A Ranking Algorithm of Non-Regular Trees in Gray-Code Order

Ro–Yu Wu, Jou–Ming Chang, and An–Hang Chen

***Abstract*—A non-regular tree T with a prescribed branching sequence is an ordered tree whose internal nodes are numbered from 1 to n in preorder such that every node in T has a prescribed number of children. Recently, Wu et al. (2010) introduced a concise representation called RD-sequences to represent all non-regular trees and proposed a loopless algorithm for generating all non-regular trees in a Gray-code order. In this paper, based on such a Gray-code order, we show that a ranking algorithm can be done in quadratic time provided a preprocessing in advance.**

***Index Terms*—Loopless algorithms, ranking algorithms, lexico-graphic order, gray-code order; non-regular trees.**

I. Introduction

A family of combinatorial objects is usually encoded by a set of integer sequences. For convenience, we say objects to mean their integer sequences. To efficiently generate such a set of objects, a generated list is urgently demanded in a Gray-code order, and so that the generating algorithm can be run in a constant time for each generation (i.e., a loopless algorithm), where the measure of computation is the total amount of data structure change and not the time required to output the objects when we analyze such an algorithm. Accordingly, a loopless algorithm is implemented non-recursively by using, after the initialization of the first object, only assignment statements and "if-then-else" statements. The reader is referred to [2], [5]–[9], [15], [17] for loopless generation of combinatorial objects. In particular, see [12] for an excellent survey of generating combinatorial objects in Gray-code orders.

Trees are among the most fundamental of combinatorial objects. A rooted tree is a tree with a distinguished root node. A rooted tree is said to be ordered if the relative order of the subtrees is fixed. An ordered rooted tree is regular (respectively, *non-regular*) if the number of subtrees for internal nodes is unvaried (respectively, varied). Many algorithms have been developed for generating regular trees including binary trees and *k*-ary trees. By contrast, the study of generating non-regular trees has received less attention except for [4], [15], [20]. Recently, Wu et al. [15] dealt with the problem of generating a family of non-regular trees whose internal nodes have a pre-specified degree sequence (called branching sequence)in preorder. Also, a new type of integer sequences called RD-sequences was introduced to represent such a set of non-regular trees. With the help of a coding tree structure, they presented a loopless algorithm to generate RD-sequences of non-regular trees with *n* internal nodes in a Gray-code order using $4n + \mathcal{O}(1)$ space.

Given a specific order on a family of combinatorial objects, a *ranking algorithm* is a function that determines the rank of a given object in the generated list, and an *unranking algorithm* is one that finds the object of a given rank. Many ranking algorithms [1]–[3], [10], [11], [13], [16], [19] and unranking algorithms [1]–[3], [11], [16] for diverse representations of regular trees have been proposed. Note that all proposed algorithms mentioned above dealt with the rank of objects in lexicographic order except for [2] in a Gray-code order. In addition, Wu et al. [14] dealt with the ranking and unranking problems of non-regular trees encoded by RD-sequences in lexicographic order and showed that, given a prescribed branching sequence $(s_1, s_2, \ldots, s_n)$, both ranking and unranking problems can be solved in $\mathcal{O}(nS_{n-1})$ time, where $S_{n-1} = \sum_{i=1}^{n-1}(s_i - 1)$.

In this paper, based on the Gray-code order presented in [15], we propose an efficient ranking algorithm for dealing with non-regular trees encoded by RD-sequences. We show that the ranking algorithm can be done in $\mathcal{O}(n^2)$ time, after a preprocessing stage that takes $\mathcal{O}(n^2 S_{n-1})$ time and space.

