Proceedings of the Jangjeon Mathematical Society
28 (2025), No. 1, pp. 53 - 71

www.jangjeon.or.kr
http://dx.doi.org/10.17777/pjms2025.28.1.53

COMBINATORIAL GENERATION ALGORITHMS FOR DISCRETE STRUCTURES ASSOCIATED WITH THE FUBINI NUMBERS

YURIY SHABLYA, VADIM POLYUGA, AND DMITRY KRUCHININ

Abstract. This article demonstrates the application of the method based on AND/OR trees in order to obtain new combinatorial generation algorithms for combinatorial objects associated with the Fubini numbers. Using three different formulas for calculating the Fubini numbers that satisfy the restrictions of the method applied, the corresponding structures of AND/OR trees were constructed. The number of variants of the constructed AND/OR trees is the same as the value of the corresponding Fubini number. This fact made it possible to determine bijection rules between the combinatorial sets associated with the Fubini numbers and the variants of the obtained AND/OR tree structures. Applying the method based on AND/OR trees, we develop three different sets of algorithms for ranking and unranking the combinatorial sets associated with the Fubini numbers. We also show that the developed algorithms have polynomial time complexity and differ in the ordering of the elements of the combinatorial set.

2010 Mathematics Subject Classification. 05A18, 06A07, 11B73, 11K31, 68W30.

Keywords and phrases. Combinatorial generation, Fubini number, Ordered Bell number, AND/OR tree, Ranking, Unranking.

1. Introduction

Special numbers are numbers that often appear in various mathematical contexts. This article is devoted to the study of special numbers called the Fubini numbers (or the ordered Bell numbers). The Fubini numbers, denoted by F_n, are the following sequence of integers for $n \geq 0$ (sequence A000670 in OEIS [1]):

$$1, 1, 3, 13, 75, 541, 4683, 47293, 545835, 7087261, 102247563, \ldots$$

The Fubini numbers are important in discrete mathematics because they have a whole set of different combinatorial interpretations. For example, the Fubini number F_n equals to the number of:

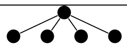
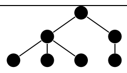
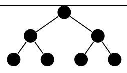
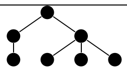
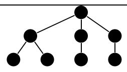
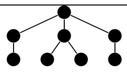
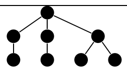
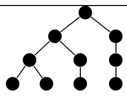
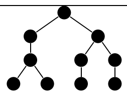
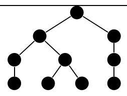
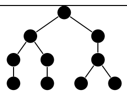
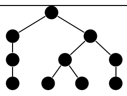
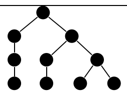
- Dense rankings of n items that can be represented as sequences of the form $(a_1, \ldots, a_n)$ where each a_i shows the place of the i-th item in the ranking of n items (*cf.* [2]);
- Weak orders on n labeled elements (*cf.* [3]);

The reported study was supported by the Russian Science Foundation (project no. 22-71-10052).

- Ordered partitions of n elements that can be represented as sequences of the form $(s_1, s_2, \ldots)$ where each s_i is a subset of a set of n elements (*cf.* [4, 5]);
- Plane trees with $n+1$ leaves where root-to-leaf paths have the same length and the number of nodes at distance i from the root must be strictly smaller than the number of nodes at distance $i+1$ (*cf.* [6]).

Table 1 presents examples of combinatorial sets associated with the Fubini number $F_3 = 13$. The Fubini numbers are also related to many other special numbers (*cf.* [7, 8, 9, 10, 11, 12]).

TABLE 1. Examples of combinatorial sets associated with the Fubini number $F_3 = 13$

Dense ranking of 3 items	Weak order on 3 labeled elements	Ordered partition of 3 elements	Plane tree with 4 leaves
$1,1,1$	$a=b=c$	$\{a,b,c\}$	
$1,1,2$	$a=b>c$	$\{a,b\},\{c\}$	
$1,2,1$	$a=c>b$	$\{a,c\},\{b\}$	
$2,1,1$	$b=c>a$	$\{b,c\},\{a\}$	
$1,2,2$	$a>b=c$	$\{a\},\{b,c\}$	
$2,1,2$	$b>a=c$	$\{b\},\{a,c\}$	
$2,2,1$	$c>a=b$	$\{c\},\{a,b\}$	
$1,2,3$	$a>b>c$	$\{a\},\{b\},\{c\}$	
$1,3,2$	$a>c>b$	$\{a\},\{c\},\{b\}$	
$2,1,3$	$b>a>c$	$\{b\},\{a\},\{c\}$	
$2,3,1$	$c>a>b$	$\{c\},\{a\},\{b\}$	
$3,1,2$	$b>c>a$	$\{b\},\{c\},\{a\}$	
$3,2,1$	$c>b>a$	$\{c\},\{b\},\{a\}$	

Since the values of the Fubini numbers F_n grow rapidly as the parameter n increases, it becomes difficult to organize the process of constructing objects of combinatorial sets associated with the Fubini numbers. Combinatorial generation algorithms make it possible to obtain a complete set of elements for a given discrete set (*cf.* [13]). Note that in the field of combinatorial generation there are no studies that address the issue of developing combinatorial generation algorithms for combinatorial sets associated with the Fubini numbers. Therefore, the main goal of this article is to develop new combinatorial generation algorithms for ranking and unranking combinatorial sets associated with the Fubini numbers.

2. Materials and Methods

Combinatorial generation is a scientific direction that studies combinatorial sets and algorithms for their generation. For example, it is possible to number the elements of a given combinatorial set based on some their ordering. In this case, the number of an element in the ordering is called a rank, and the numbering process is called a ranking. The inverse operation is called unranking (that is, the generation of an element of a given combinatorial set using its rank). In addition, there is a class of algorithms for the exhaustive generation of all elements of a given combinatorial set. The application of such algorithms makes it possible to generate fast combinatorial objects belonging to the considered set of discrete structures.

