数理解析研究所講究録
第353巻 1979年 87-96

2分検索木上のセレクティブトラバースに関するある改良

広島大学　工学部　菊野 亨
楠本 博巳

1. まえがき

2分検索木は格納されているレコードへのアクセス及びファイルの変更が容易であるので，大規模なファイルを構成する際の重要な技法としてよく知られている[1]．通常，2分検索木の各節点v_iには(k_i, P_i)の2項組のデータが保持される．ここでk_iはキーと呼ばれ，P_iは対応するレコードが実際に格納されているメモリ上の領域へのポインタとする．レコードへのアクセスに関する興味ある問題の1つに，区域条件$C_j = (l_j, r_j)$（l_j, r_jは共に非負整数で，$l_j < r_j$）の系列$L = C_1 C_2 \cdots C_n$が与えられ，少くとも1つのC_jに対し$l_j < k_i < r_j$なるキーをもつ2分検索木上の全ての節点を，その途中の過程で参照する節点の数を最小にしてアクセスするというセレクティブトラバースの問題がある．

Driscollらは2分検索木上の各節点に3ビットのマーカ

フィールドを設け，７つのマーカを用いることにより，上述の問題を能率よく解いている(1)．本論文では，若干の（時間的な）オーバヘッドを犠牲にすれば，４つのマーカで十分であること，すなわちマーカフィールドを２ビットにまで減じることが可能であることを示す．

2. 準備

［定義１］　２分検索木Tの各節点v_iに格納されているデータはポインタフィールド，データフィールド及びマーカフィールドの３つに分かれている．

（ｉ）ポインタフィールドには，左部分木へのポインタLEFT，右部分木へのポインタRIGHTが含まれている．

（ii）データフィールドには２項組のデータ（k_i，P_i）が保持されている．ここでk_iはキーと呼ばれN（Nは非負整数全体の集合）の要素である．P_iは対応するレコードが格納されている領域へのポインタであるが，今の場合，P_iの値については関心がないので省くことにする．

（iii）マーカフィールドは３ビットの容量をもつ．

［定義２］　区域条件C_jとは２項組（l_j，r_j）のことである．但し，l_jとr_jは共にNのある要素であって，関係式$l_j < r_j$が成立するものとする．区域条件の系列$C_1\ C_2 \cdots C_n$をLと表す．Lにおいて$l_p \leq l_q < r_p$あるいは$l_p < r_q \leq r_p$を満たす相

異なる添字P，qが存在するなら，Lにはオーバラップが存在するという．

［定義３］　２分検索木T上の節点v_iのキーk_iと区域条件$C_j=(l_j, r_j)$に対し，$l_j < k_i < r_j$が成立するなら，節点v_iは区域条件C_jを満たすという．Tと区域条件の系列Lに対し，Lに属する区域条件の少なくとも１つを満たす節点の集合をT(L)と表す．

［定義４］　２分検索木Tと区域条件Lが与えられ，その途中の過程で参照する節点の数を最小に押えてT(L)に属する節点を全てアクセスする問題をセレクティブトラバース（selective traverse）の問題と呼ぶ．以降では単に問題STと記すことにする．

Driscollらは，区域条件の系列Lにオーバラップが存在しない場合について問題STを解くアルゴリズムSELECT（以降ではアルゴリズムSと呼ぶ）を与えている．アルゴリズムSでは，２分検索木Tの各節点にあるマーカフィールド（３ビット）上で識別可能な７つのマーカi，l，r，α，s，n，fを用いており，スタック，キューなどは用いていない．

3．改良

アルゴリズムSに対し，使用するマーカの数を７から４に

減じる，すなわちマーカフィールドのビット数を1ビット減じることが可能であることを示す．

3．1　基本方針

ここで提案するアルゴリズムはDriscollらと基本的に同じ立場をとり，スタックは用いず，各節点ごとに付加したマーカフィールドの値を参照しながら2分検索木Tをたどる．但しマーカフィールドを1ビット減じて2ビットとするため，使用するマーカもi，ℓ，r，aの4つとする．改良アルゴリズムは次の2つのフェーズに分かれている．

フェーズI：TとLに対し，集合T(L)に属する節点を含む部分木の周囲に位置する節点に対し，マーカℓ，rとaを境界記号として書き込む．

フェーズII：フェーズIで準備した各節点のマーカと，必要ならば左子節点および右子節点のマーカとを調べながらT上をたどっていって，T(L)に属する節点だけに必要な処理を行う．

