Bull. Iranian Math. Soc.
Volume 00, Number 0, Pages 1–16
S 0000-0000(XX)0000-0

EFFICIENT GENERATION, RANKING AND UNRANKING OF (k,m)-ARY TREES IN B-ORDER

M. AMANI AND A. NOWZARI-DALINI *

Abstract. In this paper, we present a new generation algorithm with corresponding ranking and unranking algorithms for (k,m)-ary trees in B-order. (k,m)-ary tree is introduced by Du and Liu. A (k,m)-ary tree is a generalization of k-ary tree whose every node of even level of the tree has degree k and odd level of the tree has degree 0 or m. Up to our knowledge no generation, ranking or unranking algorithms are given in the literature for this family of trees. We use Zaks' encoding for representing (k,m)-ary trees, and to generate them in B-order. We also prove that, to generate (k,m)-ary trees in B-order using this encoding, the corresponding codewords should be generated in *reverse-lexicographical* ordering. The presented generation algorithm has a constant average time and $O(n)$ time complexity in the worst case. Due to the given encoding, both ranking and unranking algorithms are also presented taking $O(n)$ and $O(n \log n)$ time complexity, respectively.
Keywords: (k,m)-ary trees, Tree Generation, Ranking, Unranking.
MSC(2010): Primary: 05c05; Secondary: 68w32, 05c85.

1. Introduction

Exhaustive generation of certain combinatorial objects has always been of great interest for computer scientists [16, 24, 26, 31]. In general, generation of combinatorial structure problem is to construct all possible combinatorial structures of a particular kind in a certain order [14]. The problem of generating trees, as one of the most important combinatorial objects, has been thoroughly investigated in the literature and many papers have been published which deal with the generation of these objects. For example, we can mention the generation of binary trees in [2, 3, 28, 29], k-ary trees in [4, 11, 13, 20, 21, 22, 31],trees with n nodes and m leaves in [17, 23], neuronal trees in [5, 6, 18, 27], non-regular trees in [30], bounded ordered trees in [7], Fibonacci-isomorphic trees in [8], and AVL trees in [15].

In most of tree generation algorithms, a tree is represented by an integer or alphabet sequence called *codeword*, hence all possible sequences of this representation are generated. This operation is called *tree encoding*, and the reverse of it, construction of tree from its code, is called *tree decoding*.

Date: Received: , Accepted: .
*Corresponding author.

Basically, the uniqueness of encoding, the length of the encoding, and the capability of decoding, are essential considerations in the design of the tree encoding schema.

Any generation algorithm imposes an ordering on the set of trees. In general, on any class of trees, we can define a variety of ordering for the set of trees. When tree codewords are generated with a certain order, consequently their corresponding trees are also generated in a specific order. Classical orderings on trees are *A-order* and *B-order* [28, 27, 31], and on sequences (codewords) are lexicographical [31], cool-lex [10], and Gray code [12, 13, 22, 19, 31].

Besides the generation algorithm for trees, ranking and unranking algorithms are also important in the concept of tree generation [1, 22, 30, 31]. For a given class of trees, the position of a given tree T among all trees in that class with respect to a given ordering is called *rank*; the *ranking* function determines the rank of T. An *unranking* function returns the tree T having a specified rank. Thus, ranking and unranking can be considered as inverse operations.

One of the best algorithms for generating k-ary trees in B-order is presented by Zaks [31]. He introduced z-sequences for representing k-ary trees and presented a generation algorithm for generating these sequences. This algorithm generates each z-sequence in constant average time $O(1)$, and $O(n)$ time complexity in the worst case. He also presented two ranking and unranking algorithms for z-sequences. Both of these algorithms have time complexity $O(kn)$ using a precomputed table. In this paper, we consider (k, m)*-ary trees*. A (k, m)-ary tree is a generalization of k-ary tree. In (k, m)-ary tree every node of even levels of the tree has degree k and odd levels of the tree has degree 0 or m. A (k, m)-ary tree of order n has exactly n nodes of degree m on odd levels. This class of trees is first introduced by Du and Liu [9]. They proved that the total number of (k, m)-ary trees of order n is equal to $C_{k,m}(n)$ which is called (k, m)*-Catalan number of order* n, then they studied basic combinatorial properties of (k, m)-ary trees or (k, m)-Catalan number. But, up to our knowledge, no generation, ranking or unranking algorithms have been proposed for this class of trees. We use Zaks' encoding [31] to encode (k, m)-ary trees, so we present a new generation algorithm to generate them in B-order. The generating algorithm has constant average time and $O(n)$ time complexity in the worst case. Corresponding ranking and unranking algorithms are also designed with $O(n)$ and $O(n \log n)$ time complexities, respectively. These algorithms use a precomputed table where takes $O(kn)$ time.

