数理解析研究所講究録
第1691巻 2010年 155-161

全二分木の簡潔な表現

馬場雅大* 小野廣隆† 定兼邦彦‡ 山下雅史§

2010/2/2

概説

順序木を表現する簡潔表現にはBP, DFUDSなどが提案されている. 本発表では順序木の中でも全二分木に対する簡潔データ構造を提案する. 提案するデータ構造はDFUDSを基とし, そのサイズは索引も含めて$n+o(n)$ビットであり, 全二分木の情報理論的下限に近くなっている. またDFUDSで可能であった演算はそのまま実行することできる. またDFUDSとは違う形で木を表現するTree Covering [2, 3](TC)を紹介しつつ, それを応用して順序木で深さを求める方法を提案する.

1 準備

1.1 本発表

全二分木の情報理論的下限はnビットであるが, この全二分木に対しても通常の簡潔表現では$2n$ビット必要である. そこで本発表では全二分木を$n+1$ビットで表現し, 合わせて種々の操作を定数時間で行うことを目指した. なお簡潔表現では親ノードを定数時間で求めたりする操作のために$o(n)$の項が必要である. 以下にそのサイズを比較してみる.

表 1.1:全二分木の表現サイズ (ビット)

ポインタ表現	簡潔表現	提案表現
$O(n \lg n)$	$2n+o(n)$	$n+1+o(n)$

*九州大学システム情報科学府
†九州大学システム情報科学研究院
‡国立情報学研究所
§第2著者に同じ

また, この全二分木で表現で深さを求める方法を使い, TCのアルゴリズムを応用して順序木上のノードの深さを求める分かりやすい方法を提案する.

1.2 本稿の構成

まず2章では, 従来の括弧列を用いた簡潔表現について取り上げる. 3章でFarzan MunroのTCについて簡単に述べ, 4章ではそれを活用しつつ, 従来の括弧列表現からノードの深さを導くような方法について説明する.

2 従来からある簡潔データ構造

まず, 本論文で用いる演算について説明する. 計算モデルとしては語長$\Theta(\lg n)$ビットのword RAMを用いる. word RAMモデルは, $\Theta(\lg n)$ビットの数に関する任意の算術演算や, 連続する$\Theta(\lg n)$ビットのメモリに対する入出力が定数時間で行える.

2.1 rank/selectを実行するための簡潔データ構造

$|\mathcal{A}| = \sigma$となるアルファベット$\mathcal{A}$上の文字列$S[1..n]$を考える. ここでSに対する*rank*と*select*を以下のように定義する.

- $rank_c(S,i)$: $S[1..i]$中のcの出現回数
- $select_c(S,i)$: S中で先頭からi番目のcの位置

ビット配列 ($\sigma = 2$) の場合word RAMモデルで定数時間でrank/selectを計算する$n+o(n)$ビッ

トのデータ構造 [5] があり，さらに 1 の数が m 個であるビット配列に対して rank/select を定数時間で計算する $\lg\binom{n}{m} + O(n \lg\lg n/\lg n) = m \lg \frac{n}{m} + \Theta(n) + O(n \lg\lg n/\lg n)$ ビットのデータ構造が存在する (Raman ら [7])．このデータ構造は完全索引付辞書 (fully indexable dictionary, FID) と呼ばれている．$m = O(n/\lg n)$ ビットであれば，このデータ構造のサイズは $O(n \lg\lg n/\lg n)$ となり $o(n)$ で収まる．

また rank/select などで用いられる表引き を本論文では括弧列に対しても用いる．

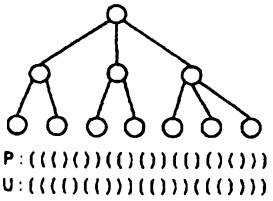

図 1: BP 表現と DFUDS 表現

2.2 括弧列表現と用いられるデータ構造

2.2.1 BP 表現 [6]

BP(Balanced Parenthesis Encoding) 表現とは，木を行きがけ順に巡回し，ノードの行きがけに開き括弧 '(' ，帰りがけに閉じ括弧 ')' を配置していくもの．括弧は全体でバランスし，括弧列 P の長さは $2n$ である．

2.2.2 DFUDS 表現 [1]

DFUDS(Depth-First Unary Degree Sequence) 表現とは，木を行きがけ順で巡回し，ノードの行きがけにそのノードの次数分だけ開き括弧を並べたあとに閉じ括弧を 1 個おくもの．括弧列の先頭に '(' を 1 つだけ追加することで，BP と同じように括弧列 U は全体でバランスし，U の長さは $2n$ である．