For various combinatorial sets there are many combinatorial generation algorithms that are ready-to-use (for more details, see [14, 15]). There are also several general methods that can be applied to develop new combinatorial generation algorithms (such as the backtracking method [13], the ECO method [16], the Flajolet method [17], and others). Each such method has its own advantages and disadvantages. In this paper, to develop new combinatorial generation algorithms, we study the application of the method based on AND/OR trees since it allows the development of ranking and unranking algorithms and requires only the cardinality function of a given combinatorial set (*cf.* [18, 19, 21]).

The application of the considered method based on AND/OR trees is limited by the need to have an expression for the cardinality function of a given combinatorial set that can contain only the following operations and operands: positive integers, addition and multiplication operations, and recursive calls. In our previous work [20], we showed that the following formulas are known for calculating the Fubini numbers, and each of them satisfies the requirements of the considered method for developing combinatorial generation algorithms:

- Based on the binomial coefficients C_n^k (*cf.* [4]):

$$F_n = \sum_{k=1}^{n} C_n^k F_{n-k}, \quad F_0 = 1, \tag{1}$$

where

$$C_n^k = C_{n-1}^k + C_{n-1}^{k-1}, \quad C_n^n = C_n^0 = 1;$$

- Based on the Stirling numbers of the second kind S_n^k (*cf.* [5]):

$$F_n = \sum_{k=1}^{n} k! S_n^k, \tag{2}$$

where

$$S_n^k = kS_{n-1}^k + S_{n-1}^{k-1}, \quad S_n^n = S_n^1 = 1,$$
$$k! = k \cdot (k-1)!, \quad 0! = 1;$$

- Based on the Eulerian numbers of the first kind E_n^k (*cf.* [22]):

$$F_n = \sum_{k=0}^{n-1} 2^k E_n^k, \tag{3}$$

where

$$E_n^k = (k+1)E_{n-1}^k + (n-k)E_{n-1}^{k-1}, \quad E_n^{n-1} = E_n^0 = 1,$$
$$2^k = 2 \cdot 2^{k-1}, \quad 2^0 = 1.$$

Since each of the presented expressions for calculating the Fubini numbers satisfies the requirements of the considered method, it is possible to construct corresponding AND/OR tree structures.

AND/OR tree structures have two kinds of nodes:

- an OR node corresponds to the addition operation in the cardinality function expression and shows the union of combinatorial subsets;
- an AND node (represented by a node whose edges to child nodes are connected by an additional arc) corresponds to the multiplication operation in the cardinality function expression and shows the Cartesian product of combinatorial subsets.

In addition, each positive integer in the cardinality function corresponds to an OR node whose number of child nodes is equal to the value of this integer. A subtree of an AND/OR tree node (including one obtained by a recursive call) is represented in the form of a triangle node. Examples of AND/OR tree structures are presented in the figures of this article.

Each AND/OR tree structure has a fixed number of its variants that is equal to the value of the corresponding cardinality function. To obtain a variant of an AND/OR tree, it is necessary to leave only one child node for each OR node (the remaining child nodes of the OR node and their subtrees are deleted).

Next, we consider in detail the representation of combinatorial sets associated with the Fubini numbers in the form of the set of AND/OR tree variants. In this case, the number of variants of the obtained AND/OR tree structures is equal to the value of the corresponding Fubini number.

3. Main results

In this section, using the above three formulas for calculating the Fubini numbers, we construct the corresponding structures of AND/OR trees and determine bijection between the combinatorial sets associated with the Fubini numbers and the variants of the obtained AND/OR trees. Then, we develop three different sets of algorithms for ranking and unranking the combinatorial sets associated with the Fubini numbers.

3.1. **AND/OR tree for F_n based on formula** (1)**.** Firstly, applying formula (1), we construct the corresponding AND/OR tree structure for F_n (see Figure 1).

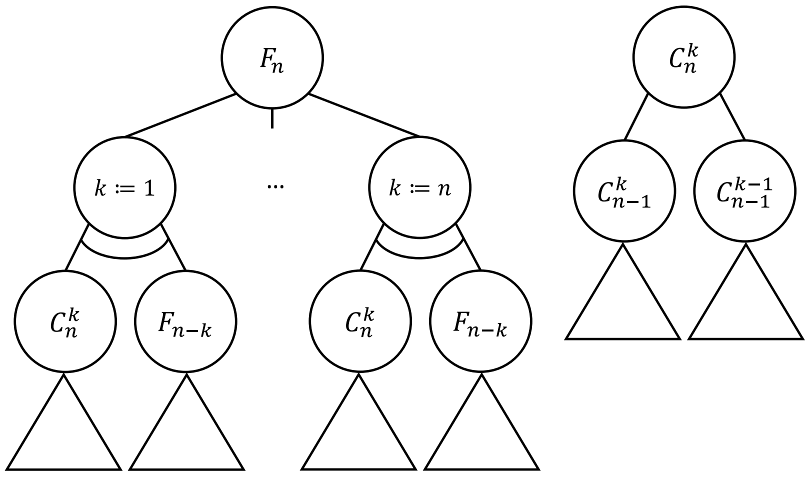


FIGURE 1. An AND/OR tree for F_n based on formula (1) and its subtree for C_n^k

Figure 2 presents an example of this AND/OR tree structure for $n = 3$. The set of all variants of this AND/OR tree structure consists of $F_3 = 13$ elements.