The remaining of this paper is organized as follows. Section 2 introduces the definitions and notions that is used further. The Zaks' encoding [31] and our new generation generation algorithm for (k, m)-ary trees in B-order are presented in Section 3. Based on the generation algorithm, the corresponding ranking and unranking algorithms are given in Section 4. Finally, in Section 5, concluding remarks and future works are discussed.

2. Definitions

A tree is *rooted* if one of its internal nodes is distinguished from the others as the *root*. Let T be a tree with root r. For each node v of T, we say that v is on *level* j if the unique path from r to v has length j, and the root is said to be on the level 0. A rooted tree where the children of each node have a designated order is called *ordered tree*. An *external node* (or a *leaf node*) is a node that has no children, and the other nodes are known as *internal nodes*. A k*-regular* tree is a rooted tree in which each internal node has exactly k children. To construct a k-regular tree from a rooted tree with maximum degree k, all nodes are considered as internal nodes and to every node which has $q < k$ children, $k - q$ special nodes are added as its children and these nodes are considered as external nodes. A k*-ary* tree is an ordered k-regular tree (every internal node has exactly k ordered children).

A (k, m)*-ary tree* is a generalization of k*-ary tree* which the degree of every internal node is determined based on the *level* of the node. This class of trees is first introduced by Du and Liu [9].

More precisely, (k, m)*-ary tree of order* n can be defined as follows [9].

Definition 2.1. For $k, m \geq 1$ and $n \geq 0$, a (k, m)-ary tree of order n is an ordered tree such that:

- All nodes in even levels have degree k (root is on the level 0).
- All nodes in odd levels have degree m or 0, and there are exactly n nodes of degree m in odd levels.

An example of a $(2, 3)$-ary tree of order 4 is shown in Figure 1. In this figure, the shaded nodes are the internal nodes in even levels so they all have degree 2 and the gray nodes are the internal nodes in odd levels with degree 3.

The set of k-ary trees with n internal nodes is shown by $T_k(n)$ and the set of (k, m)-ary trees of order n is shown by $T_{k,m}(n)$. It is well known that the number of t-ary trees with n internal nodes is counted by generalized Catalan number [25], i.e. for the set of k-ary trees we have:

$$|T_k(n)| = C_k(n) = \frac{1}{kn+1}\binom{kn+1}{n}.$$

Du and Liu [9] defined (k, m)*-Catalan number of order* n shown by $C_{k,m}(n)$, and proved that:

$$C_{k,m}(n) = \frac{1}{mn+1}\binom{(mn+1)k}{n}.$$

They also had shown that the total number of (k, m)-ary trees of order n is equal to $C_{k,m}(n)$, i.e. $|T_{k,m}(n)| = C_{k,m}(n)$.

As mentioned, any generation algorithm imposes an ordering on the set of generated trees; such an orderings is B-order [28, 27, 31] which is defined as follows.

FIGURE 1. A $(2, 3)$-ary tree of order 4 with 39 nodes.

Definition 2.2. Let T and T' be two trees and $k = max\{deg(T), deg(T')\}$, we say that T is less than T' in B-order ($T \prec_B T'$), iff

- $deg(T) < deg(T')$, or
- $deg(T) = deg(T')$ and for some $1 \leq i \leq k$, $T_j =_B T'_j$ for all $j = 1, 2, \ldots, i-1$, and $T_i \prec_B T'_i$.

Where $deg(T)$ is defined as the degree of root of the tree T, and T_i and T'_i show the i^{th} subtree of T and T', respectively.

B-order is also referred to as local order, because in this ordering, we compare the characters of the concurrent nodes (whether they are internal nodes or leaves).

In most generation algorithms, trees are encoded as integer sequences and these sequences are generated in specific order such that their corresponding trees are in predefined order (e.g. B-order). One of the most well-known ordering on sequences is *lexicographical ordering* [12, 22, 31], which is defined as follows.

Definition 2.3. The two integer sequence $v = (v_1, v_2, \cdots, v_n)$ and $v' = (v'_1, v'_2, \cdots, v'_m)$ are in lexicographical order (denoted by $v \prec_{lex} v'$ or $v < v'$), if there exists $i \in [1, min(n, m)]$ such that

(1) $v_j = v'_j$ for all $j \in [1, i-1]$,
(2) $v_i < v'_i$.

In this paper, our generation algorithm, which is given in next section, generates the tree corresponding codewords in *reverse-lexicographical* ordering, meaning the reverse manner of lexicographical ordering, such that their corresponding trees are in B-order.

3. Encoding and Generation algorithm

The main point in generating trees is to choose a suitable encoding to represent them, and generate their corresponding codewords instead. In this section, we use the Zaks' encoding [31] on trees and propose a new generation algorithm using this encoding to generate (k,m)-ary tree in B-order.

