AD-A102 229 ILLINOIS UNIV AT URBANA DEPT OF COMPUTER SCIENCE F/G 12/1
GENERATION OF K-ARY TREES.(U)
1980 C L LIU N00014-79-C-0775

UNCLASSIFIED NL

1 OF 1
AD-A
102229

END
DATE
FILMED
8-23-82
NTIS

AD-A102 229

GENERATION OF K-ARY TREES

C. L. Liu

Illionois University at Urbana
Urbana, IL

1980

AD-A102229

GENERATION OF K-ARY TREES*

Dedicated to Nick Metropolis, on the occasion of his 65th birthday

C. L. LIU
Department of Computer Science
University of Illinois at Urbana-Champaign
URBANA, IL 61801

Abstract:

This paper shows new methods for random generation, ranking and unranking of k-ary trees.

* This work was partially supported by the Office of Naval Research under contract No. ONR-N00014-79-C-0775

29

1. INTRODUCTION

The problem of generating ordered trees was studied rather extensively in the recent literature [1, 2, 3, 4, 5, 6, 7]. As in all problems concerning the generation of a set of combinatorial objects, there are three major aspects of the problem. After defining a certain linear ordering over the set of combinatorial objects that we are interested in we need to (i) design an algorithm for listing the combinatorial objects one by one, (ii) design an algorithm for determining the rank (the relative position in the linear ordering) of a given combinatorial object, and (iii) design an algorithm for generating a combinatorial object when its rank is given. In most cases, there is an additional consideration, namely, to choose a suitable way to represent the combinatorial objects. (For example, the subsets of a set can be represented by subsets of integers corresponding to the elements in the set, or by 0-1 vectors, and so on.). Therefore, we are actually concerned with the listing, ranking, and unranking of a chosen representation of the combinatorial objects. Furthermore, a chosen representation might possess a natural linear ordering (for example, the lexicographical ordering of sequences of integers) which might or might not be consistent with the linear ordering over the combinatorial objects that was defined earlier. In this paper, we study listing, ranking, and unranking algorithms for k-ary trees when they are represented by permutations of multi-sets, by sequences of 0's and 1's, and by sequences of integers.

A tree is said to be rooted if there is a distinct internal node which is identified as the root. A tree is said to be regular if every internal node of the tree has the same number of sons. A tree is said to be ordered if the subtrees of each internal node are ordered and are identified as the first subtree, the second subtree, ..., and so on. (Thus, two trees T and T' are isomorphic if and only if the first subtree of T is isomorphic to the first subtree of T', the second subtree of T is isomorphic to the second subtree of T', ..., and so on). Throughout this paper, we consider the class of rooted, regular, ordered trees in which every internal node has exactly k sons . For brevity, we shall refer to these trees as k-ary trees. The <u>level</u> <u>number</u> of a node (internal node or leaf) is defined to be the length of the path (number of edges) from the root to the node.

We define a linear ordering over the set of k-ary trees , <, as follows :

<u>Definition 1</u> : Given two k-ary trees T and T',we say that $T < T'$ if

(i) T is empty (i.e. T has only a single node) and T' is not empty, or

(ii) T is not empty and for some $1 \le i \le k$, $T_j = T'_j$ for $j = 1,2,\ldots, i-1$, and $T_i < T'_i$ where $T_1, T_2, \ldots, T_k, T'_1, T'_2, \ldots, T'_k$ denote the subtrees of T and T', respectively.

Figure 1 shows an example where T and T' are ternary trees.

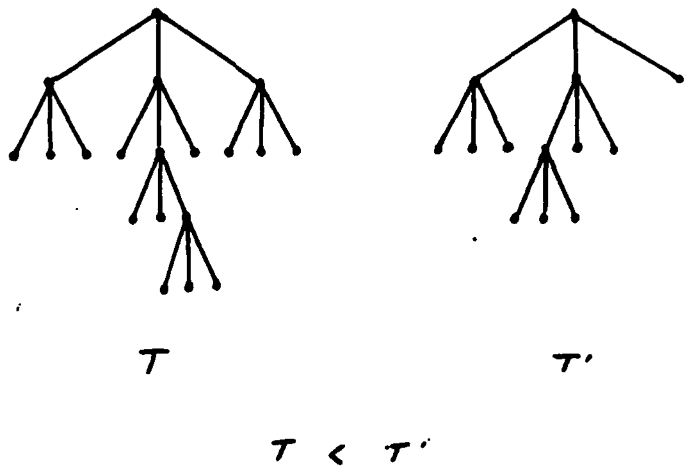


Figure 1

2. PERMUTATION REPRESENTATION OF k-ARY TREES

A binary tree with n internal nodes can be represented by a permutation of the integers 1,2,..., n [1,2,5]. The general idea is to traverse a binary tree according to a certain order and label the internal nodes in the order they are visited with the integers 1,2,..., n. The tree is then traversed again according to a different order and the labels of the internal nodes are read off in the order they are visited. We thus obtain a permutation of the integers 1,2,..., n which can be used to represent the binary tree. Knott [1] and Roten and Varol [2] studied such a representation in which we traverse and label the internal nodes in post-order (left subtree-right subtree-root) and then traverse and read off the labels of the internal nodes in preorder (root-left subtree-right subtree). Trojanowski [5] studied such a representation in which we traverse and label the internal nodes in preorder (root-left subtree-right subtree) and then traverse and read off the labels of the internal nodes in inorder (left subtree-root-right subtree). Figure 2 shows an example. For the tree in Figure 2(a), Figure 2(b) shows the labels of the internal nodes and the permutation according to the representation used by Knott. Figure 2(c) shows the labels of the internal nodes and the permutation according to the representation used by Trojanowski.