フェーズIはマーカsをℓで代用することを除けば，アルゴリズムSとほぼ同じ．フェーズIIでは，アルゴリズムSで用いていたマーカfをℓあるいはrで，マーカnをiでそれぞれ表しておいて，ある節点pにおいて(アルゴリズムSで行ったマーカの区別が)必要となればpの左子節点および右

子節点のマーカを調べることにより，アルゴリズムSのステップⅡの動きを正しくシミュレートしている．

3．2　改良アルゴリズムの概要（マーカの説明）

マーカの説明を行うことによって改良アルゴリズム，特にフェーズⅡの概要を述べる．先ず，アルゴリズムSに存在したが，改良アルゴリズムでは省いたマーカについて簡単に説明する．次に2，3の記号を準備しておく．

lson（ひ）：節点ひの左子節点（left son）．

lsub（ひ）：節点ひの左子節点を頂点とする部分木．

rson（ひ）：節点ひの右子節点（right son）．

rsub（ひ）：節点ひの右子節点を頂点とする部分木．

各マーカの設定されている節点をひとする．

マーカS：改良アルゴリズムにおいては，各節点ひに対し先ずlsub（ひ）内をたどり，その後でrsub（ひ）内をたどることとし，マーカSをマーカlで代用する．

マーカn：マーカiあるいはaのnへの変換を省き，直接，処理を行う．マーカnをマーカiで代用する．

マーカf：節点ひの親に移る時には，lson（ひ），rson（ひ）のマーカが共にiとなっていることに注目して，マーカrあるいはlで代用する．なおlson（ひ）とrson（ひ）のマーカを調べることにより，fの復元は可能（詳細は後述のマ

－カrを参照）.

次に，改良アルゴリズムで使用する４つのマーカについて詳しく述べる．アルゴリズムＳのフェーズⅡの実行前における，マーカrの節点ひの周りの状況は図１の通りである．又マーカℓの節点ひ，マーカαの節点ひの周りの状況はそれぞれ図２，図３の通りである．なお図中のマーカはα∈{r，α，s}，β∈{ℓ，α，s}とする．

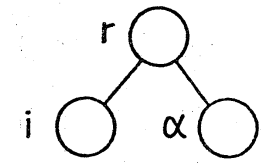


図１　マーカrの節点

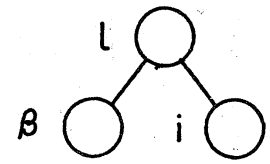
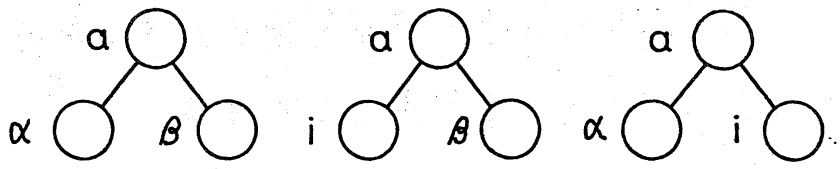


図２　マーカℓの節点



(1) 場合1　　(2) 場合2　　(3) 場合3

図３　マーカαの節点

①マーカi：基本的にはアルゴリズムＳと同じ．

②マーカr：アルゴリズムＳのフェーズⅡでは先ず節点ひのマーカr（図１参照）をfに変換し，rson（ひ）に移り，rsub（ひ）内での実行が終了すると，rson（ひ）のマーカαをiに変換し図４の形で（マーカfの）ひに再び戻ってくる．

そこで改良アルゴリズムのフェーズⅡでは，ひのマーカrを（fに）変換することなく，常にrson（ひ）に移る．必要

なら再びひに戻ってきた時点でrson(ひ)のマーカがiかどうかを調べることによって，rとfの区別は可能である．

③マーカℓ：マーカSの説明で述べた様に，改良アルゴリズムではマーカSをℓで代用するので，図5の形と図2の形のマーカℓの節点ひを同時に取り扱うことが必要となる．図でマーカは$\alpha' \in \{r, a, \ell\}$，$\beta' \in \{\ell, a\}$である．