In 1980, Zaks investigated the trees generation problem in some details, and presented a nice method for encoding k-ary trees [31]. In this encoding, each internal node of tree T is labeled with 1 and each external node with 0. By traversing the tree T in preorder and omitting the last zero, a 0-1 sequence (or bit sequence) of n 1's and $(k-1)n$ 0's is obtained. This 0-1 sequence is called *x-sequence* and is denoted by $x(T) = (x_i)_1^{kn}$. But since there are many zeros in the bit sequence representation of t-ary trees, it is common that the position of the ones are often listed. Therefore, a sequence $z(T) = (z_i)_1^n$ from $x(T)$ can be built, such that z_i is the position of i^{th} 1 in x. This sequence is called *z-sequence*. He also proved that for given trees T and T', and the corresponding sequences $x(T), x(T'), z(T)$, and $z(T')$; $T \prec_B T'$ iff $x(T) \prec_{lex} x(T')$ and $z(T') \prec_{lex} z(T)$.

To encode (k,m)-ary trees, we use Zaks' definition of x-sequence and z-sequence with a slight change on which nodes would contribute in the labeling. For this matter, the following lemma presents some basic properties of $T_{k,m}(n)$.

Lemma 3.1. *Let $T \in T_{k,m}(n)$, then:*

(1) The total number of nodes in odd levels of T is equal to $kmn + k$.
(2) The total number of nodes in even levels of T is equal to $nm + 1$.
(3) The number of leaves of T is equal to $n(km-1) + k$.
(4) The total number of nodes (size) of T is equal to $(k+1)(nm+1)$.

Proof. The proof comes immediately by counting the number of nodes or leaves below each internal node of odd levels and root as follows.

(1) The root of T which is in level 0, has k children which are in odd level (level 1), and for each of n internal nodes in odd levels of T, there are exactly m children in even levels which any of them has k children in odd levels. Therefore the total number of nodes in odd levels is $nmk + k$.

(2) Root lies in an even level and for each of n internal nodes in odd levels of T, there are exactly m children in even levels, therefore the total number of nodes in even levels is $nm + 1$.
(3) By definition, all leaves are in odd levels and the total number of nodes in odd levels is $kmn + k$ which n of them are internal nodes. Therefore the total number of leaves is $kmn+k-n = n(km-1)+k$.
(4) By summing up the number of nodes in odd levels and even levels we obtain the total number of nodes which is $kmn + k + nm + 1 = (k + 1)(nm + 1)$.

□

Regarding properties of (k, m)-ary trees, our new encoding is presented as follows. For any (k, m)-ary tree T of order n, the internal nodes in even levels (including root) are ignored, the internal nodes of odd levels are labeled by 1 and the external nodes by 0, then by a preorder traversal of T, we obtain the x-sequence codeword $x(T)$. The z-sequence of T is $z(T) = (z_i)_1^n$ such that z_i is position of i^{th} 1 in $x(T)$. Since the internal nodes of odd levels of T which have been labeled by 1, represent the order of the tree, we call them *order-internal nodes.*

For example, the x-sequence and z-sequence codewords of the $(2, 3)$-ary of Figure 1 are $x = (1000000100110000000000000000)$ and $z = (1, 8, 11, 12)$.

The x-sequence and z-sequence codewords corresponding to the first (k, m)-ary tree of order n in B-order are

$$\text{“}0^{k-1}10^{km-1}1 \dots 10^{km} = 0^{k-1}1(0^{km-1}1)^{n-1}0^{km}\text{”}$$

and “$k, k + mk, k + 2mk, \dots , k + nmk$”, respectively. These sequences and corresponding trees are shown in Figure 2. Also, the x-sequence and z-sequence codewords corresponding to the last (k, m)-ary tree of order n in B-order are “$1^n0^{n(km-1)+k}$” and “$1, 2, 3, 4, \dots , n$”, respectively.

An integer sequence is said to be feasible regarding a (k, m)-ary tree encoding scheme, if it represents a valid (k, m)-ary tree. In the following definition and theorems we study the necessary and sufficient condition for x-sequences and z-sequences to be feasible.

A x-sequence x is said to have the (k, m)*-dominating property* if in any prefix of x with p 0's, the number of 1's is at least:

$$\lceil \frac{p-k}{km-1} \rceil,$$

where $\lceil i \rceil$ means the smallest integer greater than or equal to i. In other words, if any prefix of the sequence contains q 1's, then it contains at most $(km-1)q+k$ 0's. A z-sequence is said to have the (k, m)*-dominating property* if its corresponding x-sequence has (k, m)-dominating property.

Theorem 3.2. *For the class of trees $T_{k,m}(n)$, a x-sequence codeword $x = (x_i)_1^j$ is feasible iff:*

FIGURE 2. The first (k, m)-ary tree of order n in B-order with corresponding x-sequences and z-sequences.