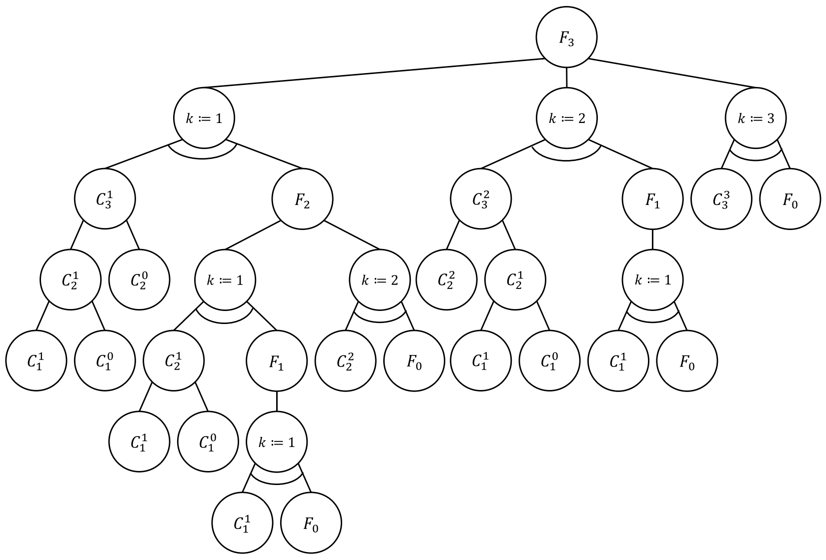


FIGURE 2. An AND/OR tree for F_3 based on formula (1)

It is proposed to encode a variant of the AND/OR tree for F_n based on formula (1) by a sequence $v = (K, v_C, v_F)$, where:

- an empty sequence $v = ()$ corresponds to the selection of a leaf F_0;
- K corresponds to the selected value of k in the AND/OR tree for F_n;
- v_C corresponds to the variant of the subtree of the node C_n^k that is encoded by a sequence $v_C = (v_{C1}, v_{C2}, \ldots)$, where:
  - an empty sequence $v_C = ()$ corresponds to the selection of a leaf C_n^n or C_n^0;
  - $v_{Ci} = 0$ corresponds to the selection of the left child of the node at i-th level of the subtree of the node C_n^k;
  - $v_{Ci} = 1$ corresponds to the selection of the right child of the node at i-th level of the subtree of the node C_n^k;
- v_F corresponds to the variant of the subtree of the node F_{n-k}.

Theorem 3.1. *There is a bijection between the set of weak orders on n labeled elements and the set of variants of the AND/OR tree for F_n from Figure 1.*

A bijection between the weak orders on n labeled elements and the variants of the AND/OR tree for F_n from Figure 1 is defined by the following rules:

- a leaf F_0 corresponds to an empty weak order;
- the selected child of the OR node F_n determines the number k of elements that together have the last place in the weak order on n labeled elements;
- the subtree of the node C_n^k determines which k elements out of n elements together have the last place in the weak order on n labeled elements:
  - $v_C = ()$ for a leaf C_n^n means that all n elements were selected;
  - $v_C = ()$ for a leaf C_n^0 means that all n elements were unselected;
  - $v_{Ci} = 0$ for a node C_n^k means that the n-th element was unselected;
  - $v_{Ci} = 1$ for a node C_n^k means that the n-th element was selected;
- the subtree of the node F_{n-k} determines the weak order on remaining $(n-k)$ labeled elements.

Table 2 presents an example of representing weak orders on 3 labeled elements by variants of the AND/OR tree for F_3 from Figure 2. Similarly, it is possible to derive a bijection between the set of variants of the AND/OR tree for F_n from Figure 1 and any other combinatorial set associated with the Fubini numbers.

Applying the obtained AND/OR tree structure for F_n, we develop algorithms for ranking (Algorithm 1) and unranking (Algorithm 2) its variants. Table 2 presents an example of ranking the variants of the AND/OR tree for F_3 from Figure 2. Thus, using the developed ranking algorithm and bijection rules, it is possible to number a given combinatorial object belonging to the combinatorial set associated with Fubini numbers. In addition, using the developed unranking algorithm and bijection rules, it is possible to generate objects of such a combinatorial set (generation of objects with given rank values or exhaustive generation by considering all rank values).

TABLE 2. Example of representing weak orders on 3 labeled elements by variants of AND/OR tree for F_3 from Figure 2

Weak order on 3 labeled elements	Variant of AND/OR tree for F_3 from Figure 2	Rank
$a = b = c$	$(3, (), ())$	12
$a = b > c$	$(1, (1), (2, (), ()))$	8
$a = c > b$	$(1, (0, 1), (2, (), ()))$	7
$b = c > a$	$(1, (0, 0), (2, (), ()))$	6
$a > b = c$	$(2, (1, 1), (1, (), ()))$	11
$b > a = c$	$(2, (1, 0), (1, (), ()))$	10
$c > a = b$	$(2, (0), (1, (), ()))$	9
$a > b > c$	$(1, (1), (1, (1), (1, (), ())))$	5
$a > c > b$	$(1, (0, 1), (1, (1), (1, (), ())))$	4
$b > a > c$	$(1, (1), (1, (0), (1, (), ())))$	2
$c > a > b$	$(1, (0, 1), (1, (0), (1, (), ())))$	1
$b > c > a$	$(1, (0, 0), (1, (1), (1, (), ())))$	3
$c > b > a$	$(1, (0, 0), (1, (0), (1, (), ())))$	0

Algorithm 1: Algorithms for ranking a variant of the AND/OR tree for F_n based on formula (1)

```
Rank_F ($v = (K, v_C, v_F)$, $n$)
begin
    if $n = 0$ then $r := 0$
    else
        $sum := 0$
        for $k := 1$ to $K - 1$ do
            $sum := sum + C_n^k F_{n-k}$
        end
        $l_1 :=$ Rank_C ($v_C$, $n$, $K$)
        $l_2 :=$ Rank_F ($v_F$, $n - K$)
        $r := sum + l_1 + C_n^K \cdot l_2$
    end
    return $r$
end

Rank_C ($v = (v_1, v_2, \ldots)$, $n$, $k$)
begin
    if $k = n$ or $k = 0$ then $r := 0$
    else
        if $v_1 = 0$ then $r :=$ Rank_C (($v_2, \ldots$), $n - 1$, $k$)
        else $r := C_{n-1}^k +$ Rank_C (($v_2, \ldots$), $n - 1$, $k - 1$)
    end
    return $r$
end
```