2.2.3 BP，DFUDS で共通するデータ構造

n ノードの根付き順序木 T に対して，括弧列を用いて表現する BP，DFUDS では木の巡回を行うために，以下の操作を $o(n)$ ビットで定数時間で行う補助データ構造が提案されている [6]．対象となる括弧列を P とすると以下の通りである．

- $findopen(P, x)$: $P[x]$ にある開き括弧に対応する閉じ括弧の位置を返す．
- $findclose(P, x)$: $P[x]$ にある閉じ括弧に対応する開き括弧の位置を返す．
- $enclose(P, x)$: $P[x]$ にある括弧とそれに対応する括弧を囲う最小の括弧対の開き括弧の位置を返す．

これらと括弧列および括弧列に対する rank/select 索引により以下の操作のうち，BP では $child$, $childrank$, $degree$ を除き，DFUDS では lca, $depth$, LA を除く操作を定数時間で行うことができる．

- $isleaf(x)$: x は葉であるか．
- $parent(x), firstchild(x), sibling(x)$: x の親，長男，弟
- $preorder\_rank(x)$: x の行きがけ順
- $preorder\_select(i)$: 行きがけ順が i であるノードの簡潔表現での位置
- $desc(x)$: x の子孫の数
- $degree(x)$: x の子の数 (次数)
- $child(x, i)$: x の左から i 番目の子
- $child(x, i)$: x がその親の左から何番目の子か
- $lca(x, y)$: x と y の最近共通祖先
- $depth(x)$: x の深さ
- $LA(x, d)$: x の祖先で深さが d のもの

2.3 区間最小値問合せとそれを用いた最近共通祖先の求め方

2.3.1 区間最小値問合せ (RMQ)

区間最小値問合せ (RMQ) の定義は以下である．

問題 1 配列 $E[1..n]$ が与えられたとき，$RMQ_E(i, j)$ は部分配列 $E[i, j]$ の中の最小値のうち最左のものの位置である．

この問題を解く省スペース・線形時間前計算を実現する方法は多く発表されているが，本稿では [8] を利用する．

2.3.2 最近共通祖先 (LCA)

LCA を RMQ から求める Jansson らの方法 [4] の概略を示す．U の超過数列を $E[i] = (U[0..i]$ 中の '(' の数 $) - (U[0..i]$ 中の ')' の数) とする．E は長さ $2n$ の ± 1 配列であり，値はすべて 0 以上である．

定理 1 順序木上で 2 つのノード x と y の LCA は $DFUDS$ と $o(n)$ ビットの補助データ構造を用いれば定数時間で以下のようにして求めることができる *[4]*. x, y, v, s, z は各ノードの U での位置と考える *(図 2 参照)*. $x = y$ のときは $x = y = z$ と自明である．

$z = lca(x, y)$ $(x < y)$ を求める手順はもし x が葉ノードであれば $x' := x$，そうでなければ $x' := firstchild(x) - 1$ とする．ここで x' はノード x を表す括弧列の末端の U での位置である．

$s := RMQ_E(x, y-1) + 1$

$z := parent(s)$

なお s は z の子のうち y を子孫にもつノードを示す．

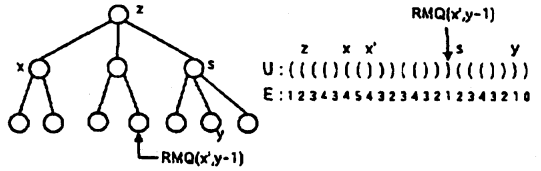

図 2: 最近共通祖先の例
順序木に対する DFUDS 表現 U とその超過数列 E

3 木分割アルゴリズム

木の分割に関しては Geary らのもの [3] があるが，本稿では Farzan, Munro の分割方法 [2] を利用する．

Farzan, Munro の木分割アルゴリズム [2] は設定したパラメータ L に対して木をサイズ $2L$ 以下の部分木 $O(n/L)$ 個に分割するものである．

3.1 Farzan, Munro の木分割アルゴリズムの性質

定義 1 L を一定のパラメータとして以下のようなノードをそれぞれ定義する．

- *heavy-node:*そのノード自身も含めて子孫の数が L 個以上であるノード
- *petiolar-node:heavy-node* を子に持たない *heavy-node*
- *stalk-node:heavy-node* を 1 つだけ子に持つ *heavy-node*
- *branching-node:*少なくとも 2 つの *heavy-node* を子に持つノード