II. Non-Regular Trees and Their Loopless Generation

Let $S = (s_1, s_2, \ldots, s_n)$ be an integer sequence where $s_i \geqslant 2$ for $i = 1, 2, \ldots, n$, and let T be a non-regular tree with n internal nodes numbered from 1 to n in preorder (i.e., visiting the root and then recursively the subtrees of T from left to right). S is called the branching sequence for T if node i in T has s_i children for each $1 \leqslant i \leqslant n$. For briefness, we say non-regular trees to mean the set of trees having a given branching sequence and let $\mathcal{T}_S$ denote the set consisting of all non-regular trees with S as their branching sequence. In [15], Wu et al. defined a concise representation, called right- distance sequences (abbreviated as RD-sequences), to describe non-regular trees. For each tree, the *right-distance* of
an internal node $i \in T$, denoted by d_i, is given as follows:

$$d_i = \begin{cases} 0 & \text{if } i = 1 \text{ (i.e., the root);} \\ d_{p(i)} + s_{p(i)} - k & \text{otherwise,} \end{cases}$$

where *p*(*i*) is the parent of node i and k is the rank of node i

Manuscript received September 12, 2012; revised November 7, 2012. This work was supported in part by the National Science Council under contracts NSC100-2221-E-262-019 and NSC101-2115-M-141-001.

Ro-Yu Wu is with the Department of Industrial Management, Lunghwa University of Science and Technology, Taoyuan, Taiwan. (e-mail:eric@mail.lhu.edu.tw)

Jou-Ming Chang and An-Hang Chen are with the Institute of Information and Decision Sciences, National Taipei College of Business, Taipei, Taiwan. (e-mail: spade@mail.ntcb.edu.tw; hang@mail.ntcb.edu.tw)

among all sons of $p(i)$. Then, T can be represented by the sequence $rd(T) = (d_1, d_2, \ldots, d_n)$. Moreover, Wu et al. [15] gave the following characterization of RD-sequences: for a given branching sequence $S = (s_1, s_2, \ldots, s_n)$, an integer sequence $(d_1, d_2, \ldots, d_n)$ is the RD-sequence of a tree $T \in \mathcal{T}_S$ if and only if $d_1 = 0$ and $0 \leqslant d_i \leqslant d_{i-1} + s_{i-1} - 1$. For $i = 2, 3, \ldots, n$ Thus, the number of all possible non-regular trees with respect to a branching sequence S can be determined by the following lemma.

Lemma 1: [15] Let $d_0 = 0$ and $s_0 = 1$. Given a branching sequence $S = (s_1, s_2, \ldots, s_n)$, the cardinality of $\mathcal{T}_S$ is as follows:

$$|\mathcal{T}_S| = \sum_{d_1=0}^{d_0+s_0-1} \sum_{d_2=0}^{d_1+s_1-1} \cdots \sum_{d_n=0}^{d_{n-1}+s_{n-1}-1} 1 \quad \text{for } n \geqslant 1.$$

In particular, every sequence $(d_1, d_2, \ldots, d_n)$ in the above formula is an RD-sequence representing a tree in $\mathcal{T}_S$.

For example, $|\mathcal{T}_{(3,4,2)}| = \sum_{d_1=0}^{0+1-1} \sum_{d_2=0}^{d_1+3-1} \sum_{d_3=0}^{d_2+4-1} 1 = \sum_{d_2=0}^{2}(d_2+4) = 4+5+6 = 15$ and all non-regular trees with branching sequence $(3, 4, 2)$ are shown in Fig. 1.

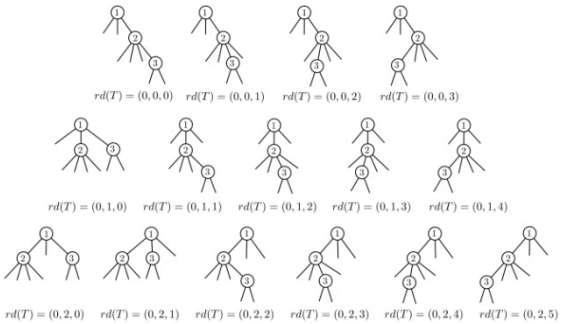

Fig. 1. Non-regular trees with branching sequence $(3, 4, 2)$.