Algorithm 2: Algorithms for unranking a variant of the AND/OR tree for F_n based on formula (1)

```
Unrank_F ($r$, $n$)
begin
    if $n = 0$ then $v := ()$
    else
        for $k := 1$ to $n$ do
            $sum := C_n^k F_{n-k}$
            if $r < sum$ then
                $K := k$
                break
            end
            $r := r - sum$
        end
        $l_1 := r \mod C_n^K$
        $l_2 := \left\lfloor \frac{r}{C_n^K} \right\rfloor$
        $v_C :=$ Unrank_C ($l_1$, $n$, $K$)
        $v_F :=$ Unrank_F ($l_2$, $n - K$)
        $v := (K, v_C, v_F)$
    end
    return $v$
end

Unrank_C ($r$, $n$, $k$)
begin
    if $k = n$ or $k = 0$ then $v := ()$
    else
        if $r < C_{n-1}^k$ then $v :=$concat ((0), Unrank_C ($r$, $n - 1$, $k$))
        else $v :=$concat ((1), Unrank_C ($r - C_{n-1}^k$, $n - 1$, $k - 1$))
    end
    return $v$
end
```

3.2. **AND/OR tree for F_n based on formula** (2)**.** Next, applying formula (2), we construct the corresponding AND/OR tree structure for F_n (see Figure 3).

Figure 4 presents an example of this AND/OR tree structure for $n = 3$. The set of all variants of this AND/OR tree structure consists of $F_3 = 13$ elements.

It is proposed to encode a variant of the AND/OR tree for F_n based on formula (2) by a sequence $v = (K, v_{k!}, v_S)$, where:

- an empty sequence $v = ()$ corresponds to the selection of a leaf F_0;
- K corresponds to the selected value of k in the AND/OR tree for F_n;
- $v_{k!}$ corresponds to the variant of the subtree of the node $k!$ that is encoded by a sequence $v_{k!} = (v_1, \ldots, v_k)$ of labels of the selected children of the OR nodes k;

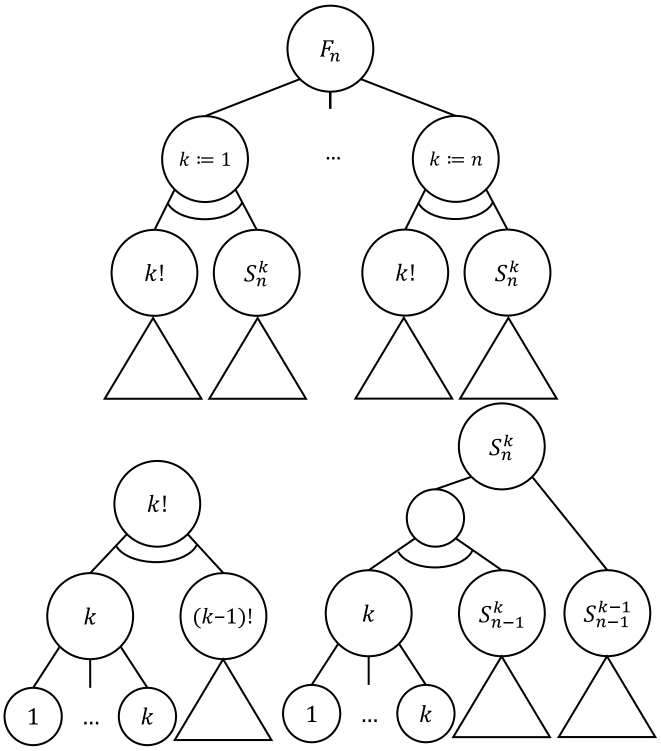


FIGURE 3. An AND/OR tree for F_n based on formula (2) and its subtrees for $k!$ and S_n^k

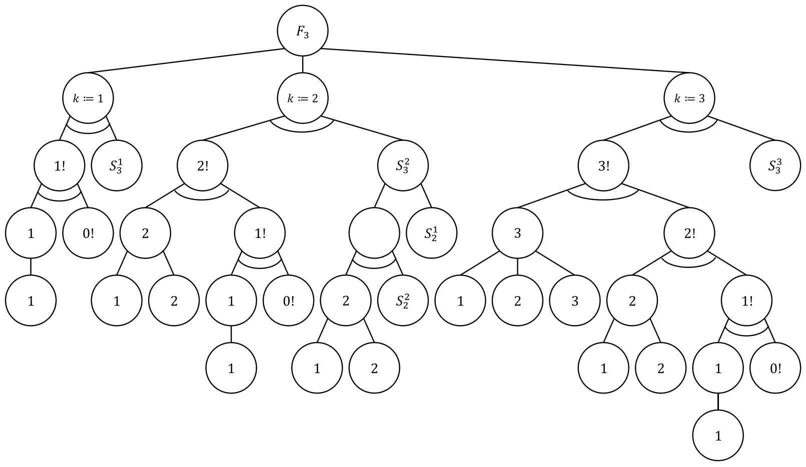


FIGURE 4. An AND/OR tree for F_3 based on formula (2)

- v_S corresponds to the variant of the subtree of the node S_n^k that is encoded by a sequence $v_S = (K_S, v_{SS})$, where:
  - an empty sequence $v_S = ()$ corresponds to the selection of a leaf S_n^n or S_n^1;
  - if $K_S = 0$, the right child of the OR node S_n^k is selected and v_{SS} corresponds to the variant of the subtree of the node S_{n-1}^{k-1};
  - otherwise, K_S corresponds to the selected value of k in the subtree of the node S_n^k and v_{SS} corresponds to the variant of the subtree of the node S_{n-1}^k.

Theorem 3.2. *There is a bijection between the set of weak orders on n labeled elements and the set of variants of the AND/OR tree for F_n from Figure 3.*

A bijection between the weak orders on n labeled elements and the variants of the AND/OR tree for F_n from Figure 3 is defined by the following rules:

- a leaf F_0 corresponds to an empty weak order;
- the selected child of the OR node F_n determines the number k of subsets of elements where all elements of each subset have the same place in the weak order on n labeled elements;
- the subtree of the node S_n^k determines the partition of a set of n elements into k non-empty subsets:
  - $v_S = ()$ for a leaf S_n^n means that n elements were divided in n singleton subsets;
  - $v_S = ()$ for a leaf S_n^1 means that n elements were combined into one set;
  - $K_S = 0$ for a node S_n^k means that the n-th element alone forms the k-th subset, and the remaining $(n-1)$ elements form the remaining $(k-1)$ subsets;
  - $K_S > 0$ for a node S_n^k means that the n-th element together with others forms the K_S-th subset together with other elements, and the remaining $(n-1)$ elements form the remaining k subsets;
- the subtree of the node $k!$ determines the order on k subsets of elements as a permutation of k elements:
  - a leaf $0!$ corresponds to an empty permutation;
  - v_1 for a node $k!$ means that the k-th element occupies the v_1-th position in the permutation of k elements, and the remaining $(k-1)$ elements occupy the remaining $(k-1)$ positions.