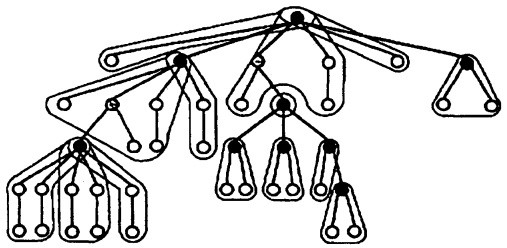

図 3: $L = 3$ で分割した例

ノードを囲んでいる実線はこのアルゴリズム [2] を $L = 3$ で実行した後のグループを示している．また色つきノードは $heavy - node$ であることを示しており，黒丸はそのうちグループの根となったものを示している．

定理 2 *Farzan, Munro* の分割アルゴリズムにより順序木 T の分割後の性質をまとめる．

1. T は大きさが高々 $2L$ のグループに分割でき，その際グループの数は $O(n/L)$ 個とすることができる．

2. これらのグループはそのグループの根の部分を除けば，互いに素である．

3. 各グループにおいてその根ノードから出る枝を除けば，グループ内のノードから他のグループにつながるような枝は高々 *1* 本である．

共に高々 n/L 個である *petiolar - node* と *branching-node* ではそれを根としたグループをつくり，*stalk-*

$node$についてはそのようにするか場合分けが行われる．ノード数L以上ののグループ$O(n/L)$個とそれ以下のグループが同じく$O(n/L)$個発生する．

4 全二分木の表現と操作プロセス

4.1 全二分木表現

4.1.1 全二分木の括弧列表現

全二分木に対して，行きがけ順に内部ノードであれば開き括弧'('，葉ノードであれば閉じ括弧')'を配置していく．この場合閉じ括弧が1つ多くなるので，先頭に開き括弧を追加する．この全二分木表現をFとする．Fの長さは$n+1$である．

括弧列を用いて，行きがけ順に表現していくほか，括弧列の先頭に仮想根として開き括弧を追加するのは$DFUDS$と同じである．しかし全二分木の場合，次数は0か2のどちらかである．そこで$DFUDS$の次数を表す部分を削減し，次数が2すなわち内部ノードであれば$($を配置し，次数0すなわち葉ノードであれば0を配置するようにしたものである．また順番に1つずつ括弧を並べているので，$F[0,..n]$中，行きがけ順でx番目のノードを示している括弧は$F[x]$である．

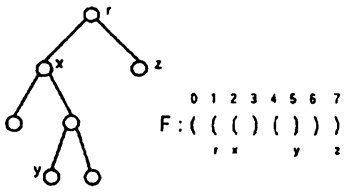

図 4: 全二分木表現の例

4.1.2 全二分木表現における操作の方法

2章で挙げた操作のうち，$findclose$などをのみ用いてできる操作は$DFUDS$と同じである．なおlcaについては$DFUDS$同様にRMQを用いる．

- $isleaf(x) : F[x] =)$ ならば yes．それ以外 no.
- $parent(x) : F[x-1] = ($ ならば $x-1$，それ以外 $findopen(x-1)$
- $firstchild(x) : x+1$
- $lastchild(x) : findclose(x)+1$
- $sibling(x) : F[x-1] = ($ ならば $lastchild$ $(parent(x))$，それ以外の場合は存在しない．
- $degree : F[x] =)$ ならば0，それ以外2．
- $desc(x) : F[x] =)$ ならば0，それ以外 $findclose(enclose(x)) - x + 1$
- $childrank(x) : F[x-1] = ($ ならば1，それ以外2．
- $child(x,i) : i = 1$ ならば $firstchild(x)$，$i = 2$ ならば $lastchild(x)$ である．$i \neq 1,2$ は存在しない．

また，lcaについてもRMQを用いればよい．2章のときと同様に$z = lca(x,y)$とする．$x < y$のとき$s = RMQ_E[x, y-1]+1$でzの子でyを子孫にもつノードsを見つけ出すことができるので，この$parent(s)$がzになる．

