Short notes

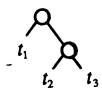
Some Properties of the Rotation Lattice of Binary Trees

We study special elements of the rotation lattice of binary ordered trees introduced in a previous paper.[3] In this lattice, we point out a Boolean sublattice. Rotation is generalised to binary unordered trees.

Received March 1988

1. Introduction

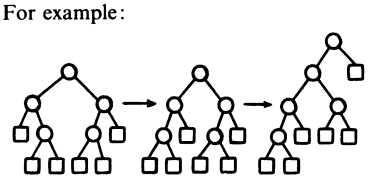
In a (rooted, ordered) binary tree, every node except the root has a parent. Every internal node ○ has a left and a right child. External nodes □ have no children. A tree is said to be of weight n if it has n external nodes. Let B_n denote the set of binary ordered trees with n internal nodes (i.e. of weight $n+1$): card $B_n = \binom{2n}{n}/(n+1)$.

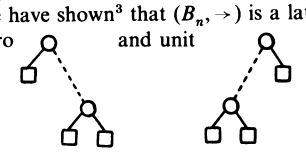
Rotation is a well-known transformation $\rightarrow$ on B_n such that a subtree  of a tree of B_n is replaced by the subtree 

Rotations maintain the symmetric order of the nodes of a tree and are used in the design of data structures.

The external nodes of a tree t are numbered by a pre-order traversal of t. Given $t \in B_n$, the weight sequence[3] of t is the integer sequence $(w_t(1), \ldots, w_t(n))$, where $w_t(i)$ is the weight of the largest subtree of t whose last external node is i.

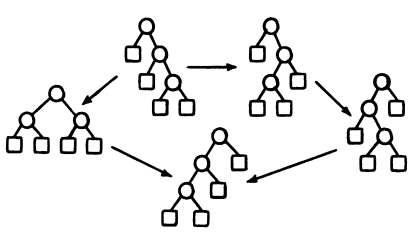
For example:



Using the fact that $t \overset{*}{\rightarrow} t'$ iff $w_t(i) \geqslant w_{t'}(i)$ for all i, we have shown[3] that $(B_n, \rightarrow)$ is a lattice with zero  and unit

We have $w_0 = (1, 1, \ldots, 1)$ and $w_1 = (1, 2, 3, \ldots, n)$. Moreover, $w_{t \wedge t'}(i) = \inf(w_t(i), w_{t'}(i))$ and $w_{t \vee t'}(i) \geqslant \sup(w_t(i), w_{t'}(i))$ for all i.

Unfortunately, this lattice is not distributive since it contains the following pentagon:



However, we give in this short note some algebraic properties of this rotation lattice using weight sequences.

2. Algebraic properties of B_n

Theorem 1

For all n, B_n is a pseudocomplemented lattice which satisfies the Stone identity.

Proof

t^* is a pseudocomplement of t iff $t \wedge t^* = 0$ and $t \wedge t' = 0$ implies that $t' \overset{*}{\rightarrow} t^*$. Thus $w_{t^*}(i) = 1$ if $w_t(i) \neq 1$ and $w_{t^*}(i) = i$ if $w_t(i) = 1$. Using weight sequences, we see that for all $t \in B_n$: $t^* \vee t^{**} = 1$.

Definition 1

Let S_n denote the subset of trees $t \in B_n$ such that $w_t(i) = 1$ or i for $i \in [1, n]$: card $S_n = 2^n - 1$.