(1) $j = kmn + k$ *(j is the number of elements in x), and the number of 1's is n and the number of 0's is* $(km - 1)n + k$.
(2) It has the (k, m)*-dominating property.*

Proof. If x is feasible, then:

(1) It corresponds to a (k, m)-ary tree, and by Lemma 3.1, it should has $kmn + k$ nodes in odd levels consist of n internal nodes and $(km - 1)n + k$ leaves, hence $j = kmn + k$ and the number of 1's is n and the number of 0's is $(km - 1)n + k$.
(2) By adding some probable missing 0's to the end of any prefix of x, we obtain an x-sequence corresponds to a (k, m)-ary tree, and by Lemma 3.1, we know that in any (k, m)-ary tree with p external nodes (= number of 0's), the number of internal nodes of odd levels (= number of 1's) is $\frac{\mathrm{p}-k}{km-1}$, hence (k, m)-dominating property is satisfied.

These two conditions are also sufficient, this can be proved by an induction on the length of x. Let x be a sequence obtained in the above manner. Initially, for a sequence of length k (a (k, m)-ary tree of order 0 which is just a root with k leaves as its children), the proof is trivial. Assume that each sequence smaller than x obtained in the above manner encodes a unique (k,m)-ary tree, for the sequence x, the first element represents the first child of the root (which its subtree is also a (k,m)-ary tree) followed by the corresponding subsequence of its subtree with above conditions, then the next element represents the second child again followed by its corresponding subsequence of its subtree and so on. By putting all these subsequences together, it can be easily seen that the (k,m)-dominating condition still holds.

□

Corollary 3.3. *For a tree $T \in T_{k,m}(n)$ and a z-sequence $z = (z_i)_1^n$ (note that the length of z is n), the following terms are equivalent:*

- *z is feasible.*
- *z has the (k,m)-dominating property.*
- *$0 < z_1 < z_2 < \ldots < z_n$ and $z_i \leq (i-1)km + k$.*

Proof. By the definition of x-sequence, z-sequence, Lemma 3.1 and Theorem 3.2. □

The following theorem and corollary show the reverse relation between (k,m)-ary trees in B-order and their corresponding codewords in lexicographic ordering.

Theorem 3.4. *Let T and T' be two (k,m)-ary trees of order n, and let x and z be the x-sequences and z-sequences corresponding to T, and x' and z' be the x-sequences and z-sequences corresponding to T'. The following terms are equivalent:*

(1) $T \prec_B T'$.
(2) $x \prec_{lex} x'$.
(3) $z \succ_{lex} z'$ (means $z' \prec_{lex} z$).

Proof. $T \prec_B T'$ iff there is an i such that all first $i-1$ order-internal nodes (internal nodes in odd levels) in preorder traversal of T and T' are the same but the i^{th} one in T is a leaf while in T' is an internal node. This is equivalent to say that the first $i-1$ elements of x and x' are the same while the i^{th} element of x is less than the i^{th} element of x'. So $x \prec_{lex} x'$. On the other hand by definition of x-sequence and z-sequence it is clear that if $x \prec_{lex} x'$, then $z \succ_{lex} z'$. Therefore all the three terms are equivalent. □

Corollary 3.5. *Given two (k,m)-ary trees T and T', T is less than T' in B-order ($T \prec_B T'$), iff their x-sequences are in lexicographic ordering and their z-sequences are in reverse-lexicographic ordering.*

Proof. Immediately from the previous theorem. □

Based on the above theorem, the generation of z-sequences in reverse-lexicographic ordering corresponds to the generation of (k,m)-ary trees in

```
Function kmNextBorder(k, m, n: integer):boolean;
begin
    kmNextBorder:=true; $i := n$;
    while ( $(Z[i-1] = (Z[i]-1))$ & $(i > 1)$ ) do $i{:=}i-1$;
    if ( $(i = 1)$ & $(Z[i] = 1)$ ) then
        kmNextBorder := false
    else begin
        $Z[i]{:=}Z[i]-1$;
        for $j := i+1$ to n do
            $Z[j]{:=}\ (j-1) \times k \times m + k$;
    end;
end;
```

FIGURE 3. Algorithm *kmNextBorder* for generating $T_{k,m}(n)$ trees in B-order.

B-order. The algorithm given in Figure 3 effectively generates reverse-lexicographically all codewords of (k, m)-ary trees of order n. Thus (k, m)-ary trees are generated in B-order.