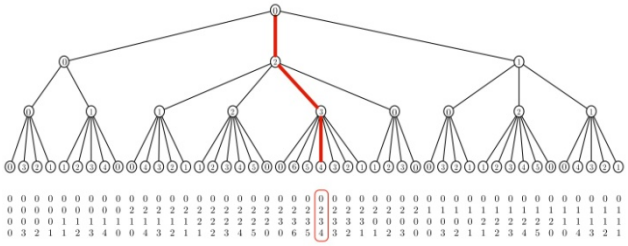

Fig. 2. A flip-flap tree $\mathbb{T}_{(3,2,4,3)}$.

In what follows, we give the concept of the loopless Gray-code generation for non-regular trees proposed in [15]. To describe all non-regular trees in a systematic way, a coding tree called *flip-flap tree* was introduced. Given a branching sequence $S = (s_1, s_2, \ldots, s_n)$, a flip-flap tree $\mathbb{T}_S$ is a rooted labeled tree with n levels constructed by the follows rules:

The first level contains the only one node (i.e., the root) with label 0;

Every node with label d in the ith level $i < n$, has $s_i + d$ sons labeled by $0, 1, \ldots, s_i + d - 1$ in the next level, where the sons are arranged in either $1, 2, \ldots, s_i + d - 1$ (i.e., an up fragment) or $0, s_i + d - 1, \ldots, 2, 1$ (i.e., a down fragment).

If the sons of a node x with label d are arranged in an up fragment, then the sons of its adjacency siblings (i.e., nodes with labels d-1 and d+1 in the same level as x), if exist, are arranged in a down fragment, and vice versa.

Accordingly, nodes on the boundary in every fragment are labeled by either 0 or 1. And, the full labels along a path from the root to a leaf in $\mathbb{T}_S$ indicate the RD-sequence of a non-regular tree. Moreover, any two adjacent RD-sequences in the flip-flap tree differ in exactly one digit. Therefore, this provides the base for designing a loopless Gray-code generation algorithm. For example, Fig. 2 shows a flip-flap tree with respect to the branching sequence S = (3,2,4,3), where the first fragment in each level is a down fragment. Thus, the first RD-sequence appears in $\mathbb{T}_{(3,2,4,3)}$ is (0,0,0,0). Also, a path with bold lines from the root to a leaf in $\mathbb{T}_{(3,2,4,3)}$ represents the non-regular tees with RD-sequence (0, 2, 4, 3).

A procedure called `NextRD()` is designed for generating the next RD-sequence in [15]. Let array s[1..*n*] be the branching sequence S and array d[1..n] be used to store the current RD-sequence in Gray-code order. Array c[1..*n*] is for the type of current fragment at level *i*, $1 \leqslant i \leqslant n$, where c[*i*] = 1 indicates an up fragment and c[*i*] = -1 indicates a down fragment. Array F[1..*n*] is for keeping the track of positions to be processed. Initially, let *i* = n and for each $j = 1, 2, \ldots, n$, set d[*j*] = 0, c[*j*] = -1 and F[*j*] = *j*. When each time `NextRD()` is invoked, d[*i*] will be changed by increasing one or decreasing one which depends on the type of c[*i*] (where arithmetic is taken modulo $d[i-1]+s[i-1]$). If the current fragment is faced on a boundary, it changes the type of c[*i*], appoints the next changed position as *i* = F[n], and performs the updates of F[*i*], F[*i*-1] and F[*n*]; otherwise, it merely keeps the next changed position as *i* = n. The algorithm terminates when the condition *i*= 1 holds. The reader please refer to the details of the algorithm written by C++ in.[15].

III. Ranking Algorithm

Let $S = (s_1, s_2, \ldots, s_n)$ be the prescribed branching sequence. For $1 \leqslant i \leqslant j \leqslant n$, let $S_{i,j} = \sum_{k=i}^{j}(s_k - 1)$, and simply write S_j instead of $S_{1,j}$. By convenience, we define $S_0 = 0$. We first observe that, for $i = 1, 2, \ldots, n$, the largest label for nodes in the ith level of $\mathbb{T}_S$ is exactly S_{i-1}. For instance, the largest labels are $S_0 = 0$, $S_1 = 3 - 1 = 2$, $S_2 = S_1 + (2 - 1) = 3$ and $S_3 = S_2 + (4 - 1) = 6$, respectively, in each level of the flip-flap tree shown in Fig. 2. Next, we observe that if two nodes have the same label in the same level of $\mathbb{T}_S$, then they have the same degree (i.e., the number of children). This further implies that the numbers of leaves in the subtrees rooted at each of the two nodes are the same.