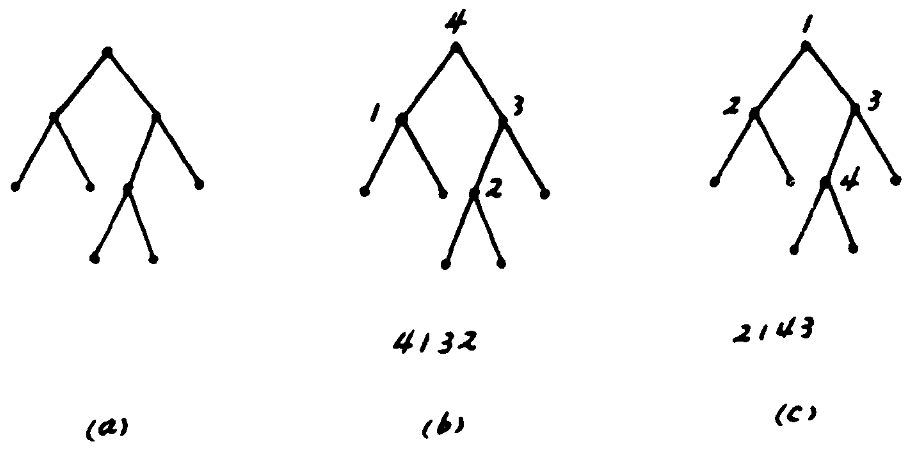


Figure 2

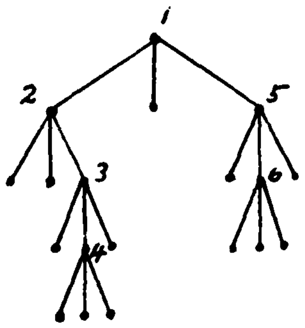
Trojanowski [5] used a similar representation for k-ary trees where a k-ary tree with n internal nodes is represented by a permutation of the integers 1,2,..., nk+1. However, such a representation does not reduce to the representation for binary trees described above when k = 2. We show now a representation of a k-ary tree with n internal nodes by a permutation of a multiset consisting of (k-1) 1's, (k-1) 2's, ..., and (k-1) n's*. Such a representation can be viewed as a natural extension of the representation for binary trees used by Trojanowski. Suppose we traverse a k-ary tree in the order root-firt subtree-second subtree - ... kth subtree+ and label the internal nodes of the tree in the order they are visited. We then traverse the tree in the order first subtree - root - second subtree - root - ... - (k-1)st subtree - root - kth subtree++ and

+ This can be viewed as an extension of the preorder traversal of binary trees.

++ This can be viewed as an extension of the inorder traversal of binary trees.

* We shall use the notation $\{1,2, \ldots, n\}^{k-1}$ to denote the multiset consisting of (k-1) 1's, (k-1) 2's, ..., and (k-1) n's.

read off the labels of the internal nodes in the order they are visited. Since the label of each internal node is read off exactly k-1 times, we obtain a permutation of the multiset $\{1,2, \ldots, n\}^{k-1}$ which we shall use as a representation of the k-ary tree. Figure 3 shows an example. We shall refer to such a permutation as the permutation of a k-ary tree. We shall also refer to such a representation as the permutation representation of k-ary trees. For a k-ary tree T, we use $\underline{p}(T)$ to denote the permutation of T and $p_1, p_2, \ldots, p_{(k-1)n}$ to denote the elements of $\underline{p}(T)$.



$\underline{p}(T) = 223443115665$

figure 3

We have the following definition :

Definition 2 : A permutation $\underline{\pi}$ of the multiset $\{1,2,\ldots, n\}^{k-1}$ is called a tree permutation if and only if the following conditions are satisfied :

(i) The (k-1) n's occupy consecutive positions in $\underline{\pi}$.

(ii) The (k-1) n's appear either to the right of the leftmost n-1 or to the immediate left of the leftmost n-1.

(iii) The permutation obtained from $\underline{\pi}$ by deleting the (k-1) n's is a tree permutation of the multiset $\{1,2,\ldots, n-1\}^{k-1}$.

We have now :

Theorem 1 : The permutation of a k-ary tree with n internal nodes is a tree permutation of the multiset $\{1,2,\ldots, n\}^{k-1}$.

Proof : Let T be a k-ary tree and x be the internal node that is labelled n. We note first that x can only have k leaves as its subtrees. (Because x is the last internal node visited in the root - first subtree - second subtree - ... - kth subtree traversal). Thus, in the permutation $\underline{p}(T)$, the (k-1) n's must occupy consecutive positions.

Let y denote the father of x. Either y is labelled n-1 (as illustrated in Figure 4(a) or the internal node labelled n-1 is in a subtree of y that is to the left of the subtree that contains x (as illustrated in figure 4(b)). For the former case, if x is the root of the first subtree of y, the (k-1) n's appear to the immediate left of the leftmost n-1 in $\underline{p}(T)$. Otherwise, the (k-1) n's appear to the right of the leftmost n-1. For the latter case, the (k-1) n's always appear to the right of the leftmost n-1 in $\underline{p}(T)$.

Finally, because the removal of the k leaves of x from T yields a k-ary tree with n - 1 internal nodes, condition (iii) in Definition 2 follows immediately.

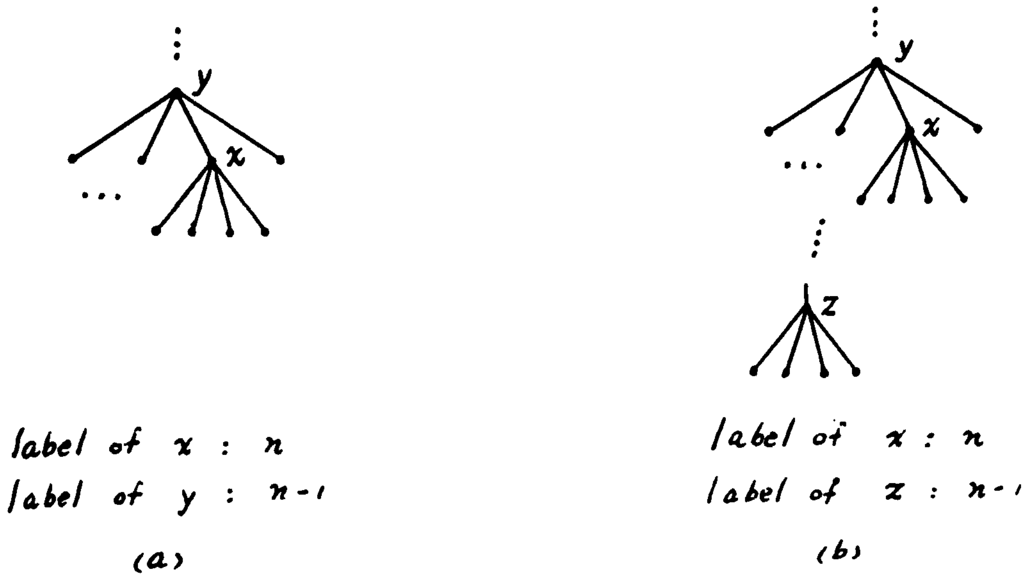


Figure 4

Theorem 2 : There is a 1-1 correspondance between the set of all k-ary trees with n internal nodes and the set of tree permutations of the multiset $\{1,2,\ldots, n\}^{k-1}$.