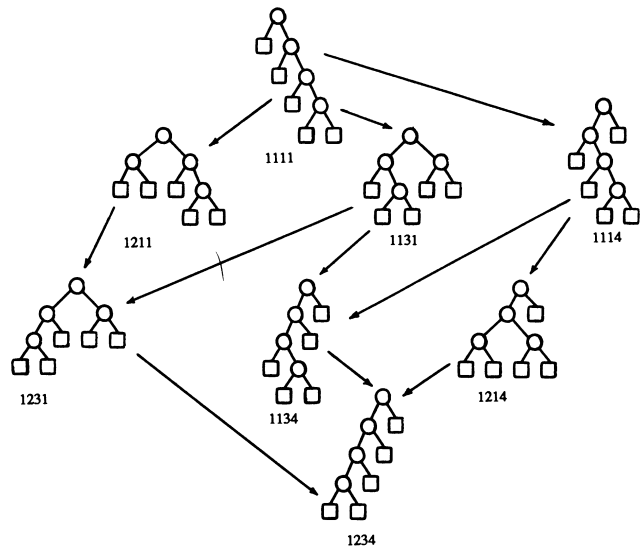
Theorem 2

The rotation ordering $\rightarrow$ of B_n partially orders S_n and makes S_n into a Boolean lattice. For t and $t' \in S_n$, we have $t \wedge t' \in S_n$ and the join in S_n is the same as the join in B_n: $t \vee t' \in S_n$.

Proof

Following Gratzer[1] (p. 49), we have $S_n = \{t^* \mid t \in B_n\}$. For t and $t' \in S_n$, $w_{t \wedge t'}(i) = \inf(w_t(i), w_{t'}(i))$ and $w_{t \vee t'}(i) = \sup(w_t(i), w_{t'}(i))$ for all $i \in [1, n]$.

S_4 diagram:



If we label the internal nodes of a tree $t \in B_n$ by the integers from 1 to n in the in-order traversal, the integer sequence which is generated by the pre-order traversal of t is called a tree permutation.[2] It is well-known that p is a tree permutation of a tree of B_n iff it contains no subsequences (p_i, p_j, p_k) such that $i < j < k$ and $p_k < p_i < p_j$. We give a similar characterisation of S_n:

Theorem 3

p is a tree permutation of a tree of S_n iff it contains neither subsequences (p_i, p_j, p_k) such that $i < j < k$ and $p_k < p_i < p_j$ nor subsequences $(p_{i'}, p_{j'}, p_{k'})$ such that $i' < j' < k'$ and $p_{i'} < p_{k'} < p_{j'}$.

Proof

By induction on n. Indeed, if p is a tree permutation of a tree of S_n then $p = (n, p')$ or $p = (p', n)$, where p' is a tree permutation of a tree of S_{n-1}.

3. Special elements of B_n

Lemma 1

S_n is the set of trees $t \in B_n$ which are distributive, i.e. $t \vee (t' \wedge t'') = (t \vee t') \wedge (t \vee t'')$ for all $t', t'' \in B_n$.

Lemma 2

Let V_n denote the set of trees $t \in B_n$ which are standard, i.e. $t' \wedge (t \vee t'') = (t' \wedge t) \vee (t' \wedge t'')$ for all $t', t'' \in B_n$.

Then $t \in V_n$ iff there exists $k \in [1, n]$ such that $w_t(i) = 1$ for $1 \leqslant i \leqslant k$ and $w_t(i) = i$ for $k+1 \leqslant i \leqslant n$. Thus card $V_n = n$.

Theorem 4[1]

$t \in S_n$ iff the binary relation $R(t)$ on B_n defined by $t' R(t) t''$ iff $t \vee t' = t \vee t''$ is a congruence relation.

Theorem 5[1]

$t \in V_n$ iff the binary relation $R'(t)$ on B_n defined by $t' R'(t) t''$ iff $(t' \wedge t'') \vee t_1 = t' \vee t''$ for some $t_1 \overset{*}{\rightarrow} t$ is a congruence relation.

4. Rotation on binary unordered trees

We can define a metric on B_n in the following way:[4] given two trees $t, t' \in B_n$. the rotation distance of t and t', denoted $d(t, t')$, is the minimum number of applications of $\rightarrow$ and $\overset{-1}{\rightarrow}$ which will transform t into t'. An algorithm for computing $d(t, t')$ is given in a previous paper.[4]

Let C_n denote the set of binary rooted unordered trees with n internal nodes. C_n is the quotient set of B_n by the equivalence relation 'isomorphism between rooted trees'.