Table 3 presents an example of representing weak orders on 3 labeled elements by variants of the AND/OR tree for F_3 from Figure 4. Similarly, it is possible to derive a bijection between the set of variants of the AND/OR tree for F_n from Figure 3 and any other combinatorial set associated with the Fubini numbers.

Applying the obtained AND/OR tree structure for F_n, we develop algorithms for ranking (Algorithm 3) and unranking (Algorithm 4) its variants. Table 3 presents an example of ranking the variants of the AND/OR tree for F_3 from Figure 4.

Algorithm 3: Algorithms for ranking a variant of the AND/OR tree for F_n based on formula (2)

```
Rank_F ($v = (K, v_{k!}, v_S)$, $n$)
begin
    if $n = 0$ then $r := 0$
    else
        $sum := 0$
        for $k := 1$ to $K - 1$ do
            $sum := sum + k! S_n^k$
        end
        $l_1 :=$ Rank_K ($v_{k!}$, $K$)
        $l_2 :=$ Rank_S ($v_S$, $n$, $K$)
        $r := sum + l_1 + K! \cdot l_2$
    end
    return $r$
end

Rank_K ($v = (v_1, \ldots, v_k)$, $k$)
begin
    if $k = 0$ then $r := 0$
    else
        $l_1 := v_1 - 1$
        $l_2 :=$ Rank_K (($v_2, \ldots, v_k$), $k - 1$)
        $r := l_1 + k \cdot l_2$
    end
    return $r$
end

Rank_S ($v = (K_S, v_{SS})$, $n$, $k$)
begin
    if $k = n$ or $k = 1$ then $r := 0$
    else
        if $K_S > 0$ then
            $l_1 := K_S - 1$
            $l_2 :=$ Rank_S ($v_{SS}$, $n - 1$, $k$)
            $r := l_1 + k \cdot l_2$
        end
        else
            $sum := k S_{n-1}^k$
            $l_3 :=$ Rank_S ($v_{SS}$, $n - 1$, $k - 1$)
            $r := sum + l_3$
        end
    end
    return $r$
end
```

Algorithm 4: Algorithms for unranking a variant of the AND/OR tree for F_n based on formula (2)

```
Unrank_F ($r$, $n$)
begin
    if $n = 0$ then $v := ()$
    else
        for $k := 1$ to $n$ do
            $sum := k!S_n^k$
            if $r < sum$ then
                $K := k$
                break
            end
            $r := r - sum$
        end
        $l_1 := r \mod K!$
        $l_2 := \lfloor \frac{r}{K!} \rfloor$
        $v_{k!} :=$ Unrank_K ($l_1$, $K$)
        $v_S :=$ Unrank_S ($l_2$, $n$, $K$)
        $v := (K, v_{k!}, v_S)$
    end
    return $v$
end

Unrank_K ($r$, $k$)
begin
    if $k = 0$ then $v := ()$
    else
        $l_1 := r \mod k$
        $l_2 := \lfloor \frac{r}{k} \rfloor$
        $v :=$concat (($l_1 + 1$), Unrank_K ($l_2$, $k - 1$))
    end
    return $v$
end

Unrank_S ($r$, $n$, $k$)
begin
    if $k = n$ or $k = 1$ then $v := ()$
    else
        $sum := kS_{n-1}^k$
        if $r < sum$ then
            $l_1 := r \mod k$
            $l_2 := \lfloor \frac{r}{k} \rfloor$
            $K_S := l_1 + 1$
            $v_{SS} :=$ Unrank_S ($l_2$, $n - 1$, $k$)
        end
        else
            $l_3 := r - sum$
            $K_S := 0$
            $v_{SS} :=$ Unrank_S ($l_3$, $n - 1$, $k - 1$)
        end
        $v := (K_S, v_{SS})$
    end
    return $v$
end
```

TABLE 3. Example of representing weak orders on 3 labeled elements by variants of AND/OR tree for F_3 from Figure 4

Weak order on 3 labeled elements	Variant of AND/OR tree for F_3 from Figure 4	Rank
$a = b = c$	$(1, (1), ())$	0
$a = b > c$	$(2, (2, 1), (0, ()))$	6
$a = c > b$	$(2, (2, 1), (1, ()))$	2
$b = c > a$	$(2, (1, 1), (2, ()))$	3
$a > b = c$	$(2, (2, 1), (2, ()))$	4
$b > a = c$	$(2, (1, 1), (1, ()))$	1
$c > a = b$	$(2, (1, 1), (0, ()))$	5
$a > b > c$	$(3, (3, 2, 1), ())$	12
$a > c > b$	$(3, (2, 2, 1), ())$	11
$b > a > c$	$(3, (3, 1, 1), ())$	9
$c > a > b$	$(3, (1, 2, 1), ())$	10
$b > c > a$	$(3, (2, 1, 1), ())$	8
$c > b > a$	$(3, (1, 1, 1), ())$	7

3.3. **AND/OR tree for F_n based on formula** (3). Finally, applying formula (3), we construct the corresponding AND/OR tree structure for F_n (see Figure 5). Figure 6 presents an example of this AND/OR tree structure for $n = 3$. The set of all variants of this AND/OR tree structure consists of $F_3 = 13$ elements.