For $1 \leqslant \ell \leqslant n$, let $\mathbb{T}_S^{\ell}$ be the partial tree of $\mathbb{T}_S$ which is formed by the top ℓ levels. Thus, $\mathbb{T}_S^1$ consists of a rooted node and $\mathbb{T}_S^n = \mathbb{T}_S$. For $1 \leqslant i \leqslant \ell \leqslant n$, let $A_{i,k}^{\ell}$ denote the number of leaves in the subtree rooted at a node with label k in the ith level of $\mathbb{T}_S^{\ell}$. Obviously, $A_{\ell,k}^{\ell} = 1$

for $1 \leqslant \ell \leqslant n$ and $0 \leqslant k \leqslant S_{\ell-1}$ and $A^{\ell}_{\ell-1,k} = s_{\ell-1} + k$ for $2 \leqslant \ell \leqslant n$ and $0 \leqslant k \leqslant S_{\ell-2}$. In general, we have

$$A^{\ell}_{i,k} = \sum_{j=0}^{s_i+k-1} A^{\ell}_{i+1,j} \tag{1}$$

where $1 \leqslant i < \ell \leqslant n$ and $0 \leqslant k \leqslant S_{i-1}$. For instance, if we consider the flip-flap tree $\mathbb{T}_{(3,2,4,3)}$, we can build the following tables for calculating $A^{\ell}_{i,k}$.

$A^4_{i,k}$		k						
		0	1	2	3	4	5	6
i	1	46						
	2	9	15	22				
	3	4	5	6	7			
	4	1	1	1	1	1	1	1

$A^3_{i,k}$		k			
		0	1	2	3
i	1	9			
	2	2	3	4	
	3	1	1	1	1

$A^2_{i,k}$		k		
		0	1	2
i	1	3		
	2	1	1	1

For the efficiency of computation, we define the following formula:

$$B^{\ell}_{i,k} = \sum_{j=0}^{k} A^{\ell}_{i,j} \tag{2}$$

where $1 \leqslant i \leqslant \ell \leqslant n$ and $0 \leqslant k \leqslant S_{i-1}$. That is, the term $B^{\ell}_{i,k}$ indicates the total number of leaves for those subtrees rooted at nodes with labels from 0 to k in the ith level of $\mathbb{T}^{\ell}_{S}$. Hence, a table built from Eq. (2) is named as the accumulation table with respect to $\mathbb{T}_S$. For instance, accumulation tables with respect to the partial trees of $\mathbb{T}_{(3,2,4,3)}$ are shown below.

$B^4_{i,k}$		k						
		0	1	2	3	4	5	6
i	1	46						
	2	9	24	46				
	3	4	9	15	22			
	4	1	2	3	4	5	6	7

$B^3_{i,k}$		k			
		0	1	2	3
i	1	9			
	2	2	5	9	
	3	1	2	3	4

$B^2_{i,k}$		k		
		0	1	2
i	1	3		
	2	1	2	3

Obviously, the time and space required for building accumulation tables are

$$\sum_{i=1}^{n}(S_{i-1}+1)i = \sum_{i=0}^{n-1}(S_i+1)(i+1) = \mathcal{O}(n^2 S_{n-1}).$$

We are now in a position to determine the rank of a non-regular tree in the Gray-code order of [15]. For a given non-regular tree $T \in \mathcal{T}_S$ associated with the RD-sequence $rd(T) = (d_1, d_2, \ldots, d_n)$, let x_i denote the node with label d_i in $\mathbb{T}_S$ such that the full labels along the path $x_1, x_2, \ldots, x_n$ represent the given RD-sequence. For each $i = 1, 2, \ldots, n$, let *R*(*i*) be the rank of x_i in the ith level of $\mathbb{T}_S$. Hereafter, the rank of a node in each level of $\mathbb{T}_S$ is ordered from left to right and a ranking always starts with 0. Note that *R*(1) = 0 and our ranking algorithm is to obtain R(n). Since each level of $\mathbb{T}_S$ begins with a down fragment and the two types of fragments appear alternately, it is easy to check the following property.