This generation algorithm returns the successor of a given z-sequence $z = \{z_1, z_2, \cdots, z_n\}$ stored in array Z with length n. This global array Z will be used both as input and then as output when it has been updated to the successive z-sequence. In this algorithm, array Z is scanned from right to left, and the first element which can be decreased is obtained and decremented (if there is no such an element, then this codeword corresponds to the last codeword and there is no successor), then all the elements in the right side of the decremented element will be replaced with their maximum possible value regarding Lemma 3.1 and Corollary 3.3.

Theorem 3.6. *The generation algorithm presented in Algorithm 3 has a worst case time complexity of $O(n)$ and an average time complexity of $O(1)$ (n is the order of the given (k, m)-ary tree which is the number of nodes of degree m in odd levels).*

Proof. Worst case time complexity of this algorithm is $O(n)$ because the codeword is scanned just once. For computing the average time, first notice that since k and m are fixed, for big values of n we can say $k = O(1)$ and $m = O(1)$. Now, consider generation of all $C_{k,m}(n) = \frac{1}{mn+1}\binom{(mn+1)k}{n}$ trees of $T_{k,m}(n)$, and let F_i be the number of times that in the generation process the array Z is scanned up to the position i, then $Z[i]$ is decremented and the right side of $Z[i]$ is updated. Therefore the average cost will be equal to:

$$\frac{\sum_{i=1}^{n} F_i O(n-i)}{C_{k,m}(n)},$$

since Z should be scanned from the end to the i^{th} position. On the other hand, for the first tree of $T_{k,m}(n)$ in B-order, $Z[1] = k$ as it has been shown in Figure 2.a. Therefore, $F_1 = k$ (because $Z[1] = k$ for the first tree and then decrements by one each time the algorithm reaches $Z[1]$). Let us show by $v_{Z[i]}$ the node of the tree which has been labeled by $Z[i]$. During the process of the generation of all $T_{k,m}(n)$ trees, every time $Z[1]$ is decremented, km grandchildren of $v_{Z[1]}$ will be recreated in the tree which every one of them has a potential to be $v_{Z[2]}$ in the successive generated trees, until $Z[1]$ is decremented again. This means $F_2 \geq F_1 \times km$. In the same manner for every $2 \leq i \leq n$, $F_i \geq F_{i-1} \times km$.

This is equivalent to say that F_i increases at least exponentially, therefore:

$$\frac{\sum_{i=1}^{n} F_i O(n-i)}{C_{k,m}(n)} = \frac{O(F_n)}{C_{k,m}(n)} < O(\frac{C_{k,m}(n)}{C_{k,m}(n)}) = O(1).$$

Which means the average cost is constant. □

While in this section we have presented the feasible encoding of $T_{k,m}(n)$ with an efficient and simple algorithm to generate them in B-order, in the next section we study the corresponding ranking and unranking algorithms.

4. Ranking and Unranking algorithms

As mentioned before, the rank of a tree $T \in T_{k,m}(n)$ with respect to B-order is the number of trees $T' \in T_{k,m}(n)$ that come before it in B-order, and the unranking algorithm determines the (k,m)-ary tree (i.e. the corresponding z-sequence) having a particular rank. In this section, ranking and unranking algorithms for (k,m)-ary trees in B-order are given.

To propose our ranking and unranking algorithms, we consider k and m as fixed and constant values. For designing the ranking and unranking algorithms, we need the following theorems and definitions.

Let us define *extended-(k,m,d)-ary trees of oder n* as (k,m)-ary trees where the root has degree d instead of k (i.e. besides the degree of the root, the rest of the tree has the same structure). Let $B^{k,m}_{n,d}$ be the number of extended-(k,m,d)-ary trees of order n. For the sake of simplicity, since we fixed k and m, we denote $B^{k,m}_{n,d}$ by $B_{n,d}$. Note that by definition, $B_{n,k} = C_{k,m}(n)$.

Theorem 4.1. *$B_{n,d}$ can be computed as follows.*

$$B_{n,d} = \begin{cases} 1 & \textit{if } n = 0 \\ d & \textit{if } n = 1 \\ B_{n-1,km} & \textit{if } d = 1 \\ B_{n,d-1} + \sum_{i=1}^{n-1}(B_{i,km} \times B_{n-1-i,d-1}) & \textit{if } n,d > 1 \end{cases}$$

Proof. The initial cases for $n = 0$ are $n = 1$ are trivial. If $d = 1$, the root has only one child which is an order-internal node. Therefore, if we remove all the nodes on the two levels bellow the root and connect directly the root to the km nodes on the third level (i.e. remove the root's unique child and its m grandchildren and connecting the km great-grandchildren to the root), we obtain an extended-(k, m, km)-ary tree of order $n-1$ which is counted by $B_{n-1,km}$.

If both n and d are greater than 1, then, let T be an extended-(k, m, d)-ary tree of oder n, and let the first subtree of T be T_1. The root of T_1 is either a leaf or an order-internal node.