It is proposed to encode a variant of the AND/OR tree for F_n based on formula (3) by a sequence $v = (K, v_{2^k}, v_E)$, where:

- an empty sequence $v = ()$ corresponds to the selection of a leaf F_0;
- K corresponds to the selected value of k in the AND/OR tree for F_n;
- v_{2^k} corresponds to the variant of the subtree of the node 2^k that is encoded by a sequence $v_{2^k} = (v_1, \ldots, v_k)$ of labels of the selected children of the OR nodes 2;
- v_E corresponds to the variant of the subtree of the node E_n^k that is encoded by a sequence $v_E = (K_{E_1}, K_{E_2}, v_{EE})$, where:
  - an empty sequence $v_E = ()$ corresponds to the selection of a leaf E_n^{n-1} or E_n^0;
  - if $K_{E_1} = 1$, then the left child of the OR node E_n^k is selected, K_{E_2} corresponds to the label of the selected child of the OR node $(k + 1)$, v_{EE} corresponds to the variant of the subtree of the node E_{n-1}^k;
  - if $K_{E_1} = 2$, then the right child of the OR node E_n^k is selected, K_{E_2} corresponds to the label of the selected child of the OR node $(n - k)$, v_{EE} corresponds to the variant of the subtree of the node E_{n-1}^{k-1}.

Theorem 3.3. *There is a bijection between the set of weak orders on n labeled elements and the set of variants of the AND/OR tree for F_n from Figure 5.*

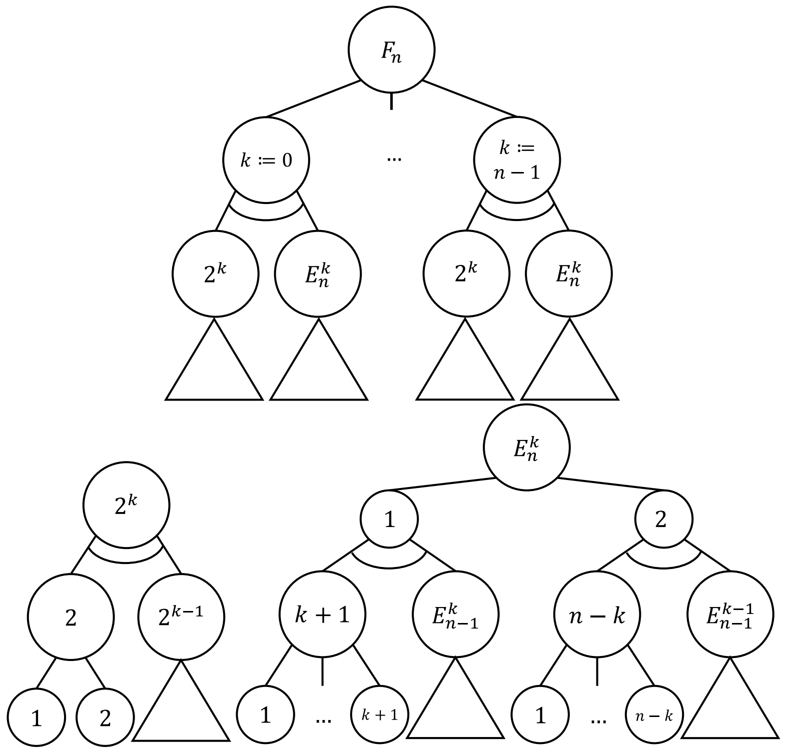


FIGURE 5. An AND/OR tree for F_n based on formula (3) and its subtrees for 2^k and E_n^k

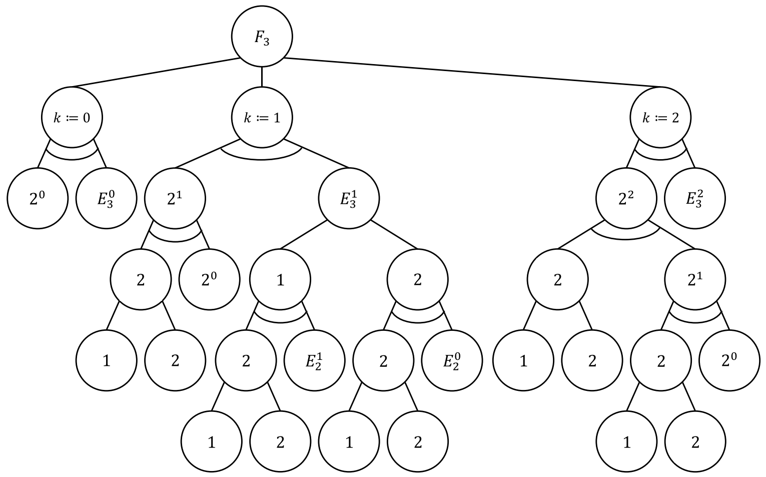


FIGURE 6. An AND/OR tree for F_3 based on formula (3)

A bijection between the weak orders on n labeled elements and the variants of the AND/OR tree for F_n from Figure 5 is defined by the following rules:

- a leaf F_0 corresponds to an empty weak order;
- the selected child of the OR node F_n determines the number k of ascents in the permutation of n elements that determines the order of the elements in the weak order on n labeled elements;
- the subtree of the node E_n^k determines the permutation of n elements with k ascents:
  - $v_E = ()$ for a leaf E_n^{n-1} means that n elements were arranged in ascending order;
  - $v_E = ()$ for a leaf E_n^0 means that n elements were arranged in descending order;
  - $K_{E_1} = 1$ for a node E_n^k means that the n-th element does not add an ascent in the permutation of the remaining $(n-1)$ elements, and K_{E_2} determines the position of the n-th element in the permutation (there are $(k+1)$ possible such positions);
  - $K_{E_1} = 2$ for a node E_n^k means that the n-th element add an ascent in the permutation of the remaining $(n-1)$ elements, and K_{E_2} determines the position of the n-th element in the permutation (there are $(n-k)$ possible such positions);
- the subtree of the node 2^k determines the need to equate the elements arranged at the place of the k-th ascent:
  - a leaf 2^0 corresponds to doing nothing with permutation;
  - $v_1 = 1$ for a node 2^k means that the elements arranged at the place of the k-th ascent have different places in the weak order;
  - $v_1 = 2$ for a node 2^k means that the elements arranged at the place of the k-th ascent have the same place in the weak order.