Proposition 1: For $1 \leqslant i < n$, if the rank of a node in the ith level of $\mathbb{T}_S$ is even (respectively, odd), then its sons are arranged in a down fragment (respectively, an up fragment).

Initially, we set *R*(*i*) = 0 for $i = 1, \ldots, n$. Then, the computation of *R*(*i*) requires i - 1 updates by means of Eqs. (3) and (4). For $i = 1, \ldots, n-1$, let j = *i* + 1. If x_j is not the first node in a fragment, let y_j be the node with rank *R*(*i*) - 1 in the *jth* level of $\mathbb{T}_S$. Then, for each $h = j, j+1, \ldots, n$, we update R(h) to represent the rank of the rightmost descendant of y_j in the hth level of $\mathbb{T}_S$. By Proposition 1, there are two cases to compute *R*(*h*). If $R(i) \equiv 0 \pmod 2$, the sons of x_i are arranged in a do wn fragment. In this case, we have

$$R(h) = R(h) + (B^h_{j,d_i+(s_i-1)} - B^h_{j,d_j} + B^h_{j,0}). \tag{3}$$

On the other hand, if $R(i) \equiv 1 \pmod 2$, the sons of x_i are arranged in an up fragment. In this case, we have

$$R(h) = R(h) + (B^h_{j,d_i+(s_i-1)} - B^h_{j,d_j} + B^h_{j,0}). \tag{4}$$

The meanings of Eqs. (3) and (4) are illustrated in Fig. 3and Fig. 4, respectively. The details of the ranking algorithm are shown below.

Function Ranking$(d_1, d_2, \ldots, d_n)$
begin
  for $i = 1$ **to** n **do** $R(i) = 0$;
  for $i = 1$ **to** $n-1$ **do**
    $j = i + 1$;
    if $R(i) \equiv 0 \pmod 2$ **then** // a down fragment
      if $d_j \neq 0$ **then**
        for $h = j$ **to** n **do**
          $R(h) = R(h) + (B^h_{j,d_i+(s_i-1)} - B^h_{j,d_j} + B^h_{j,0})$;
    else // an up fragment
      if $d_j \neq 1$ **then**
        for $h = j$ **to** n **do**
          $R(h) = R(h) + (B^h_{j,d_j-1} - B^h_{j,0})$;
  return $R(n)$;

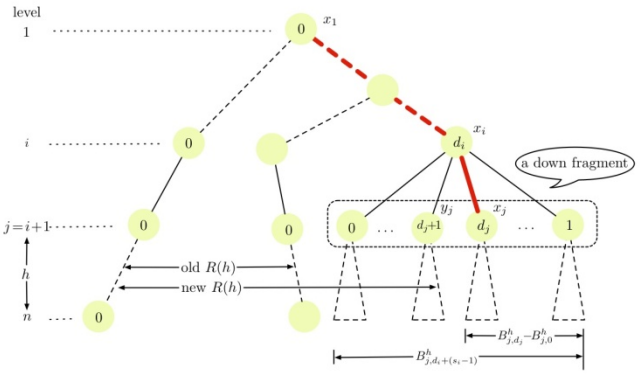

Fig. 3. Illustration of Eq. (3).

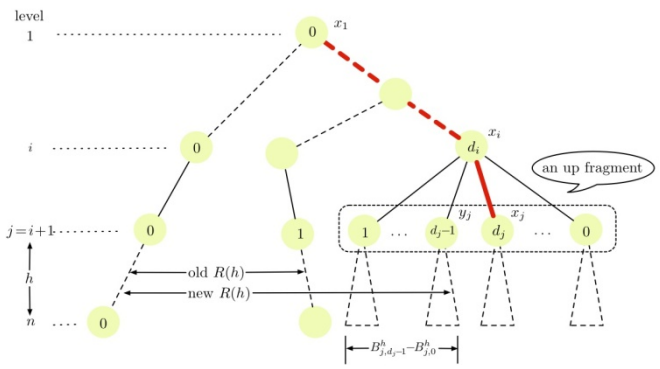

Fig. 4. Illustration of Eq. (4).