Definition 2

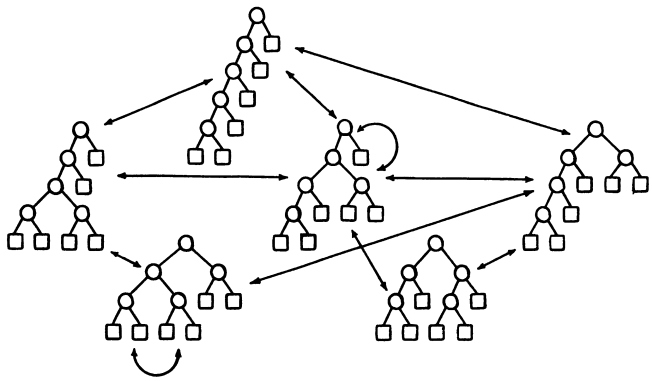
The rotation $\leftrightarrow$ in C_n is defined as follows: for

T and $T' \in C_n$: $T \leftrightarrow T'$ iff there exist $t \in T$ and $t' \in T'$ such that $t \to t'$ or $t' \to t$.

It is worth noting that $T \leftrightarrow T$ may occur. We can also define a metric on C_n.

Definition 3
Given T and $T' \in C_n$, the rotation distance of T and T', denoted $D(T, T')$, is 0 if $T = T'$ or the minimum number of applications of $\leftrightarrow$ which will transform T into T' if $T \neq T'$.

Note that $D(T, T') \leqslant d(t, t')$ for all $t \in T$ and $t' \in T'$. Unfortunately, isomorphism between rooted trees is not a congruence on the lattice B_n. C_5 diagram (equivalence classes of C_n are represented by a tree of B_n for convenience):



An open problem is to determine the complexity of computing $D(T, T')$.

J. PALLO
Département d'Informatique
Université de Bourgogne
B.P. 138
21004 Dijon
France

References

1. G. Gratzer, *General Lattice Theory*. Academic Press, New York (1978).
2. R. F. Hille, Binary trees and permutations. *The Australian Computer Journal* **17** (2), 85–87 (1985).
3. J. M. Pallo, Enumerating, ranking and unranking binary trees. *The Computer Journal* **29** (2), 171–175 (1986).
4. J. Pallo, On the rotation distance in the lattice of binary trees. *Information Processing Letters* **25**, 369–373 (1987).

A Recursive Performance Formula of the Disc Modulo Allocation Method for Binary Cartesian Product Files

In this paper an efficient recursive formula for evaluating the performance of the Disc Modulo method of allocating binary Cartesian product files to multi-disc systems is presented. It significantly improves the recently derived performance formula by the authors.

Received June 1987, revised March 1988

1. Introduction

The file allocation problem is an important and interesting aspect of database design. This problem can be defined as follows. Given a preconstructed file system, the task is to allocate all buckets of the file to a fixed number of independently accessible discs in such a way that the average response time, over all possible partial-match queries, is minimised (i.e. the concurrency of disc access is maximised). Here a file is a set of records, a bucket is a package of records in a file and a partial-match query is a request to retrieve all buckets satisfying the conditions specified by the query itself. For a file stored on $m (m \geqslant 2)$ independently accessible discs, the response time to a query is dominated by the maximum number of buckets needed to be accessed by the query on a particular disc.

Since the binary Cartesian product file (BCPF for short) has practical importance and is the commonly assumed file structure for partial-match retrieval,[6,8] in this paper we shall concentrate particularly on the N-attribute BCPF allocation problem. By an N-attribute BCPF we mean a set of N-attribute records for which each attribute domain contains only two elements, say 0 and 1, and each bucket can be uniquely identified by an N-tuple $[b_1, b_2, \ldots, b_N]$ where each $b_i = 0$ or 1.