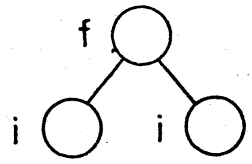


図4 マーカfの節点

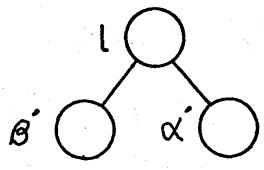


図5 マーカℓの説明図

改良アルゴリズムのフェーズⅡでは，節点ひのマーカℓをrに一旦変換しておいて，ℓson(ひ)に移る．こうして，ℓsub(ひ)内での実行が終了して再びひに戻った時，(②で述べた手順に従えば)図2なら図4(fはrとなっている)の形に，図5なら図1の形にそれぞれ遷移して，ℓとSの区別が正しく実現される．

④マーカa：アルゴリズムSのフェーズⅡではマーカaは図3の形で現れる．この時，マーカiの節点(ℓson(ひ)あるいはrson(ひ))を頂点とする部分木内には，ある区域条件を満たす(マーカiの)節点だけが存在する．改良アルゴリズムのフェーズⅡでは，ℓson(ひ)のマーカがiか否か判定し，iならℓsub(ひ)に対しLEFTINNERを呼ぶ．

ＬＥＦＴＩＮＮＥＲの実行の最後に（ひ∈了（ム）なので）ひに対し処理を行い，ひのマーカαをｒに変換してひに戻り，以降は②のマーカｒの手順に帰着される．ｉでないならひのマーカαをｉに変換しℓson（ひ）に移る．ℓsub（ひ）内での実行が終了してひに戻ると，ひに対し処理を行う．次にrson（ひ）のマーカがｉか否かを判定し，ｉならひのマーカｉをαに変換し，rsub（ひ）に対しＲＩＧＨＴＩＮＮＥＲを呼び，ＲＩＧＨＴＩＮＮＥＲの実行の最後にひのマーカαをｒに変換する．以降は②のマーカｒの手順に帰着される．

4．評価

評価に必要な記号の説明をしておく．

Ｃ：定数（例えば区域条件に含まれるℓ_iとかr_i）と２分検索木内の節点のキーの大小比較演算．

Ｔ：２分検索木内の節点のマーカの判定演算．

Ｐ：２分検索木内の節点（に含まれるデータ）に対し実際に処理を行う演算で，アルゴリズムではＰＲＯＣＥＳＳという命令で記述している演算．

次にＮ個の節点を含む完全２分検索木了とオーバラップのない区域条件のリストＬ＝$C_1 C_2 \cdots C_n$に対する問題ＳＴについて，アルゴリズムの実行時間に関する評価を行う．以下ではいずれも最悪の場合を取扱うものとする．但し，＃（C_k）

は区域条件C_kを満たす節点の数とする．

Driscoll らのアルゴリズムSの場合[1]

$$[6n\log N+4\sum_{k=1}^{n}\#(C_k)]T$$
$$+[2n\log N]C+[\sum_{k=1}^{n}\#(C_k)]P\cdots(1)$$

改良アルゴリズムの場合

$$[8n\log N+4\sum_{k=1}^{n}\#(C_k)]T$$
$$+[2n\log N]C+[\sum_{k=1}^{n}\#(C_k)]P\cdots(2)$$

（1）式と（2）式を比較すれば改良アルゴリズムの場合，アルゴリズムSに比べて，（$2n\log N$）だけ余計に時間がかかることが分かる．しかし，上記の3つの演算C，T，Pの内，演算Tに要する時間は他の2つに比べ比較的少ないと予想される[1]．したがって，この時間的損失はマーカフィールドの1ビットの削減により，十分償われると思われる．

5．むすび

Driscoll らが与えていたアルゴリズムSに対し，1ビットの削減が可能であることを示した．なお，改良アルゴリズムの場合の評価を厳密に行えば，$0\leq P\leq 1$なるある数Pに対し，$[2n\log N-P\sum_{k=1}^{n}\#(C_k)]T$だけのオーバヘッドとなり，改良アルゴリズムの方がアルゴリズムSよりも速くなることも多くの場合について実測されている．フェーズⅡをフェーズⅠに組み込んだアルゴリズムも得ており，別の機

会に報告する予定である.

文 献

(1) J.R.Driscoll and Y.E.Lien:"A selective traversal algorithm for binary search trees", CACM,21,6,p.445 (June 1978).