Example 1. We consider a non-regular tree $T \in \mathscr{T}_{(3,2,4,3)}$ with rd(T)=(0,2,3,4) and perform Ranking(0,2,3,4).

Initially, $R(1) = R(2) = R(3) = R(4) = 0$.

When $i = 1$, since $R(1) = 0$ is even, the sons of x_1 are arranged in a down fragment. Thus, we have

$$R(2) = R(2) + B^2_{2,2} - B^2_{2,2} + B^2_{2,0} = 0 + 3 - 3 + 1 = 1,$$
$$R(3) = R(3) + B^3_{2,2} - B^3_{2,2} + B^3_{2,0} = 0 + 9 - 9 + 2 = 2,$$
$$R(4) = R(4) + B^4_{2,2} - B^4_{2,2} + B^4_{2,0} = 0 + 46 - 46 + 9 = 9.$$

When $i = 2$, since $R(2) = 1$ is odd, the sons of x_2 are arranged in an up fragment. Then, we have the following updates:

$$R(3) = R(3) + B^3_{3,2} - B^3_{3,0} = 2 + 3 - 1 = 4,$$
$$R(4) = R(4) + B^4_{3,2} - B^4_{3,0} = 9 + 15 - 4 = 20.$$

When $i = 3$, since $R(3) = 6$ is even, the sons of x_3 are arranged in a down fragment. We obtain

$$R(4) = R(4) + B^4_{4,6} - B^4_{4,4} + B^4_{4,0} = 20 + 7 - 5 + 1 = 23.$$

As a result, the algorithm outputs $R(4) = 23$.

Since the time complexity and space requirement for building accumulation tables are both $\mathcal{O}(n^2 S_{n-1})$ and the ranking algorithm can be run in $\mathcal{O}(n^2)$ time, we conclude the following.

Theorem 1: Determining the rank of a non-regular tree T with a prescribed branching sequence $(s_1, s_2, \ldots, s_n)$ in a Gray-code order can be done in $\mathcal{O}(n^2 S_{n-1})$time and space,

where $S_{n-1} = \sum_{i=1}^{n-1}(s_i - 1)$

IV. Conclusion and Future Work

In this paper, we present an efficient ranking algorithm of non-regular trees (encoded by RD-sequences) with a prescribed branching sequence in a Gray-code order. Obviously, if the given branching sequence $S = (s_1, s_2, \ldots, s_n)$ satisfies $s_1 = s_2 = \cdots, s_n = k$, then $\mathscr{T}_S$ is the set of k-ary trees with n internal nodes. Thus, our algorithm can also be used to deal with k-ary trees in $\mathcal{O}(kn^3)$ time and space. As a future work, we will design an efficient unranking algorithm. In particular, an interesting and challenging problem is how to reduce the time complexity and space requirement for dealing with the ranking and unranking problems of non-regular trees encoded by RD-sequences in a Gray-code order.

References

[1] H. Ahrabian and A. Nowzari-Dalini, "Generation of t-ary trees with ballot-sequences," *Int. J. Comput*. Math., 80 (2003) pp. 1243-1249.

[2] A. Ahmadi-Adl, A. Nowzari-Dalini, and H. Ahrabian, "Ranking and unranking algorithms for loopless generation of t-ary trees," *Logic J. IGPL*, vol.19, pp. 33-43, 2011.

[3] M.C. Er, Lexicographic listing and ranking t-ary trees, Comput. J., 30 (1987) pp.569-572.

[4] M. C. Er, "A simple algorithm for generating non-regular trees in lexicographic order," *Computer J.*, vol. 31, pp. 61-64, 1988.