Proof : We show first that for two distinct k-ary trees T and T' $\underline{p}(T)$ and $\underline{p}(T')$ must be different. The proof is carried out by induction on n. For n = 1,2, the statement is obviously true. Let T and T' be two k-ary trees with n internal nodes. Let $T_1, T_2, \ldots T_k$ denote the k subtrees of T and $T'_1, T'_2, \ldots, T'_k$ denote the k subtrees of T'. Let us assume that $T_1 = T'_1$, $T_2 = T'_2$, $\ldots$, $T_{i-1} = T'_{i-1}$ and $T_i \neq T'_i$. Thus, $\underline{p}(T)$ will be of the form

$$\underline{p}(T_1)\ 1\ \underline{p}(T_2)\ 1\ \ldots\ \underline{p}(T_{i-1})\ 1\ \underline{p}(T_i)\ 1\ \ldots$$

and $\underline{p}(T')$ will be of the form

$$\underline{p}(T'_1)\ 1\ \underline{p}(T'_2)\ 1 \ldots \underline{p}(T'_{i-1})\ 1\ \underline{p}(T'_i)\ 1 \ldots$$

where $\underline{p}(T_1)$, $\underline{p}(T_2)$, ... , $\underline{p}(T_i)$, $\underline{p}(T'_1)$, $\underline{p}(T'_2)$, ..., $\underline{p}(T'_i)$ denote the permutations of the subtrees T_1, T_2, ..., T_i, T'_1, T'_2, ..., T'_i. We note that $\underline{p}(T_1) = \underline{p}(T'_1)$, $\underline{p}(T_2) = \underline{p}(T'_2)$, ..., $\underline{p}(T_{i-1}) = \underline{p}(T'_{i-1})$.

If T_i and T'_i contain the same number of internal nodes, by the induction hypothesis, $\underline{p}(T_i) \neq \underline{p}(T'_i)$. If T_i and T'_i contain different number of internal nodes, i must be less than k, and in $\underline{p}(T)$ and $\underline{p}(T')$ $\underline{p}(T_i)$ and $\underline{p}(T'_i)$ must be followed by a 1. Since $\underline{p}(T_i)$ and $\underline{p}(T'_i)$ do not contain the integer 1, it follows that $\underline{p}(T) \neq \underline{p}(T')$.

We show now that given a tree permutation $\underline{\pi}$ of the multiset $\{1, 2, ..., n\}^{k-1}$ there is a corresponding k-ary tree T such that the permutation of T is equal to $\underline{\pi}$. The proof is carried out by induction on n. Clearly, our claim is valid for n=1,2. For n > 2, let $\mathcal{R}(\underline{\pi})$ denote the permutation obtained from $\underline{\pi}$ by removing the k-1 n's in $\underline{\pi}$. By induction hypothesis, there is a k-ary tree $\mathcal{R}(T)$ such that the permutation of $\mathcal{R}(T)$ is equal to $\mathcal{R}(\underline{\pi})$. We examine two cases:

1. The k-1 n's occupy the rightmost k-1 positions in $\underline{\pi}$. Let c denote the integer at the immediate left of the k-1 n's. Clearly, the kth subtree of the internal node labelled c in $\mathcal{R}(T)$ is a leaf. Thus, we can append a subtree with k leaves to this leaf to obtain a tree T.

2. The k-1 n's do not occupy the rightmost k-1 positions in $\underline{\pi}$. Let C denote the integer at the immediate left and d denote the integer at the immediate right of the k-1 n's. Suppose c > d. In this case, all the appearances of c in $\underline{\pi}$ must be to the left of the k-1 n's. Consequently, the kth son of the internal node labelled c in $\mathcal{R}(T)$ is a leaf, and we can append a subtree with k leaves to this leaf to obtain a tree T. Suppose $c \leq d$. If d is the ith leftmost appearance of the integer d in $\underline{\pi}$, then the ith subtree of the internal node labelled d in $\mathcal{R}(T)$ is a leaf. Again, we can append a subtree with k leaves to this leaf to obtain a tree T.

In both cases, the permutation of the tree T is equal to $\underline{\pi}$.

<u>Theorem 3</u> : The lexicographical ordering of all the tree permutations of the multiset $\{1,2,\ldots,n\}^{k-1}$ is consistent with the linear ordering of k-ary trees in Definition [illegible].

<u>Proof</u> : By induction on n.

We present now an algorithm for listing all the tree permutations of the multiset $\{1,2,\ldots,n\}^{k-1}$. It is slightly simpler to have an algorithm that lists all the permutations in the reverse-lexicographical order. Thus, the first permutation in the listing will be nnn..(n-1)(n-1) ... 222..111.., and the last permutation will be 111...222...(n-1)(n-1)(n-1)...nnn,...

<u>Listing Algorithm</u> :

1. Initially, let $\underline{\pi}$ = nnn...(n-1)(n-1)(n-1) ... 222...111...

2. If the (k-1) n's are not in the rightmost k-1 positions of $\underline{\pi}$, i.e. $\underline{\pi}$ =... nnn s_i..., change $\underline{\pi}$ to ... s_i nnn ...

3. If the (k-1) n's are in the rightmost k-1 positions of $\underline{\pi}$, and the (k-1) (n-1)'s are in the rightmost k-1 positions in front of the n's, ..., search for the largest m such that not all the (k-1) m's are in these extreme positions, i.e. $\underline{\pi}$ = ... mmm s_i ..., where $s_i < m$, and change $\underline{\pi}$ to ...s_i nnn ... (n-1)(n-1)(n-1) ... (n-2)(n-2)(n-2) ...(m+1)(m+1)(m+1) ... mmm ...
4. Stop when $\underline{\pi}$ = 111...222 ... (n-1)(n-1)(n-1)...nnn...

3. RANKING ALGORITHM

We show in this section an algorithm for determining the position (rank) of a given k-ary tree in the linear ordering of k-ary trees according to Definition 1. We show first a reduction scheme which is the basis of our ranking algorithm. Given a k-ary tree with n internal nodes, we remove the leftmost* k-1 leaves of the tree and then reconstruct a k-ary tree with n-1 internal nodes as follows: Let y be the leftmost son of x in a (not necessarily k-ary) tree. If y has fewer than k sons, remove y and let the sons of y become leftmost sons of x as illustrated in Figure 5(a). If y has more than k sons, say m. Let the rightmost m-k sons of y become the second, third, ..., $(m-k)^{th}$, $(m-k+1)^{st}$ leftmost sons of x as illustrated in Figure 5(b). (In the examples in Figure 5, k = 3). Now, suppose that the leftmost k-1 leaves have been removed from a k-ary tree. If these k-1 leaves are at the same level, we apply the reduction step described above to the father of these k-1 leaves. Otherwise, let x_1 denote the internal node which is the leftmost brother of the first leaf that was removed. Let x_i denote the father of the $(k-1)^{st}$ leaf that was removed. Since there is a path between x_1 and x_i, let $x_1, x_2, x_3, \ldots, x_{i-1}, x_i$ be the sequence of nodes on this path. Applying the reduction step described above to x_i, and then to x_{i-1}, and then to x_{i-2}, . . ., and finally to x_1, we shall obtain a k-ary tree with n-1 internal nodes. Let T be a k-ary tree. We shall use $\mathcal{R}(T)$ to denote the k-ary tree obtained by the reduction procedure described above. Figure 6 shows an example.

