V. and Yu, C. T.: A Note on a Multidimensional Searching Problem. Information Processing Letters. Vol. 6, No. 4, 133-135 (Aug. 1977).

[Reingold 71] Reingold, E. M.: Notes on AVL Trees. CS-Rep. No. 441, Univ. of Illinois (May 1971).

[Reinwald 66] Reinwald, L. T. and Soland. R. M.: Convertion of Limited-Entry Decision Tables to Optimal Computer Programs (1): Minimum Average Processing Time. J. ACM, Vol. 13, No. 3, 339-358 (July 1966).

[Reinwald 67] Reinwald, L. T. and Soland, R. M.: Convertion of Limited-Entry Decision Tables to Optimal Computer Programs (2): Minimum Storage Requirement. J. ACM, Vol. 14, No. 4, 742-755 (Oct. 1967).

[Rivest 74] Rivest, R. L.: On the Optimarity of Elias's Algorithm for Performing Best-Match Searches. Information Processing 74, North-Holland Pub., 678-681 (1974).

[Rivest 76a] Rivest, R.: On Self-Organizing Sequential Search Heuristics. Commun. ACM, Vol. 19, No. 2, 63-67 (Feb. 1976).

[Rivest 76b] Rivest, R. L.: Partial-Match Retrieval Algorithms. SIAM J. Comput., Vol. 5, No. 1, 19-50 (March 1976).

[Rosenberg 78] Rosenberg, A. L. and Snyder, L.: Minimal-Comparison 2, 3-Trees. SIAM J. Comput., Vol. 7, No. 4, 465-480 (Nov. 1978).

[Rothnie 74] Rothnie, J. B., Jr. and Lozano, T.: Attribute Based File Organization in a Paged Memory Environment. Commun. ACM, Vol. 17, No. 2, 63-69 (Feb. 1974).

[Samadi 76] Samadi, B.: B-Trees in a System

with Multiple Users. Information Processing Letters, Vol. 5, No. 4, 107-112 (Oct. 1976).

[Schkolnick 75] Schkolnick, M.: The Optimal Selection of Secondary Indices for Files. Information Systems, Vol. 1, 141-146 (1975).

[Schkolnick 77] Schkolnick, M.: A Clustering Algorithm for Hierarchical Structures. ACM Trans. on Database Systems, Vol. 2, No. 1, 27-44 (March 1977).

[Severance 74] Severance, D. G.: Identifier Search Mechanisms: A Survey and Generalized Model, Comput. Surv., Vol. 6, No. 3, 175-194 (Sept. 1974).

[Shapiro 77] Shapiro, M.: The Choice of Reference Points in Best-Match File Searching. Commun. ACM, Vol. 20, No. 5, 339-343 (May 1977).

[Sheil 78] Sheil, B. A.: Median Split Trees: A Fast Lookup Technique for Frequently Occurring Keys. Commun. ACM, Vol. 21, No. 11, 947-958 (Nov. 1978).

[Silva-Filho 79] Silva-Filho, Y. V.: Average Case Analysis of Region Search in Balanced k-d Trees. Information Processing Letters, Vol. 8, No. 5, 219-222 (June 1979).

[Stanfel 69] Stanfel, L. E.: A Comment on Optimal Tree Structures. Commun. ACM, Vol. 12, No. 10, p. 582 (Oct. 1969).

[Stanfel 70] Stanfel, L. E.: Tree Structures for Optimal Searching. J. ACM, Vol. 17, No. 3, 508-517 (July 1970).

[Stanfel 72] Stanfel, L. E.: Practical Aspects of Doubly Chained Trees for Retrieval. J. ACM, Vol. 19, No. 3, 425-436 (July 1972).

[Sussenguth 63] Sussenguth, E. H., Jr.: Use of Tree Structures for Processing Files. Commun. ACM, Vol. 6. No. 5, 272-279 (May 1963).

[Tan 71] Tan, K. C.: Path Length of Binary Search Trees. Ph. D. Thesis, Univ. of Wisconsin (1971).

[Tan 72] Tan, K. C.: On Foster's Information Storage and Retrieval Using AVL Trees. Commun. ACM, Vol. 15, No. 9, p. 843 (Sept. 1972).

[Unterauer 79] Unterauer, K.: Dynamic Weighted Binary Search Trees. Acta Informatica, Vol. 11, No. 4, 340-362 (1979).

[Walker 72] Walker, W. A. and Gotlieb, C. C.: A Top-Down Algorithm for Constructing Nearly Optimal Lexicographic Trees. In R. C. Read (ed.): Graph Theory and Computing. Academic Press. 303-323 (1972).

[Walker 75] Walker, A. and Wood, D.: Locally Balanced Binary Trees. The Computer Journal, Vol. 19, No. 4, 322-325 (1975).

[Wessner 76] Wessner, R. L.: Optimal Alphabetic Search Trees with Restricted Maximal Height. Information Processing Letters, Vol. 4, No. 4, 90-94 (Jan. 1976).

[Yao 78] Yao, A. C-C.: On Random 2-3 Trees. Acta Informatica, Vol. 9, 159-170 (1978).

[Yohe 72] Yohe, J. M.: Hu-Tucker Minimum Redundancy Alphabetic Coding Method. Commun. ACM, Vol. 15, No. 5, 360-362 (1972).

[Yuval 76] Yuval, G.: Finding Nearest Neighbours. Information Processing Letters, Vol. 5, No. 3, 63-65 (Aug. 1976).

[Zolonwsky 78] Zolonwsky, J. E.: Topics in Computational Geometry. STAN-CS-78-659, Stanford Univ. (May 1978).

[Zweben 78] Zweben, S. H. and McDonald, M. A.: An Optimal Method for Deletion in One-Sided Height-Balanced Trees. Commun. ACM, Vol. 21, No. 6, 441-445 (June 1978).

(昭和54年9月12日受付)