- If the root of T_1 is a leaf, then just remove it from the tree, what remains is an extended-$(k, m, d-1)$-ary tree of order n which is counted by $B_{n,d-1}$.
- If the root of T_1 is an order-internal node, let us assume that T_1 has i order-internal nodes (excluding T_1's root). Then, by removing in T_1 all the children of its root and connecting its root to all grandchildren, T_1 becomes an extended-(k, m, km)-ary tree of order i which is counted by $B_{i,km}$. Moreover, by removing the entire T_1, we are left with another extended-$(k, m, d-1)$-ary tree of order $n-1-i$ which is counted by $B_{n-1-i,d-1}$. Since $i \in [1, n-1]$, in this case the number of possible trees is:

$$\sum_{i=1}^{n-1}(B_{i,km} \times B_{n-1-i,d-1}).$$

Putting all together, in the case of both n and d are greater than 1, the total number of possible trees is counted by

$$B_{n,d-1} + \sum_{i=1}^{n-1}(B_{i,km} \times B_{n-1-i,d-1}).$$

Hence, the proof is complete. □

The following theorem also shows another recursive formula to compute $B_{n,d}$.

Theorem 4.2. *$B_{n,d}$ can also be computed as follows.*

$$B_{n,d} = \begin{cases} 1 & \text{if } n = 0 \\ d & \text{if } n = 1 \\ B_{n,d-1} + B_{n-1,km+d-1} & \text{if } n > 1 \end{cases}$$

Proof. As in Theorem 4.1, the initial cases for $n = 0$ and $n = 1$ are trivial. If $n > 1$, then, again let T be an extended-(k, m, d)-ary tree of oder n, and let the first subtree of T be T_1. T_1 is either a leaf or an order-internal node.

- If the root of T_1 is a leaf then just remove it from the tree, and obtain an extended-$(k, m, d-1)$-ary tree of order n which is counted by $B_{n,d-1}$.

```
**Function**$ZseqBorderRank(k, m, n$:**integer**):**integer**;
**begin**
    $NegR := 0$;
    **for** $i := 1$ **to** $n$ **do**
    **begin**
        $NegR := NegR + B_{n+1-i,k+(i-1)km-Z[i]}$;
    **end**;
    $ZseqBorderRank := B_{n,k} - NegR$;
**end**;
```

FIGURE 4. Ranking algorithm for (k, m)-ary trees in B-order.

- If the root of T_1 is an order-internal node, by removing in T_1 all the children of its root and connecting its root to all grandchildren, we are left with an extended-$(k, m, km + d - 1)$-ary tree of order $n - 1$ which is counted by $B_{n-1,km+d-1}$.

Hence, the proof is complete. □

For the ranking and unranking algorithms we need to compute $B_{n,k}$ in advance. This can be computed either by Theorems 4.1 or 4.2; in either cases these computations can be performed in time $O(kn)$.

Now, with regard to the above theorems and definitions, in Figure 4 we present the algorithm to compute the rank. In this algorithm, at first, we compute the rank for any given codeword in lexicographic ordering, then we subtract the result from the total number of trees in $T_{k,m}(n)$ to get the rank in reverse-lexicographic ordering. This will be equivalent to the rank of the corresponding tree in B-order. In this algorithm, array Z is the codeword (z-sequence), and $NegR$ is a variable which stores the number of codewords less than Z in lexicographic ordering.

For computing the rank of a tree T with a z-sequence stored in the array Z, we have to enumerate trees generated before T. Since these trees, in B-order, have codewords which are in reverse-lexicographic ordering, we enumerate the codewords less than T's corresponding codeword and then we subtract it from $C_{k,m}(n) = B_{n,k}$.

The number of codewords whose first element is greater than $Z[1]$ is equal to the number of all codewords starting with a value between $Z[1] + 1$ and k. Let us assume that $Z[1]$ corresponds to the i^{th} subtree T_i of T (i is not necessarily equal to 1). Obviously all the left siblings of T_i (if any) are leaves (because $Z[1]$ corresponds to the first order-internal node), and they should remain leaves also in the successive codewords. So for counting the successive codewords, if just ignore (remove) the node corresponding to $Z[1]$ and all its left siblings (which are leaves), then we are left with an extended-$(k, m, k - Z[1])$-ary tree of order n which is counted by $B_{n,k-Z[1]}$.