TABLE 4. Example of representing weak orders on 3 labeled elements by variants of AND/OR tree for F_3 from Figure 6

Weak order on 3 labeled elements	Variant of AND/OR tree for F_3 from Figure 6	Rank
$a = b = c$	$(2, (2, 2), ())$	12
$a = b > c$	$(2, (2, 1), ())$	10
$a = c > b$	$(1, (2), (1, 2, ()))$	4
$b = c > a$	$(1, (2), (2, 1, ()))$	6
$a > b = c$	$(2, (1, 2), ())$	11
$b > a = c$	$(1, (2), (2, 2, ()))$	8
$c > a = b$	$(1, (2), (1, 1, ()))$	2
$a > b > c$	$(2, (1, 1), ())$	9
$a > c > b$	$(1, (1), (1, 2, ()))$	3
$b > a > c$	$(1, (1), (2, 2, ()))$	7
$c > a > b$	$(1, (1), (1, 1, ()))$	1
$b > c > a$	$(1, (1), (2, 1, ()))$	5
$c > b > a$	$(0, (), ())$	0

Table 4 presents an example of representing weak orders on 3 labeled elements by variants of the AND/OR tree for F_3 from Figure 6. Similarly, it is possible to derive a bijection between the set of variants of the AND/OR tree for F_n from Figure 5 and any other combinatorial set associated with the Fubini numbers.

Applying the obtained AND/OR tree structure for F_n, we develop algorithms for ranking (Algorithm 5) and unranking (Algorithm 6) its variants.

Algorithm 5: Algorithms for ranking a variant of the AND/OR tree for F_n based on formula (3)

```
Rank_F ($v = (K, v_{2^k}, v_E)$, $n$)
begin
    if $n = 0$ then $r := 0$
    else
        $sum := 0$
        for $k := 0$ to $K - 1$ do
            $sum := sum + 2^k E_n^k$
        end
        $l_1 :=$ Rank_2K ($v_{2^k}$, $K$)
        $l_2 :=$ Rank_E ($v_E$, $n$, $K$)
        $r := sum + l_1 + 2^K \cdot l_2$
    end
    return $r$
end
Rank_2K ($v = (v_1, \ldots, v_k)$, $k$)
begin
    if $k = 0$ then $r := 0$
    else
        $l_1 := v_1 - 1$
        $l_2 :=$ Rank_2K (($v_2, \ldots, v_k$), $k - 1$)
        $r := l_1 + 2 \cdot l_2$
    end
    return $r$
end
Rank_E ($v = (K_{E_1}, K_{E_2}, v_{EE})$, $n$, $k$)
begin
    if $k = n - 1$ or $k = 0$ then $r := 0$
    else
        if $K_{E_1} = 1$ then
            $l_1 := K_{E_2} - 1$
            $l_2 :=$ Rank_E ($v_{EE}$, $n - 1$, $k$)
            $r := l_1 + (k + 1) \cdot l_2$
        end
        else
            $sum := (k + 1)E_{n-1}^k$
            $l_3 := K_{E_2} - 1$
            $l_4 :=$ Rank_E ($v_{EE}$, $n - 1$, $k - 1$)
            $r := sum + l_3 + (n - k) \cdot l_4$
        end
    end
    return $r$
end
```

Algorithm 6: Algorithms for unranking a variant of the AND/OR tree for F_n based on formula (3)

```
Unrank_F ($r$, $n$)
begin
    if $n = 0$ then $v := ()$
    else
        for $k := 0$ to $n - 1$ do
            $sum := 2^k E_n^k$
            if $r < sum$ then $K := k$, break
            $r := r - sum$
        end
        $l_1 := r \mod 2^K$
        $l_2 := \left\lfloor \frac{r}{2^K} \right\rfloor$
        $v_{2^k} :=$ Unrank_2K ($l_1$, $K$)
        $v_E :=$ Unrank_E ($l_2$, $n$, $K$)
        $v := (K, v_{2^k}, v_E)$
    end
    return $v$
end
Unrank_2K ($r$, $k$)
begin
    if $k = 0$ then $v := ()$
    else
        $l_1 := r \mod 2$
        $l_2 := \left\lfloor \frac{r}{2} \right\rfloor$
        $v :=$concat (($l_1 + 1$), Unrank_2K ($l_2$, $k - 1$))
    end
    return $v$
end
Unrank_E ($r$, $n$, $k$)
begin
    if $k = n - 1$ or $k = 0$ then $v := ()$
    else
        $sum := (k + 1)E_{n-1}^k$
        if $r < sum$ then
            $l_1 := r \mod k + 1$
            $l_2 := \left\lfloor \frac{r}{k+1} \right\rfloor$
            $K_{E_1} := 1$
            $K_{E_2} := l_1 + 1$
            $v_{EE} :=$ Unrank_E ($l_2$, $n - 1$, $k$)
        end
        else
            $r := r - sum$
            $l_3 := r \mod n - k$
            $l_4 := \left\lfloor \frac{r}{n-k} \right\rfloor$
            $K_{E_1} := 2$
            $K_{E_2} := l_3 + 1$
            $v_{EE} :=$ Unrank_E ($l_4$, $n - 1$, $k - 1$)
        end
        $v := (K_{E_1}, K_{E_2}, v_{EE})$
    end
    return $v$
end
```

4. Conclusion

As the main result of this article, we have obtained new combinatorial generation algorithms for combinatorial sets associated with the Fubini numbers using the method based on AND/OR trees. Since each of the presented expressions for calculating the Fubini numbers satisfies the requirements of the considered method, the corresponding AND/OR tree structures were constructed. The number of variants of the constructed AND/OR tree structures is equal to the value of the corresponding Fubini number. This fact made it possible to determine bijection rules between the weak orders on n labeled elements (this is one of the combinatorial sets associated with the Fubini numbers) and the variants of the AND/OR tree for F_n. Applying the obtained AND/OR tree structures for F_n, we develop three different sets of algorithms for ranking and unranking its variants. Thus, the combinatorial generation algorithms for the combinatorial sets associated with the Fubini numbers were obtained. The developed algorithms have polynomial time complexity and differ in the ordering of the elements of the combinatorial set. Table 5 presents the comparison of the obtained ways of ordering weak orders when using different ranking and unranking algorithms.