Let

$$N(x,y,z) = \frac{ky-y-z}{kx-y-z}\binom{kx-y-z}{x-y}$$

we have:

Theorem 4: Let $\mathcal{J}$ be the set of all k-ary trees with n internal nodes and with the level numbers of the leftmost k-1 leaves being $a_1, a_2, \ldots, a_{k-1}$.** Let $\mathcal{R}(\mathcal{J})$ denote the set of k-ary trees obtained by the reduction procedure described above. Then :

*The left to right order of the leaves of a tree is defined in the obvious way.

**Clearly, $a_1 \le a_2 \le \ldots \le a_{k-1}$.

(i) $\mathcal{R}(\mathcal{T})$ is the set of all k-ary trees with n-1 internal nodes and with the level number of the leftmost leaf being $a_{k-1}-1$ or larger.

(ii) There is 1-1 correspondence between the trees in $\mathcal{T}$ and the trees in $\mathcal{R}_0(\mathcal{T})$.

(iii) For T and T' in $\mathcal{T}$, if T T', then $\mathcal{R}_0(T) < \mathcal{R}_0(T')$.

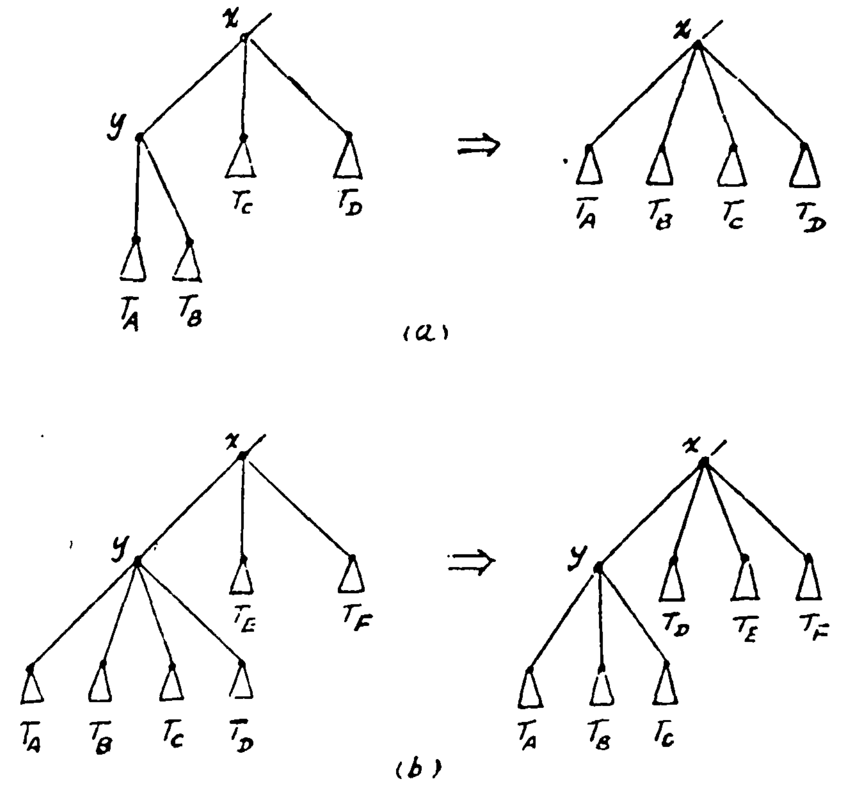


Figure 5

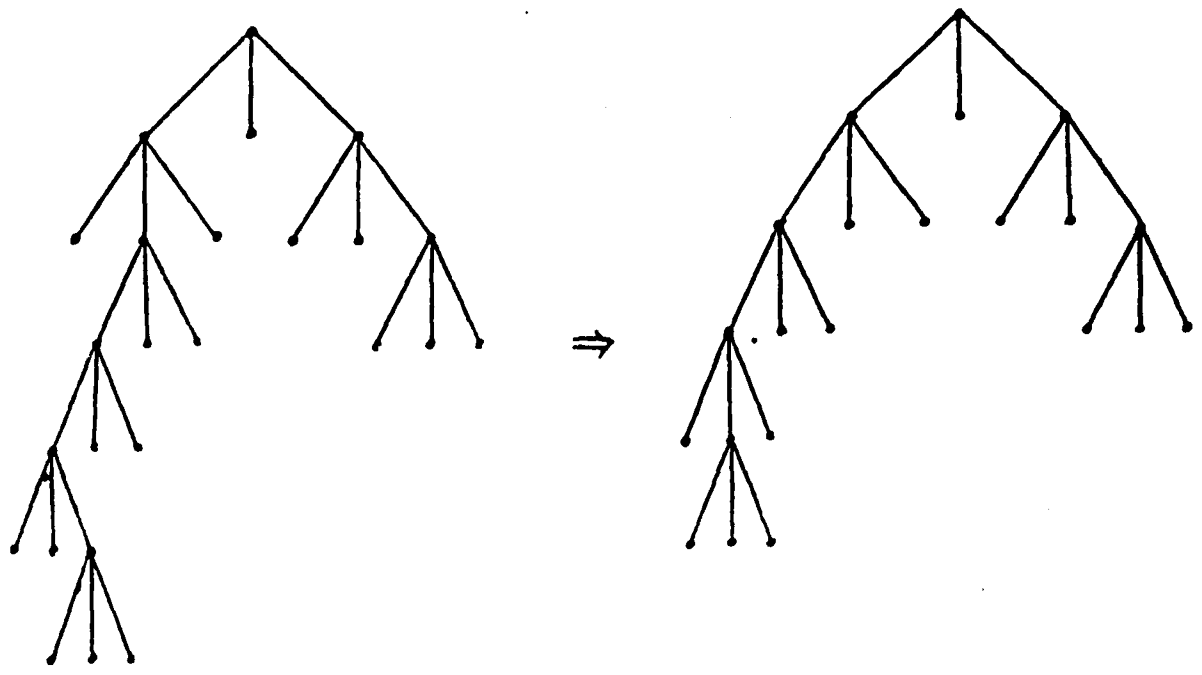


Figure 6

<u>Proof</u> : Let T be a k-ary tree with n internal nodes and with the level numbers of its leftmost k-1 leaves being $a_1, a_2, \ldots, a_{k-1}$.