[5] J. F. Korsh, "Loopless generation of k-ary tree sequences," *Inform. Process. Lett*, vol. 52, pp. 243-247, 1994.

[6] J. F. Korsh and P. LaFollette, "Loopless generation of Gray codes for k-ary trees," *Inform. Process. Lett.*, vol. 70, pp. 7-11, 1999.

[7] J. F. Korsh and P. LaFollette, "A loopless Gray code for rooted trees," *ACM Trans. Algorithms*, vol. 2, pp. 135-152, 2006.

[8] J. F. Korsh and S. Lipschutz,Shift, "loopless generation of k-ary trees," *Inform. Process. Lett*., vol. 65 ,pp. 235-240, 1998.

[9] D. Roelants van Baronaigien, "A loopless Gray-code algorithm for listing k-ary trees," *J. Algorithms*.vol. 35, pp.100-107, 2000.

[10] D. Roelants van Baronaigien and F. Ruskey, "Generating t-ary trees in A-order," *Inform. Process. Lett.*, vol. 27, pp. 205-213, 1988.

[11] F. Ruskey, "Generating *t*-ary trees lexicographically," *SIAM J. Computing*, vol. 7, pp. 424-439, 1988.

[12] C. D. Savage, "A survey of combinatorial Gray codes," *SIAM Review*, vol. 39, pp. 605-629, 1997.

[13] A.E. Trojanowaki, "Ranking and listing algorithms for k-ary trees," SIAM J.Computing, vol. 7, pp. 492-509, 1978.

[14] R.-Y. Wu, J.-M. Chang, and C.-H. Chang, "Ranking and unranking of non-regular trees with a prescribed branching sequence," *Math. Comput. Modelling,* vol. 53, pp. 1331-1335, 2011.

[15] R. Y. Wu, J. M. Chang, and Y.L. Wang, "Loopless Generation of non-regular trees with a prescribed branching sequence," *Comput. J*., vol. 53, pp. 661-666, 2010.

[16] R. Y. Wu, J. M. Chang, and Y. L. Wang, "Ranking and unranking of t-ary trees using RD-sequences," *IEICE Trans. Inform. Systems,* E94-D, pp.226-232, 2011.

[17] L. Xiang, K. Ushijima, and C. Tang, "Efficient loopless generation of Gray codes for k-ary trees," *Inform. Process. Lett*, vol. 7, pp. 169-174, 2000.

[18] S. Zaks, "Lexicographic generation of ordered trees," *Theore. Comput. Sci*.,vol. 10, pp. 63-82, 1980.

[19] S. Zaks, "Generation and ranking of k-ary trees", *Inform. Process. Lett.,* vol. 14, pp. 44-48, 1982.

[20] S. Zaks and D. Richards, "Generating trees and other combinatorial objects lexicographically," *SIAM J. Computing,* vol. 8, pp. 73-81, 1979.

Ro-Yu Wu received his B.S. degree in industrial engineering from Tunghai University in 1985, an M.B.A. degree in industrial management from National Cheng Kung University in 1987, and a Ph.D. degree in information management from National Taiwan University of Science and Technology in 2007. Currently, he is an associate professor in the Department of Industrial Management of Lunghwa University of Science and Technology. Dr. Wu's major research interest includes reliability, graph theory, and combinatorial algorithms.

Jou-Ming Chang received his B.S. degree in applied mathematics from Chinese Culture University in 1987, an M.S. degree in information management from National Chiao Tung University in 1992, and a Ph.D. degree in computer science and information engineering from National Central University in 2001. Currently, he is a professor in the Institute of Information and Decision Sciences of National Taipei College of Business. Dr. Chang's major research interest includes algorithm design and analysis, graph theory, and parallel and distributed computing.

An-Hang Chen received his B.S. degree in applied mathematics from Chinese Culture University in 1987, an M.B.A. degree in management science from National Chiao Tung University in 1993, and a Ph.D. degree in information management from National Taiwan University of Science and Technology in 2008. Currently, he is an associate professor and the director in the Institute of Information and Decision Sciences of National Taipei College of Business. Dr. Chen's major research interest includes graph theory and algorithms.