Table 5. Comparison of the obtained ways of ordering weak orders when using different ranking algorithms

Rank	Algorithm 2	Algorithm 4	Algorithm 6
0	$c > b > a$	$a = b = c$	$c > b > a$
1	$c > a > b$	$b > a = c$	$c > a > b$
2	$b > a > c$	$a = c > b$	$c > a = b$
3	$b > c > a$	$b = c > a$	$a > c > b$
4	$a > c > b$	$a > b = c$	$a = c > b$
5	$a > b > c$	$c > a = b$	$b > c > a$
6	$b = c > a$	$a = b > c$	$b = c > a$
7	$a = c > b$	$c > b > a$	$b > a > c$
8	$a = b > c$	$b > c > a$	$b > a = c$
9	$c > a = b$	$b > a > c$	$a > b > c$
10	$b > a = c$	$c > a > b$	$a = b > c$
11	$a > b = c$	$a > c > b$	$a > b = c$
12	$a = b = c$	$a > b > c$	$a = b = c$

Acknowledgment

This paper is dedicated to Professor Yilmaz SIMSEK on the occasion of his 60th anniversary. Some part of this paper has been presented at "The 13th Symposium on Generating Functions of Special Numbers and Polynomials and their Applications" (GFSNP 2023), dedicated to Professor Yilmaz SIMSEK on the Occasion of his 60th Anniversary, held in Antalya, Turkey on March 11–13, 2023.

References

[1] The On-Line Encyclopedia of Integer Sequences. Available online: https://oeis.org/ (accessed 1 May 2023).

[2] Pippenger, N. The hypercube of resistors, asymptotic expansions, and preferential arrangements. *Math. Mag.* **2010**, *83*, 331–346.

[3] Jacques, M.; Wong, D. Greedy universal cycle constructions for weak orders. *Lecture Notes in Comput. Sci.* **2020**, *12016*, 363–370.

[4] Gross, O.A. Preferential arrangements. *Amer. Math. Monthly.* **1962**, *69*, 4–8.

[5] Good, I.J. The number of orderings of n candidates when ties are permitted. *Fibonacci Quart.* **1975**, *13*, 11–18.

[6] Cayley, A. On the analytical forms called trees. Second Part. In *The Collected Mathematical Papers*; Cambridge University Press: New York, NY, USA, 2009; pp. 112–115.

[7] Diagana, T.; Maïga, H. Some new identities and congruences for Fubini numbers. *J. Number Theory* **2017**, *173*, 547–569.

[8] Kilar, N.; Simsek, Y. A new family of Fubini type numbers and polynomials associated with Apostol-Bernoulli numbers and polynomials. *J. Korean Math. Soc.* **2017**, *54*, 1605–1621.

[9] Gun, D.; Simsek, Y. Combinatorial sums involving Stirling, Fubini, Bernoulli numbers and approximate values of Catalan numbers. *Advanced Studies in Contemporary Mathematics* **2020**, *4*, 503–513.

[10] Sharma, S.K.; Khan, W.A.; Ryoo, C.S. A parametric kind of the degenerate Fubini numbers and polynomials. *Mathematics* **2020**, *8*, 405.

[11] Kim, T.; Kim, D.S.; Kim, H.K.; Lee, H. Some properties on degenerate Fubini polynomials. *Appl. Math. Sci. Eng.* **2022**, *30*, 235–248.

[12] Jin, S.; Dagli, M.C.; Qi, F. Degenerate Fubini-type polynomials and numbers, degenerate Apostol–Bernoulli polynomials and numbers, and degenerate Apostol–Euler polynomials and numbers. *Axioms* **2022**, *11*, 477.

[13] Kreher, D.L.; Stinson, D.R. *Combinatorial Algorithms: Generation, Enumeration, and Search*; CRC Press: Boca Raton, FL, USA, 1999.

[14] Knuth, D.E. *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1*; Addison-Wesley Professional: Boston, MA, USA, 2011.

[15] Ruskey, F. *Combinatorial Generation.* Available online: https://page.math.tu-berlin.de/ felsner/SemWS17-18/Ruskey-Comb-Gen.pdf (accessed on 1 May 2023).

[16] Barcucci, E.; Del Lungo, A.; Pergola, E.; Pinzani, R. ECO: A methodology for the enumeration of combinatorial objects. *J. Differ. Equ. Appl.* **1999**, *5*, 435–490.

[17] Flajolet, P.; Zimmerman, P.; Cutsem, B. A calculus for the random generation of combinatorial structures. *Theoret. Comput. Sci.* **1994**, *132*, 1–35.

[18] Shablya, Y.; Kruchinin, D.; Kruchinin, V. Method for developing combinatorial generation algorithms based on AND/OR trees and its application. *Mathematics* **2020**, *8*, Article 962.

[19] Shablya, Y. Combinatorial generation algorithms for some lattice paths using the method based on AND/OR trees. *Algorithms* **2023**, *16*, Article 266.

[20] Shablya, Y.; Polyuga, V. Development of combinatorial generation algorithms for discrete structures associated with the Fubini numbers. In Proceedings of the 13th Symposium on Generating Functions of Special Numbers and Polynomials and their Applications (GFSNP), Antalya, Turkey, 11–13 March 2023, 293–296.

[21] Shablya, Y.; Merinov, A.; Kruchinin, D. Combinatorial generation algorithms for directed lattice paths. *Mathematics* **2024**, *12*, Article 1207.

[22] Velleman, D.J.; Call, G.S. Permutations and combination locks. *Math. Mag.* **1995**, *68*, 243–253.

Laboratory of Algorithms and Technologies for Discrete Structures Research, Tomsk State University of Control Systems and Radioelectronics, 634050 Tomsk, Russia

Email address: `shablya-yv@mail.ru`