We note that the level number of the immediate right brother of the $(k-1)^{st}$ leaf (which is either an internal node or a leaf) is also a_{k-1}.

Thus, after the removal of the leftmost k-1 leaves of T, the level number of the leftmost leaf of the remaining tree (which is no longer a k-ary tree) is at least a_{k-1}. According to the reduction procedure, the level number of the leftmost leaf of $\mathcal{R}(T)$ is at least $a_{k-1}-1$.

It is also easy to see that any k-ary tree with n-1 internal nodes and with the level number of its leftmost leaf being $a_{k-1}-1$ or larger can be obtained from a k-ary tree with n internal nodes and with the level numbers of its leftmost k-1 leaves being a_1, a_2, ..., a_{k-1}.

The 1-1 correspondence between the trees in $\mathcal{T}$ and the trees in $\mathcal{R}(\mathcal{T})$ can now be established by showing that $|\mathcal{T}|=|\mathcal{R}(\mathcal{T})|$. It is well-known that the number of k-ary trees with n internal nodes and with the level numbers of its leftmost k-1 leaves being a_1, a_2, ..., a_{k-1} is

$$N(n,\ a_{k-1},\ k-3)\quad -\quad (n,\ a_{k-1}+1,\ k-3) \qquad (*)$$

and the number of the k-ary trees with n-1 internal nodes with the level number of its leftmost leaf is $a_{k-1}-1$ or larger is

$$N(n,\ a_{k-1},\ k-2) \qquad (**)$$

It is a straight forward computation to show that (*) is equal to (**).

Finally, because the reduction procedure does not change the relative positions of the subtrees, thus if $T < T'$, then $\mathcal{R}(T) < \mathcal{R}(T')$.

<u>Theorem 5</u> : Let T be a k-ary tree with n internal nodes and with the level numbers of its leftmost k-1 leaves being a_1, a_2,..., a_{k-1}. Let Index (T) denote the number of k-ary trees with n internal nodes that are larger than T in the linear ordering in Definition 1. Then

$$\text{Index}(T) = N(n+1,\ a_1+2,\ k-2) + \sum_{i=2}^{k-1} N(n,\ a_i+1,\ i-2) + \text{Index}\ (\mathcal{R}(T))$$

Proof : Index (T) is equal to the number of k-ary trees with the level numbers of its leftmost k-1 leaves being $a_1, a_2, \ldots, a_{k-1}$ that are larger than T in the linear ordering, plus the number of trees with the level numbers of its leftmost k-2 leaves being $a_1, a_2, \ldots, a_{k-2}$ and the level number of its $(k-1)^{st}$ leaf being $a_{k-1}+1$ or larger, plus the number of trees with the level number of its leftmost k-3 leaves being $a_1, a_2, \ldots, a_{k-3}$, and the level number of its $(k-2)^{nd}$ leaf being $a_{k-2}+1$ or larger, ..., plus the number of trees with the level number of its leftmost leaf being a_1+1 or larger. According to theorem 4 the number of trees with the level numbers of its leftmost k-1 leaves being $a_1, a_2, \ldots, a_{k-1}$ that are larger than T in the linear ordering is equal to Index $(\mathcal{R}(T))$.
The number of trees with the level numbers of its leftmost k-2 leaves being $a_1, a_2, \ldots, a_{k-2}$ and the level number of its $(k-1)^{st}$ leaf being $a_{k-1}+1$ orlarger is $N(n, a_{k-1}+1, k-3)$*, the number of trees with the level numbers of its leftmost k-3 leaves being $a_1, a_2, \ldots, a_{k-3}$ and the level number of its $(k-2)^{nd}$ leaf being $a_{k-2}+1$ orlarger is $N(n, a_{k-2}+1, k-4)$, ..., the number of trees with the level number of its leftmost leaf being a_1 and the level number of the second leaf being a_2+1or larger is $N(n, a_2+1, 0)$, and the number of trees with the level number of its leftmost leaf being a_1+1 orlarger is $N(n+1, a_1+2, k-2)$.

Theorem 6 : Let $(p_1, p_2, \ldots, p_{(k-1)n})$ be the permutation of T. Then $p_1, p_2, \ldots, p_{k-1}$ are the level numbers of the leftmost k-1 leaves of T.

Proof : Let $x_1, x_2, \ldots, x_{k-1}$ denote the fathers of the leftmost k-1 leaves of the T. There is a path from the root to x_1, to x_2, ..., to x_{k-1} as shown in Figure 7. In a root-first subtree- second subtree - ... - k^{th} subtree traversal of T, $x_1, x_2, \ldots, x_{k-1}$ will be labelled exactly with the level numbers of the k-1 leftmost leaves of T. Furthermore, in a first subtree - root - second subtree - root - ... - k^{th} subtree traversal of T, these level numbers will be the first k-1 labels that are read off.

* Note that the number of such trees is independent of the values of $a_1, a_2, \ldots, a_{k-2}$

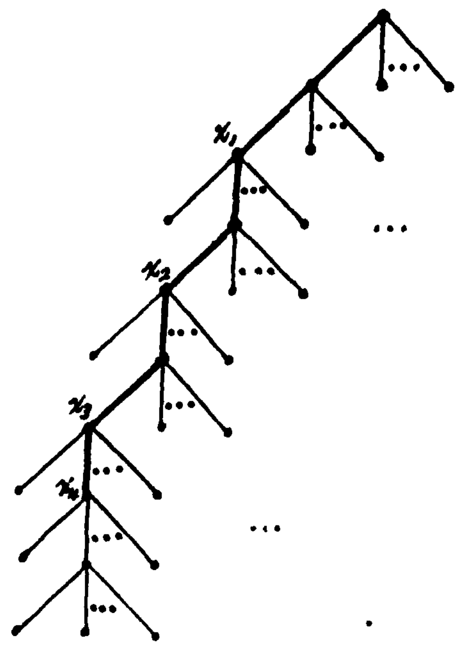


Figure 7

Theorem 7 : Let $(p_1, p_2, \ldots, p_{(k-1)n})$ be the permutation of T. The permutation of $\mathcal{R}(T)$ can be obtained as follows :

(i) Remove $p_1, p_2, \ldots, p_{k-1}$.

(ii) The remaining p's retain their relative positions. However, their values are changed as follows : for $1 \le p \le n$

$p \le p_1$	$p_1 < p \le p_2$	$p_2 < p \le p_3$	
p is unchanged	The rightmost p becomes p-1, the others unchanged	The rightmost two p's become p-1, the others unchanged	...