Now, the number of codewords with the first element equal to $Z[1]$ and the second element greater than $Z[2]$ is equal to $B_{n-1,k+km-Z[2]}$; since by

```
**Procedure** *ZseqBOrderUnRank*($k$, $m$, $n$, $R$: **integer**);
**Var** $NegR$: **integer**;
**begin**
    $NegR := B_{n,k} + 1 - NegR$;
    **for** $i := 1$ **to** $n$ **do**
    **begin**
        $j := \{ \max d \mid B_{n+1-i,d} < R\}$;
        $Z[i] := k + (i-1)km - j$;
        $NegR := NegR - B_{n-i+1,j}$;
    **end**;
**end**;
```

Figure 5. Unranking algorithm for (k,m)-ary trees in B-order. .

removing the node corresponding to $Z[1]$ and its children and connecting all its grandchildren to the root of the tree we obtain another tree with degree of root $k+km$ and in that tree, as in the previous case, all the left siblings of the node corresponding to $Z[2]$ are leaves which again by removing them we are left with an extended-$(k,m,k+km-Z[2])$-ary tree of order $n-1$ which is counted by $B_{n-1,k+km-Z[2]}$. Similarly the number of codewords whose first $(j-1)$ elements are equal to $Z[1],Z[2],\ldots,Z[j-1]$ and the j^{th} element is greater than $Z[j]$ is equal to: $B_{n+1-j,k+(j-1)km-Z[j]}$.

We now discuss the time complexity of this algorithm. Obviously the time complexity of an algorithm for computing $B_{n,k}$ is $O(kn)$, but this algorithm is executed just once before calling the ranking algorithm. So we can assume $B_{n,k}$ values are already available in a two dimensional array. Therefore with this assumption, it is easy to see that the ranking algorithm given in Figure 4 is $O(n)$ since the input codeword is scanned just once.

To design an unranking algorithm, for a given rank R, we have to find a z-sequence and store it in an array Z corresponding to the given rank. In this case, since for (k,m)-ary trees in B-order, the corresponding codewords (z-sequences) are in reverse-lexicographic ordering, we convert this rank (R) to the corresponding lexicographic ordering rank, and we denote it by $NegR$. This can be easily done as follows:

$$NegR = C_{k,m}(n) - R + 1 = B_{n,k} + 1 - R.$$

The main idea of the rest of the unranking algorithm is doing reversely what we did in the ranking algorithm, applying the same arguments. Here, in the unranking algorithm, for i from 1 to n, at any iteration we find the maximum $j > 0$ such that $B_{n+1-i,j} < R$ and by subtracting it from $k+(i-1)km$ we calculate $Z[i]$, then we subtract $B_{n,j}$ from the current rank $NegR$ to update the rank and continue to the next element until the whole codeword is built. This unranking algorithm is given with more details in Figure 5. In this algorithm R is the input and $NegR$ is the corresponding rank in lexicographic ordering. The generated z-sequence is stored in the array Z.

As mentioned before, it is assumed that the array $B_{n,k}$ is computed in advance. Since any iteration of the main loop of unranking algorithm can be performed in time $O(\log n)$ by using a simple binary search, the overall cost of the unranking algorithm is $O(n \log n)$.

5. Conclusion

(k, m)-ary trees is first introduced by Du and Liu [9]. They proved that the total number of (k, m)-ary trees of order n is equal to $C_{k,m}(n)$ which is called (k, m)-*Catalan number of order* n. In this work, we used an encoding similar to *Zaks' encoding* [31] for representing the class of (k, m)-ary trees of order n (denoted as $T_{k,m}(n)$). This encoding is used for generating (k, m)-ary trees in B-order in which the corresponding codewords are generated in reverse-lexicographic ordering (reverse manner of lexicographic ordering). Generating algorithm has constant average time and $O(n)$ time complexity in the worst case. Also new ranking and unranking algorithms were presented with time complexities of $O(n)$ and $O(n \log n)$, receptively. We emphasize that in $T_{k,m}(n)$ trees, n is the number of none-zero degree nodes in odd levels which is much more less than the total number of nodes in tree $((k + 1)(nm + 1))$. For the future work, designing the generation, ranking and unranking algorithms for this class of trees in A-order and also in Minimal change ordering are suggested.

References

[1] A. Ahmadi-Adl, A. Nowzari-Dalini, and H. Ahrabian, Ranking and unranking algorithms for loopless generation of t-ary trees, *Logic Journal of IGPL* **19** (2011), 33-43.

[2] H. Ahrabian and A. Nowzari-Dalini, On the generation of binary trees in A-order. *International Journal of Computer Mathematics* **71** (1999), 351-357.

[3] H. Ahrabian and A. Nowzari-Dalini, Parallel generation of binary trees in A-order. *Parallel Computing* **31** (2005), 948-955.

[4] H. Ahrabian and A. Nowzari-Dalini, Parallel Generation of t-ary trees in A-order, *Computer Journal* **50** (2007), 581-588.

[5] M. Amani, A. Nowzari-Dalini, and H. Ahrabian, Generation of neuronal trees by a new three letters encoding, *Computing and Informatics Journal* **33 (6)** (2014), 1428-1450.