Du and Sobolewski[7] proposed a heuristic allocation method called the Disc Modulo (DM for short) allocation method. In the DM allocation method, each bucket $[b_1, b_2, \ldots, b_N]$ is assigned to disc $(b_1 + b_2 + \ldots + b_N) \bmod m$, where m is the total number of available discs. It was shown in Ref. 7 that, under many conditions commonly occurring in practice, the DM allocation method is optimal. Here 'optimal' means that the average response time of all partial-match queries is minimal. However, it is not optimal in general. Nevertheless, in the past few years the DM allocation method has been explored by many researchers.[1-7]

Let $t_{DM}(q_n^*)$ denote the response time for a partial-match query q_n^* with n unspecified attributes when we apply the DM allocation method to assign all buckets of a BCPF to an m-disc system ($m \geqslant 2$ and discs labelled as units $0, 1, \ldots, m-1$). Chang and Chen[4] showed that

$$t_{DM}(q_n^*) = \max\Big\{ \sum_{r \bmod m = i} C_r \,\Big|\, 0 \leqslant i \leqslant m-1, \tag{1.1}$$

and C_r is the coefficient of x^r in polynomial $(x + x^2)^n$, $n \leqslant r \leqslant 2n\}$.

However, the evaluation of (1.1) will take excessive time if n becomes large. And it should be pointed out that the number of attributes of a BCPF is usually large. In the next section we shall first show that, in fact

$$t_{DM}(q_n^*) = \sum_{r \bmod m = \lfloor n/2 \rfloor \bmod m} C_r.$$

Furthermore, we shall present a very efficient recursive formula for evaluating

$$\sum_{r \bmod m = \lfloor n/2 \rfloor \bmod m} C_r.$$

Conclusions are given in Section 3.

2. A Recursive Performance Formula of the DM Allocation Method for BCPFs

From (1.1) we have

$$t_{DM}(q_n^*) = \max\Big\{ \sum_{(n+r) \bmod m = i} C_{n+r} \,\Big|\, 0 \leqslant i \leqslant m-1, \tag{2.1}$$

and C_{n+r} is the coefficient of x^{n+r} in $(x + x^2)^n$, $0 \leqslant r \leqslant n\}$.

Since $(n + r_1) \bmod m = (n + r_2) \bmod m$ if and only if $r_1 \bmod m = r_2 \bmod m$ and the coefficient of x^{n+r} in $(x + x^2)^n$ is identical to that of x^r in $(1 + x)^n$ (i.e. $C_{n+r} = \binom{n}{r}$ for $0 \leqslant r \leqslant n$), hence (2.1) can be further expressed as

$$t_{DM}(q_n^*) = \max\left\{ \sum_{r \bmod m = i} \binom{n}{r} \,\middle|\, 0 \leqslant i \leqslant m-1 \right\}. \tag{2.2}$$

Now, let us use $S_i^{(n)}$ to denote $\sum_{r \bmod m = i} \binom{n}{r}$. Then

$$t_{DM}(q_n^*) = \max\{S_0^{(n)}, S_1^{(n)}, \ldots, S_{m-1}^{(n)}\}. \tag{2.3}$$

In the following, we claim that $S_{\lfloor n/2 \rfloor \bmod m}^{(n)}$ is the largest one among $S_i^{(n)}$'s.

Lemma 2.1

$$S_i^{(n+1)} = S_i^{(n)} + S_{(i-1) \bmod m}^{(n)} \text{ for } 0 \leqslant i \leqslant m-1. \tag{2.4}$$

Proof

Note that for each $0 \leqslant i \leqslant m-1$,

$$S_i^{(n+1)} = \sum_{r \bmod m = i} \binom{n+1}{r}.$$

Since $\binom{n+1}{r} = \binom{n}{r} + \binom{n}{r-1}$, we have

$$S_i^{(n+1)} = \sum_{r \bmod m = i} \binom{n}{r} + \sum_{r \bmod m = i} \binom{n}{r-1} = S_i^{(n)} + \sum_{r \bmod m = i} \binom{n}{r-1}.$$

However,

$$\sum_{r \bmod m = i} \binom{n}{r-1} = \sum_{r' \bmod m = (i-1) \bmod m} \binom{n}{r'} = S_{(i-1) \bmod m}^{(n)},$$

where $r' = r - 1$. Therefore, we have

$$S_i^{(n+1)} = S_i^{(n)} + S_{(i-1) \bmod m}^{(n)}. \quad \text{Q.E.D.}$$