	$p_{k-3} < p \leq p_{k-2}$	$p_{k-2} < p \leq p_{k-1}$	$p_{k-1} < p$
...	The rightmost k-3 p's become p-1, the others unchanged	The rightmost k-2 p's become p-1, the others unchanged	all p's become p-1

Proof : To simplify the notations, we prove the theorem for the case k = 3. In this case, the theorem can be stated as : Let $(p_1, p_2, \ldots p_{2n})$ be the permutation of a ternary tree T with n internal nodes. The permutation of $\mathcal{R}$(T) can be obtained as follows :

(i) Remove p_1 and p_2.

(ii) The remaining p's retain their relative positions with the following changes in value :

$p \leq p_1$	$p_1 < p \leq p_2$	$p_2 < p$
p is unchanged	the right p is changed to p-1 the left p remains unchanged	both p's are changed to (p-1)'s

Figure 8 shows T and $\mathcal{R}$(T) where the subtrees are identified by the letters A, B, C, ..., L, M. We note first that in traversing T and labelling the internal nodes in the root-first subtree - second subtree - third subtree order, after labelling the internal nodes with 1, 2, ... p_1, ... p_2 as shown, the subtrees A, B, C, ..., L, M will be traversed and their internal nodes labelled in that order. For the tree $\mathcal{R}$(T), after labelling the internal nodes with 1, 2, ..., p_1, ..., p_2-1 as shown, the subtrees A, B, C, ..., L, M will also be traversed and their internal nodes labelled in that order. Now, when we traverse the tree T in the first subtree - root - second subtree - root - third subtree order, we read off the labels as :

$$p_1\ p_2\mathcal{L}(A)\ p_2\mathcal{L}(B)\ (p_2-1)\mathcal{L}(C)\ (p_2-1)\mathcal{L}(D)\ \ldots\ (p_1+2)$$

$$\mathcal{L}(E)\ (p_1+2)\mathcal{L}(F)\ (p_1+1)\mathcal{L}(G)\ (p_1+1)\mathcal{L}(H)\ p_1\ \mathcal{L}(I)\ \ldots$$

where we use $\mathcal{L}(A), \mathcal{L}(B), \ldots \mathcal{L}(H), \mathcal{L}(I)$ to denote the sequences of labels we read off from the subtrees A, B, ..., H, I. Note that the labels in $\mathcal{L}(A), \mathcal{L}(B), \ldots \mathcal{L}(H), \mathcal{L}(I)$ are all larger than p_2. When we traverse the tree $\mathcal{R}(T)$ in the same manner, we read off the labels as

$$\mathcal{L}'(A)\ (p_2-1)\mathcal{L}'(B)\ (p_2-1)\mathcal{L}'(C)\ (p_2-2)\mathcal{L}'(D)\ldots\ (p_1+2)$$

$$\mathcal{L}'(E)\ (p_1+1)\mathcal{L}'(F)\ (p_1+1)\mathcal{L}'(G)\ p_1\mathcal{L}'(H)\ p_1\mathcal{L}'(I)\ \ldots$$

where we use $\mathcal{L}'(A), \mathcal{L}'(B), \ldots, \mathcal{L}'(H), \mathcal{L}'(I)$ to denote the sequences of labels we read off the subtrees A, B, ..., H, I. Note that $\mathcal{L}'(A)$ is exactly equal to $\mathcal{L}(A)$ with each of the labels in $\mathcal{L}(A)$ reduced exactly by 1, and so on. Comparing the two sequences of labels, we note that the theorem follows immediately.

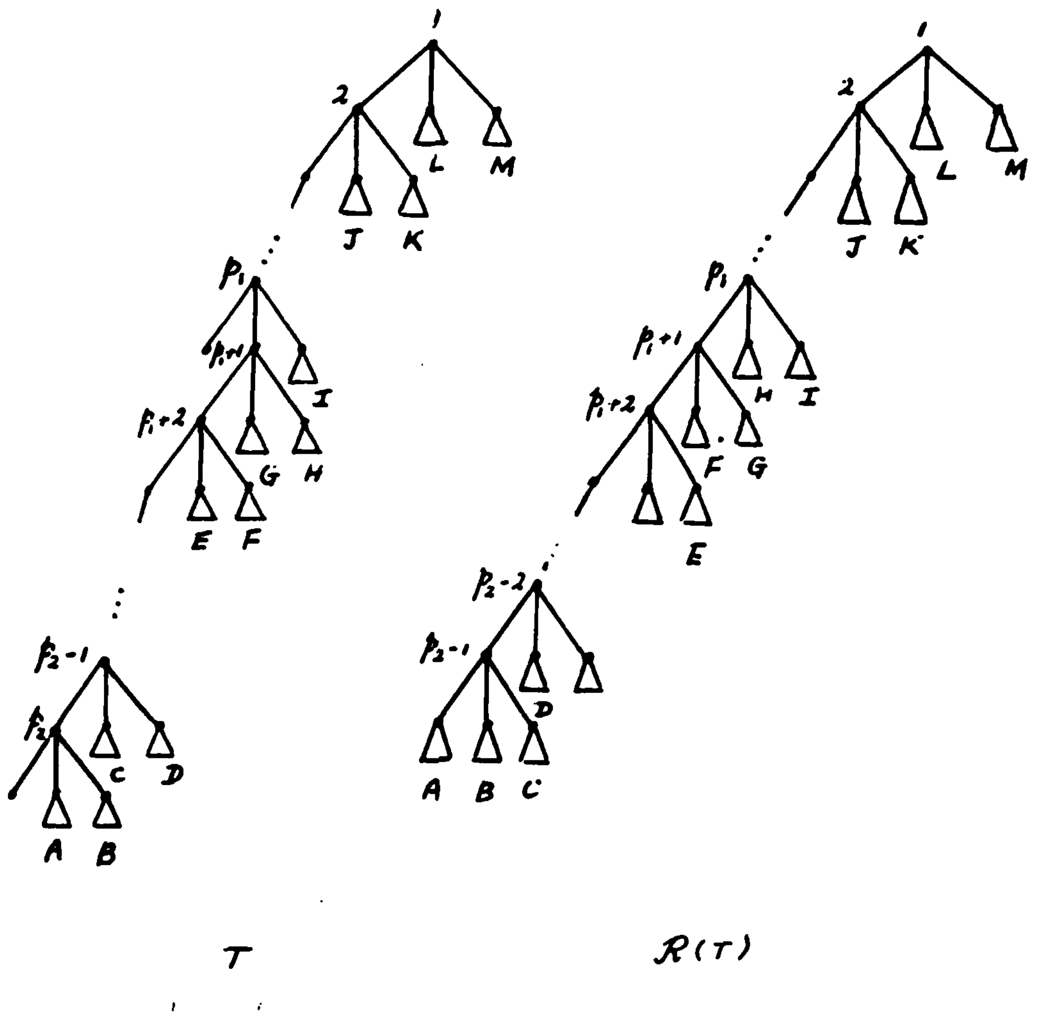

1
2
L
M
J
K
p_1
p_1+1
I
p_1+2
G
H
E
F
p_2-1
p_2
C
D
A
B
T
1
2
L
M
J
K
p_1
p_1+1
H
I
p_1+2
F
G
E
p_2-2
p_2-1
D
A
B
C
$\mathcal{R}(T)$