4.2 グループの根ノードを求める手法の提案

分割アルゴリズム [2] により，あるノードxはいずれかのグループに属しているとする．このグループの根を$R[x]$として以下のような問題を考える．

問題 2 x，$R[x]$を行きがけ順とする．このときノードxを含んだグループの根$R[x]$を返す．

4.2.1 RMQを利用した$R[x]$を求める手法の提案

提案 1 根であることを表す長さ$2n$のビット配列$B_r[1,\ldots,2n]$と$DFUDS$におけるLCAを用いて，$DFUDS$上でxの位置から$R[x]$の位置を求める．

$DFUDS$上で，p,qの位置にあるノードの最近共通祖先の位置を$lca(p,q)$，$(p<q)$とする．またグループの根となっているあるノードの$DFUDS$表現での位置をcとしたとき，$B_r[c]=1$となるような長さ$2n$

のビット配列をつくる．グループの根の数は $O(n/L)$ なので $L = \Theta(\lg n)$ であれば，*FID* を用いた B_r のサイズは $o(n)$ である．

木を分割した後の状態では，*preorder* 順で x の直前にあるグループの根が $R[x]$ になるとは限らない．しかし，*Farzan*，*Munro* の分割アルゴリズム *[2]* を用いれば定理 *2* より各グループはその根を除けば，他のグループに伸びている枝は高々 1 本である．そのため *preorder* で巡回したとき，x と $R[x]$ の間では以下の 4 パターンが考えられる．以下分かりやすくするため x，$R[x]$，s などはそれぞれ *DFUDS* 表現 U での位置を表しているとする．また $x = R[x]$ のときは自明なので以下では省略する．

パターン

1. $R[x]$ と x の間に他にグループの根は存在しない．
2. $R[x]$ と x の間に他のグループの根が存在している．しかし，それは $R[x]$ の子のうち x を子孫に持つノード s を根とする部分木には存在せず，r_i と s の間に 1 個以上存在している．
3. $R[x]$ と x の間に他のグループの根が存在している．それは s を根とする部分木に含まれている．つまり s と x の間には 1 個以上存在しているが $R[x]$ と s の間には存在しない．
4. r_i と x の間に他のグループの根が存在している．それは s と x の間，$R[x]$ と s の間にそれぞれ 1 個以上存在している．

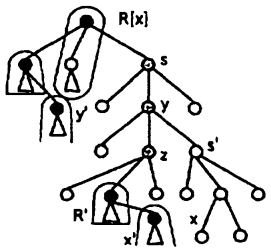

図 5: パターン 4 の例

最も単純なのはパターン 1 の場合であり，直前にある根を示せばよいので B_r に対する *rank/select* を用いて $select_1(B_r, (rank_1(B_r, x)))$ で求めることができる．しかし，パターン 2 ～ 4 のような場合も考えられる．特にパターン 4 の場合，$R[x]$ と x の間にある他のグループの根は少なくとも 2 個存在する．これらを無視するため *LCA* を利用する．以下にそのアルゴリズムを示す．なお x'，y，y'，w，R' も x，$R[x]$ 同様に *DFUDS* 表現 U での位置を表しているとする．

x から $R[x]$ を求めるアルゴリズム

1. $x' := select_1(B_r, rank_1(B_r, x))$
2. $y := lca(x, x')$
3. $y' := select_1(B_r, rank_1(B_r, y))$
4. $w := lca(y, y')$

これにより得られる w が $R[x]$ となっている．

4.3 順序木のノードの深さを求める手法の提案

BP 表現では，超過数列がそのまま深さに対応するのに対して *DFUDS* 表現は超過数列は深さに対応していない．そこで *DFUDS* 上でパイオニアという概念を用いて深さを求める方法 *[4]* のほか，*TC* で 2 レベルに再帰して求めることが提案されてきた．本章では *DFUDS* を基本としつつも，*[2]* の分割アルゴリズムを応用してノードの深さを定数時間で求める方法を提案する．

まず全二分表現 F に対して深さを求める方法を提案し，それを応用して一般の順序木の *DFUDS* 表現 U 上でどのようにして深さを求めるのか考察する．

4.3.1 Farzan，Munro の分割で括弧列から深さを求める方法の提案

全二分木の表現 F は *DFUDS* と似通っており，*DFUDS* において深さを求める *Jansson* らの方法 *[4]* を流用することが可能だが，本論文では *TC* の考え