[6] M. Amani, and A. Nowzari-Dalini, Ranking and Unranking Algorithm for Neuronal Trees in B-order, *Journal of Physical Sciences* **20** (2015), 19-34.

[7] M. Amani, and A. Nowzari-Dalini, Generation, Ranking and Unranking of Ordered Trees with Degree Bounds, in Proc. DCM 2015, *Electronic Proceedings in Theoretical Computer Science* **204** (2015), 31-45.

[8] M. Amani, Gap terminology and related combinatorial properties for AVL trees and Fibonacci-isomorphic trees, *AKCE International Journal of Graphs and Combinatorics*, **15 (1)** (2018), 14-21.

[9] R. R. X. Du and F. Liu, (k, m)-Catalan numbers and hook length polynomials for plane trees, *European Journal of Combinatorics* **28** (2007), 1312-1321.

[10] S. Durocher, P.C. Li, D. Mondal, F. Ruskey, and A. Williams, Cool-lex order and k-ary Catalan structures, *Journal of Discrete Algorithms*, **16** (2012), 287-307.

[11] S. Heubach, N. Li, and T. Mansour, Staircase tilings and k-Catalan structures. *Discrete Mathematics* **308** (2008), 5954-5964.

[12] J. I. Joichi, D. E. White, and S. G. Williamson, Combinatorial Gray codes, *SIAM J. Comput.* **9** (1980), 130-141.

[13] J. F. Korsh and P. LaFollette, Loopless generation of Gray codes for k-ary trees. *Information Processing letters* **70** (1999), 7-11.

[14] D. L. Kreher and D. R. Stinson, *Combinatorial Algorithms*, 2nd ed., CRC Press, New York, 1999.

[15] L. Li, Ranking and unranking AVL trees. *SIAM Journal of Computing* **15** (1986), 1025-1035.

[16] J. Pallo, Enumerating, ranking and unranking binary trees, *Comput. J.* **29** (1986), 171-175.

[17] J. Pallo, Generating trees with n nodes and m leaves, *International Journal of Computer Mathematics* **21** (1987), 133-144.

[18] J. Pallo, A simple algorithm for generating neuronal dendritic trees, *Computer Methods and Programs in Biomedicine* **33** (1990), 165-169.

[19] D. Roelants van Baronaigien, A loopless algorithm for generating binary trees sequences, *Inform. Process. Lett.* **39** (1991), 189–194.

[20] D. Roelants van Baronaigien and F. Ruskey, Generating t-ary trees in A-order, *Inform. Process. Lett.* **27** (1988), 205–213.

[21] , D. Roelants van Baronaigien, A loopless Gray code algorithm for listing k-ary trees, *J. Algorithms* **35** (2000), 100–107.

[22] F. Ruskey, Generating t-ary trees lexicographically, *SIAM Journal of Computing* **7** (1978), 424-439.

[23] E. Seyedi-Tabari, H. Ahrabian, and A. Nowzari-Dalini, A new algorithm for generation of different types of RNA. *International Journal of Computer Mathematics***87** (2010), 1197-1207.

[24] M. Solomon and R. A. Finkel, A note on enumerating binary trees, *J. ACM* **27** (1980), 3-5.

[25] R. P. Stanley, *Enumerative Combinatorics, vol. 2*, Cambridge University Press, Cambridge, UK, 1999.

[26] I. Stojmenovic, Listing combinatorial objects in parallel, *The International Journal of Parallel, Emergent and Distributed Systems*, **21** (2006), 127–146.

[27] V. Vajnovszki, Listing and random generation of neuronal trees coded by six letters. *The Automation, Computers, and Applied Mathematics* **4** (1995), 29-40.

[28] V. Vajnovszki and J. Pallo, Generating binary trees in A-order from codewords defined on four-letter alphabet. *Journal of Information and Optimization Science* **15** (1994), 345-357.

[29] R. Wu, J. Chang, and Y. Wang, A linear time algorithm for binary tree sequences transformation using left-arm and right-arm rotations. *Theoretical Computer Science* **335** (2006), 303-314.

[30] R. Wu, J. Chang, and C. Chang, Ranking and unranking of non-regular trees with a prescribed branching sequence. *Mathematical and Computer Modelling* **53** (2011), 1331-1335.

[31] S. Zaks, Lexicographic generation of ordered trees. *Theoretical Computer Science* **10** (1980), 63-82.

(M. Amani) Department of Computer Science (IDI), Norwegian University of Science and Technology (NTNU), Trondheim, Norway

(M. Amani,A. Nowzari-Dalini) School of Mathematics, Statistics, and Computer Science, Colleague of Science, University of Tehran, Tehran, Iran
E-mail address: mahdi.amani@ntnu.no
E-mail address: nowzari@ut.ac.ir