Figure 8

With the results in Theorems 6 and 7, Theorem 5 provides immediately a recursive algorithm for determing the rank of a tree T when the permutation $\underline{p}(t)$ is given. We show an illustrative example : Given the permutation of a tree to be 2234431155, we want to determine its rank. We have

$$\begin{aligned}
\text{index}(2234431155) &= N(6,4,1) + N(5,3,0) + \text{index}(23321144)\\
&= 42 + 33 + \text{index}(23321144)
\end{aligned}$$

$$\begin{aligned}
\text{index}(23321144) &= N(5,4,1) + N(4,4,0) + \text{index}(221133)\\
&= 7 + 1 + \text{index}(221133)
\end{aligned}$$

$$\begin{aligned}
\text{index}(221133) &= N(4,4,1) + N(3,3,0)\\
&= 1 + 1 + \text{index}(1122)
\end{aligned}$$

$$\begin{aligned}
\text{index}(1122) &= N(3,3,1) + N(2,2,0) + \text{index}(11)\\
&= 1 + 1 + 0
\end{aligned}$$

Thus

$$\text{index}(2234431155) = 42 + 33 + 7 + 1 + 1 + 1 + 1 + 1 = 87$$

Since there are 273 ternary trees with 5 internal nodes. The given tree is the 186th tree (273 - 87 = 186) according to the linear ordering in Definition 1.

4. UNRANKING ALGORITHM

We show now an unranking algorithm which is the reverse of the ranking algorithm. Given index(T), the unranking algorithm will determine $\underline{p}(T)$ based on the following results :

Theorem 8 : Given index(T), the first k-1 elements of $\underline{p}(T)$, $p_1, p_2, \ldots, p_{k-1}$, can be determined as follows :

(1) p_1 is determined to be the smallest integer j such that index(T) > N(n+1, j+2, k-2)

(2) p_2 is determined to be the smallest integer j such that $\text{index}(T) - N(n+1,\ p_1+2,\ k-2) > N(n, j+1, o)$.

(3) $p_3, p_4, \ldots, p_{k-1}$ is determined recursively to be the smallest integer j such that

$$\text{index}(T) - N(n+1,\ p_1+2,\ k-2) - \sum_{i=2}^{m-1} N(n,\ p_i+1,\ i-2)$$

$$> N(n,\ j+1,\ m-2)$$

for $m = 3, 4, \ldots, k-1$.

Furthermore, $\text{index}(\mathcal{G}(T))$ can be computed as

$$\text{index}(T) - N(n+1,\ p_1+2,\ k-2) - \sum_{i=2}^{k-1} N(n,\ p_i+1,\ i-2) .$$

<u>Proof</u> : The proof of this theorem follows directly from Theorem 5.

<u>Theorem 9</u> : Let $\underline{p}(\mathcal{G}(T))$ denote the permutation of $\mathcal{G}(T)$. If the first k-1 elements of $\underline{p}(T)$ are known to be $p_1, p_2, \ldots, p_{k-1}$, then $\underline{p}(T)$ can be determined as follows :

(1) Place $p_1, p_2, \ldots, p_{k-1}$ in front of the elements in $\underline{p}(\mathcal{G}(T))$.

(2) The elements in $\underline{p}(\mathcal{G}(T))$ retain their relative positions. However, their values are changed as follows : for $1 \le p \le (n-1)$.

$p < p_1$	$p_1 \le p < p_2$	$p_2 \le p < p_3$	
p is unchanged	The leftmost p becomes p+1, the others unchanged	The leftmost two p's become p+1, the others unchanged	...

	$p_{k-1} < p < p_{k-2}$	$p_{k-2} \le p < p_{k-1}$	$p_{k-1} \le p$
...	the leftmost k-3 p's become p+1, the others unchanged	the leftmost k-2 p's become p+1, the others unchanged	all p's become p+1

<u>Proof</u> : The proof of this theorem follows almost directly from Theorem 7. However, we should point out why the word "rightmost" in Theorem 7 is changed to "leftmost" in this Theorem, a fact that is not obvious. Note that the sequence obtained from $\underline{p}(T)$ by removing the first k-1 elements p_1, p_2, ..., p_{k-1} has the property that all p_{k-1}'s appear to the left of all $(p_{k-1}-1)$'s which appear to the left of all $(p_{k-1}-2)$'s which appear to the left of all $(p_{k-1}-3)$'s and so on. Thus, the operation which reverses the operation in Theorem 7 is that stated in this Theorem.

As an example, suppose we are given the index of a ternary tree with 5 internal nodes to be 87

Since

$N(6,4,1)=42$ $\quad$ $N(6,3,1)=130$

we have

$p_1=2$

Since

$87- N(6,4,1)=45$

$N(5,3,0)=33$ $\quad$ $N(5,2,0)=88$

we have

$p_2=2$

we have now

$\text{index}(\mathcal{R}_o(T))=87-N(6,4,1)-N(5,3,0)=12$

Since

$N(5,4,1)=7$ $\quad$ $N(5,3,1)=25$

we have

$p_1'=2$

Since

$12-N(5,4,1)=5$

$N(4,4,0)=1$ $\quad$ $N(4,3,0)=6$

we have

$p_2'=3$

We have now

$\text{index}\ \mathcal{R}_o(\mathcal{R}_o(T)))=12-N(5,4,1)-N(4,4,0)=4$

Since

$N(4,4,1) = 1 \qquad N(4,3,1) = 5$

we have

$P''_1 = 2.$

since $4 - N(4,4,1) = 3$

we have $N(3,3,0) = 1 \qquad N(3,2,0) = 4$

$P''_2 = 2$

we have now $\text{index}(\mathcal{A}(\mathcal{A}(\mathcal{A}(T)))) = 4 - N(4,4,1) - N(3,3,0) = 2$

Since $N(3,3,1) = 1 \qquad N(3,2,1) = 3$

we have $p'''_1 = 1.$

Since $2 - N(3,3,1) = 1$

we have $N(2,2,0) = 1 \qquad N(2,1,0) = 2$

$p'''_2 = 1.$

we have now $\text{i dex}(\mathcal{A}(\mathcal{A}(\mathcal{A}(\mathcal{A}(T))))) = 4 - N(3,3,1) - N(2,2,0) = 0$