方を応用しながら，深さを求めるのに必要な括弧列のみを取り出す方針をとる．

まず表引きで x とグループの根 $R[x]$ との深さの差を求める点は同じである．しかし，$R[x]$ の深さはさらに大きなグループの根に再帰せずに求める．分かりやすくするため以下のようにまとめると，
$depth(x) =$ (① x と $R[x]$ の深さの差 $depth(\delta)$)
+ (② $R[x]$ の深さ $depth(R[x])$).
としてこの2つの求め方を示す．

全二分木表現の場合

[3] に対して本論文では木全体を1つの *DFUDS* で表現しておき，その中から必要な括弧列のみを定数回の操作で取り出して表引きする．

① x と $R[x]$ の深さの差 $depth(\delta)$ の求め方

まず $L = \frac{1}{4}\lg n$ として木を分割した上で，*4.2* 節の方法で $R[x]$ を求める．すると $R[x]$ と x の関係を以下の4つに場合分けされる．

1. $R[x]$ と x との間に他のグループの根ノードが存在しない．
2. $R[x]$ と x との間には1個以上他のグループの根ノードが存在する．
   - *(a)* x は $R[x]$ の左分木に存在．
   - *(b)* x は $R[x]$ の右分木に存在．
     - *i.* $R[x]$ の右分木内では $R[x]$ と x の間に他のグループの根が存在しない．
     - *ii.* $R[x]$ の右分木内で $R[x]$ と x の間に他のグループの根が存在する．

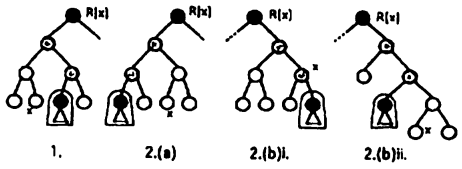

図 6: x と $R[x]$ の位置などで場合分けした例

x は $R[x]$ を根としたグループに存在しているとする．x と $R[x]$ が共に含まれていないグループは実線で囲んでいる部分とする．

それぞれの場合で取り出す括弧列を $Q[1,\ldots,k_x]$ とする．以下が取り出す Q のパターンである．

- *1.* の場合は
  $Q[1,\ldots] := F[R[x],\ldots,x]$
- *2(a)* の場合は
  $Q[1,\ldots] := F[R[x],\ldots,y]\ F[lastchild(y),\ldots,x]$
- *2(b)i* の場合は
  $Q[1] := ($
  $Q[2,\ldots] := F[lastchild(R[x]),\ldots,x]$
- *2(b)ii* の場合は
  $Q[1] := ($
  $Q[2,\ldots] := F[lastchild(y),\ldots,y]$
  $Q[y - lastchild(R[x]) + 3,\ldots]$
  $:= F[lastchild(y),\ldots x]$

ここで k_x は取り出したい括弧列の長さであるので，*1.* の場合なら $k_x = x - R[x] + 1$，*2(b)ii* の場合なら $k_x = y - lastchild(R[x]) + 3 + x - lastchild(y)$ といった具合にして分かる．

2(b)ii の場合に，括弧列を取り出してみた例が図 *7* である．F 上で $R[x]$ を示す括弧から x を示す括弧までを全部または一部を $Q[1,\ldots,2L]$ の先頭から並べて表引きを行う．表のサイズは $2^{2L-1}\cdot(2L-1)\cdot\lg(2L-1) = O(\sqrt{n}\lg n\lg\lg n) = o(n)$ ビットである．

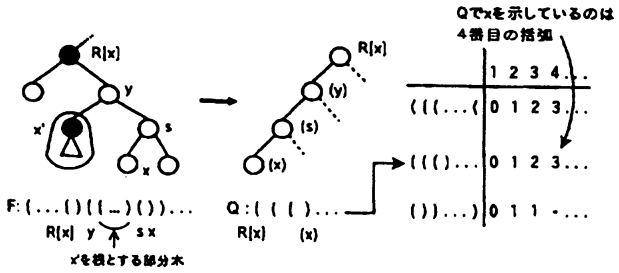

図 7: 表引きで $depth(\delta)$ を求める表の例

全二分木を表している F から長さ $k_x = 4$ の括弧列を取り出して，そこから全二分木を描いたものと，表を用いて $depth(\delta) = 3$ を求める様子．$Q[k_x+1,\ldots]$ は必要はないので $F[x+1,\ldots]$ でもすべて開き括弧であってもかまわない．

② $R[x]$ の深さ $depth(R[x])$ の求め方

グループの根となっているノードにのみ深さを格

納する．根の数は $O(n/L)$ である．グループの根となっているノードを行きがけ順に並べて k 番目にくるノードを r_k とする．$1 \leq k \leq$(グループの根の数) である．ここで $D_r[k] = depth(r_k)$ とする．この配列を基に以下のようなビット配列と 1 進数配列の FID を作成する．なお木 T の根 r_1 の深さは 0 とする．

- $A : D_r[k-1] < D_r[k]$ ならば 0，それ以外は 1 となるビット配列．ただし $A[1] = 1$ とする．
- $D_1 : D_r[k-1] < D_r[k]$ ならば 0 を $D_r[k] - D_r[k-1]$ 個並べてその次に 1 をおく 1 進数配列．
- $D_2 : D_r[k-1] \geq D_r[k]$ ならば 0 を $D_r[k-1] - D_r[k]$ 個並べてその次に 1 をおく 1 進数配列．

いずれの配列のサイズも $o(n)$ ビットとすることができる．

$R[x]$ の深さは以下のように求める．まず $R[x]$ がグループの根としては i，$(1 \leq i \leq k)$ 番目にくるとしたとき，$i = rank_1(B_r, R[x])$ である．木そのものの根から r_i にいたるまでに i_1 回増加して i_2 回減少してきたとすると，$i_1 = rank_1(A, r_i)$，$i_2 = i - i_1$ である．これらを用いて $R[x]$ の深さ $depth(R[x])$ は以下のようになる．

配列 A，D_1，D_2 を用いて深さを求める手順

$depth(R[x]) = depth(r_i)$

$= \{select_1(D_1, i_1) - i_1\} - \{select_1(D_2, i_2) - i_2\}$

新たに必要となるのは，i_1 と i_2 を求めるための A の $rank$ 索引とそれにより深さの増加・減少分をそれぞれ導き出すための D_1・D_2 の $select$ 索引である．

順序木を表す $DFUDS$ の場合

順序木の場合，$depth(R[x])$ に関しては同じであるが，x と r_i の深さの差 $depth(\delta)$ を求める場合が異なるものの，定数回の操作で必要な括弧列のみ取り出すことが可能である．

表引きすることを考えると，取り出す括弧列がグループのノード数の高々 2 倍であるため $L = \frac{1}{8} \lg n$ とする．当然のことながら全二分木表現 F を用いた場合とは表も異なる．

5 おわりに

全二分木の括弧列と *Farzan, Munro* の TC を活用することで深さを求める別の手法を提案した．既存の方法と索引サイズのオーダーは変わらないが，よりわかりやすく求めることができる．また木構造の形状エントロピーまで圧縮できることが *[4]* で示されている $DFUDS$ を格納しておくことにも応用できる．一方で順序木の簡潔データ構造の弱点としてデータの動的な構造変化に時間がかかるという課題がある．

参考文献

[1] D. Benoit, E. D. Demaine, J. I. Munro, R. Raman, V. Raman, S. S. Rao : " Representing trees of higher degree." Algorithmica 43(4), *275–292, 2005.*

[2] A. Farzan, J. I. Munro : "A Uniform Approach Towards Succint Representation of Trees." Algorithm Theory - SWAT 2008, *pages. 173–184, 2008.*

[3] Richard F. Geary, R. Raman, V. Raman : "Succinct ordinal trees with level- ancestor queries." ACM Transactions on Algorithms 2, *4 (2006) 510–534.*

[4] J. Jansson, K. Sadakane, Sung, W.-K. : "Ultra-succinct representation of ordered trees." In SODA (2007), N. Bansal, K. Pruhs, and C. Stein, Eds., SIAM, *pages. 575–584.*

[5] J. I. Munro : "Tables" In Proceedings of the 16th Conference on Foundations of Software Technology and Computer Science (FSTTCS '96), *LNCS 1180, pages 37–42, 1996.*

[6] J. I. Munro, V. Raman : "Succint Representation of Balanced Parentheses and Static Trees." SIAM Journal on Computing, *31(3):762–776, 2001.*

[7] R. Raman, V. Raman, S. S. Rao : "Succinct Indexable Dictionaries with Applications to Encoding k-ary Trees and Multisets." In Proc. ACM-SIAM SODA, *pages 233–242, 2002.*

[8] K. Sadakane : "Space-Efficient Data Structures for Flexible Text Retrieval Systems." In Proc. ISAAC2002, *pages. 14–24. LNCS 2518, 2002.*