Now, we can assemble the permutation of T as follows :

the permutation of $\mathcal{A}(\mathcal{A}(\mathcal{A}(\mathcal{A}(T)))))$	is (11)
the permutation of $\mathcal{A}(\mathcal{A}(\mathcal{A}(T))))$	is (1122)
the permutation of $\mathcal{A}(\mathcal{A}(T)))$	is (221133)
the permutation of $\mathcal{A}(T)$	is (23321144)
the permutation of T	it (2234431155)

5. THE 0-1 REPRESENTATION OF TREES

It is well-known that a k-ary tree with n internal nodes can be represented by a sequence of n 1's and n(k-1)+1 0's. If we label an internal node with a 1 and a leaf with a 0 and traverse the tree in the order of root-first subtree-second subtree - ... - k^{th} subtree, we obtain a sequence of n 1's and n(k-1)+1 0's.

We shall refer to such a representation of a tree as the 0-1 representation. Figure 9 shows an example. It can be shown that:

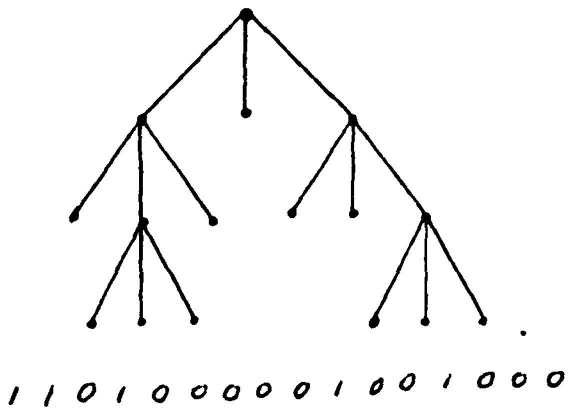


Figure 9

Theorem 10 : The lexicographical ordering of the 0-1 representations of k-ary trees is consistent with the linear ordering of k-ary trees in Definition 1.

The ranking and unranking algorithm presented in Sections 3 and 4 are particularly suitable when the 0-1 sequence representation is used. We only need to note the followings :

Theorem 11 : In the 0-1 representation of k-ary trees, for $i = 1, 2, \ldots, k-1$, the level number of the i^{th} leftmost leaf is equal to the number of 1's in front of the i^{th} 0.

Theorem 12 :

(i) The 0-1 representation of $\mathcal{R}(T)$ can be obtained from that of T by deleting the first k-1 0's and the rightmost 1 in front of the $(k-1)^{st}$ 0.*

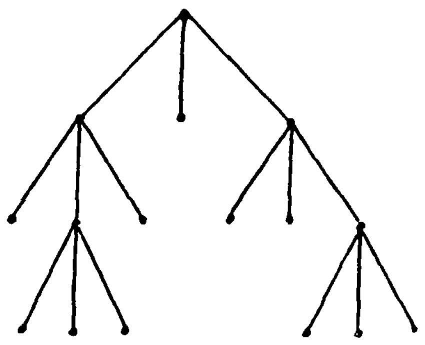
(ii) The 0-1 representation of T can be obtained from that of $\mathcal{R}(T)$, when the values of $a_1, a_2, \ldots, a_{k-1}$ are given, by adding in front of the 0-1 sequence representation of $\mathcal{R}(T)$ a sequence consisting of k-1 0's and a suitable number of 1's such that a_i is equal to the number of 1's in front of the i^{th} 0 for $i = 1, 2, \ldots, k-1$.

Indeed, the ranking and unranking algorithm described here is a natural extension of that for binary tree due to Zaks [6].

6. THE LEVEL NUMBER REPRESENTATION OF TREES

As was pointed out above, the level number of a leaf is defined to be the length of the path (number of edges) from the root to the leaf. By the level number representation of a k-ary tree we mean the sequence of level numbers of the leaves reading from left to right. In [3] and [4] listing ,ranking, and unranking algorithm for k-ary trees were studied, where the level number representation of k-ary trees was employed. Figure 10 shows an example. It is known that :

* Actually, removing any of the 1's in front of the $(k-1)^{st}$ 0 will do. However, the rightmost 1 in front of the $(k-1)^{st}$ 0 corresponds to the internal node that was removed in the reduction procedure in Section 3.



2333 21 22 333

Figure 10

Theorem 13 : The lexicographical ordering of the level number representation of k-ary trees is consistent with the linear ordering of k-ary trees in Definition 1.

It is quite clear that our ranking and unranking algorithm can be applied to the case when the level number representation of k-ary trees is used. To obtain the level number representation of $\mathcal{G}$(T) from the level number representation of T, we need to identify the leaves of those subtrees that are moved up in the reduction scheme. Conversely, in the unranking algorithm, we also need to identify the leaves of those subtrees that are moved down to recover T from $\mathcal{G}$(T). Since such a step is conceptually very simple, we shall leave the detailed description as the standard " exercise for the reader". (Hint: The subtrees can be identified by carrying out a sequence of "left reductions". By a left reduction we mean starting from the left of the level number representation, replace the first occurrence of k consecutive level number qq...q by q-1.)

7. REMARKS

We show in this paper :

(1) A new generalization of the permutation representation of binary trees studied in [5]. Our ranking and unranking algorithm are also different from that in [5].

(2) A generalization of the ranking and unranking algorithm for binary trees studied in [6] when the 0-1 representation is used.

(3) Our ranking and unranking algorithms for k-any trees when the level number representation is used are also different from that in [3, 4].

The author wishes to thank Mr. P. Ramanan for many helpful suggestions.

REFERENCES

[1] KNOTT, G.D. A numbering system for binary trees. Comm. ACM, 20(1977), 113-115

[2] ROTEN, D., VAROL, Y.L. Generating binary trees from ballot sequences. J. ACM, 25(1978), 396-404.

[3] RUSKEY, F. Generating t-ary trees lexicographically, SIAM J. Comput., 7(1978), 424-439.

[4] RUSKEY, F., HU, T.C. Generating binary trees lexicographically. SIAM J. Comput., 6(1977), 754-758.

[5] TROJANOWSKI, A.E. Ranking and listing algorithms for k-ary trees. SIAM J. Comput., 7(1978), 492-509.

[6] ZAKS, S. Lexicographic generation of ordered trees. Th. Comput. Sci., 10(1980), 63-82.

[7] ZAKS, S., RICHARDS, D. Generating trees and other Combinatorial objects lexicographically. SIAM J. Comput., 8(1979), 73-81.