



Combinatorial Generation Algorithms for Directed Lattice Paths

Yuriy Shablya *, Arsen Merinov and Dmitry Kruchinin 

Laboratory of Algorithms and Technologies for Discrete Structures Research, Tomsk State University of Control Systems and Radioelectronics, 634050 Tomsk, Russia; merinovarsen@mail.ru (A.M.); kruchinindm@gmail.com (D.K.)

* Correspondence: syv@fb.tusur.ru

Abstract: Graphs are a powerful tool for solving various mathematical problems. One such task is the representation of discrete structures. Combinatorial generation methods make it possible to obtain algorithms that can create discrete structures with specified properties. This article is devoted to issues related to the construction of such combinatorial generation algorithms for a wide class of directed lattice paths. The main method used is based on the representation of a given combinatorial set in the form of an AND/OR tree structure. To apply this method, it is necessary to have an expression for the cardinality function of a combinatorial set that satisfies certain requirements. As the main result, we have found recurrence relations for enumerating simple directed lattice paths that satisfy the requirements of the applied method and have constructed the corresponding AND/OR tree structure. Applying the constructed AND/OR tree structure, we have developed new algorithms for ranking and unranking simple directed lattice paths. Additionally, the obtained results were generalized to the case of directed lattice paths.

Keywords: directed lattice path; AND/OR tree; bijection; recurrence; combinatorial generation

MSC: 68R10; 05C05



Citation: Shablya, Y.; Merinov, A.; Kruchinin, D. Combinatorial Generation Algorithms for Directed Lattice Paths. *Mathematics* **2024**, *12*, 1207. <https://doi.org/10.3390/math12081207>

Academic Editor: Janez Žerovnik

Received: 6 March 2024

Revised: 12 April 2024

Accepted: 15 April 2024

Published: 17 April 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Lattice paths represent a fundamental construct in discrete mathematics and have found applications in various fields. In simple terms, a lattice path can be defined as a sequence of moves on an integer lattice, usually constrained to certain steps, that form a path from an origin to a destination point. The elegance of lattice paths lies in their simplicity in terms of their representation. At the same time, it is possible to map a set of specific lattice paths into another set of more complex combinatorial objects. In this research, we focus on the problem of enumerating and generating lattice paths.

A lattice path P is a sequence $P = (P_0, P_1, \dots, P_l)$ of points P_i in the d -dimensional integer lattice (i.e., $P_i = (p_{i1}, p_{i2}, \dots, p_{id})$, where $p_{ij} \in \mathbb{Z}$ and $P_i \in \mathbb{Z}^d$). For a given lattice path, P_0 is the starting point and P_l is the end point. In this article, we consider lattice paths in a plane, i.e., $d = 2$. Moreover, it is required to specify a set $S = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_k\}$ of possible steps in the lattice path, where each step \mathbf{s}_i is a vector in the d -dimensional integer lattice, i.e., $\mathbf{s}_i \in \mathbb{Z}^d$. For the two-dimensional case, we have $\mathbf{s}_i = (a_i, b_i)$, where $a_i, b_i \in \mathbb{Z}$. Then, a lattice path $P = (P_0, P_1, \dots, P_l)$ can be represented as a sequence of steps performed, i.e., $P = (\overrightarrow{P_0P_1}, \dots, \overrightarrow{P_{l-1}P_l}) = (\mathbf{ps}_1, \mathbf{ps}_2, \dots, \mathbf{ps}_l)$, where $\overrightarrow{P_{i-1}P_i} = \mathbf{ps}_i \in S$. In this article, we will use only the representation of lattice paths in the form of sequences of the performed steps.

One of the problems considered in this article is the enumeration of lattice paths. A brief historical review of research related to lattice paths and their enumeration was presented in [1]. A detailed description of the main results in the field of lattice path enumeration and the methods for obtaining these results can be found in [2–4]. The most

famous and studied class of lattice paths are the so-called Dyck paths due to their connection with Catalan numbers. A large amount of information on explicit formulas for enumerating Dyck paths, including an analysis of their various parameters/statistics and their corresponding generating functions, can be found in [5–7]. In addition, there are different generalizations of Dyck paths that add new properties. For example, one of such generalizations is Dyck paths with catastrophes, motivated by a natural model in queuing theory. The problem of enumerating such lattice paths in terms of finding closed forms of corresponding generating functions was solved in [8]. The same property regarding catastrophes was considered for skew Dyck paths in [9]. Another generalization of Dyck paths related to catastrophes was considered in [10], where generating functions for enumerating such lattice paths and asymptotic approximations for their coefficients were obtained. The enumeration of labeled Dyck paths with ascents on return steps can be found in [11]. Moreover, there are a large number of other special cases of lattice paths, such as Delannoy paths [12], Schroder paths [13], Motzkin paths [14], Riordan paths [15], Lukasiewicz paths [16], etc.

This study examines a wide class of lattice paths called directed lattice paths [17]. A directed lattice path is a lattice path in the plane, where each possible step $s_i = (a_i, b_i)$ has $a_i > 0$. A simple directed lattice path is a directed lattice path where each possible step $s_i = (a_i, b_i)$ has $a_i = 1$. If a simple directed lattice path begins at the origin $P_0 = (0, 0)$ and consists of n steps of the form $s_i = (1, b_i)$, then it ends at $P_n = (n, m)$, where $m \in \mathbb{Z}$. Using the kernel method, generating functions for enumerating directed lattice paths were obtained in [17]. Additionally, a summary of results related to directed lattice paths can be found in [18]. The problem of obtaining explicit formulas for enumerating simple directed lattice paths with the possible steps of the form $(1, b_i)$, where $-h \leq b_i \leq +h$, was considered in [19]. Using a new vectorial kernel method [20], these results in enumeration and asymptotics of directed lattice paths were extended to the case of pattern-avoiding directed lattice paths. In addition, there are studies in which non-directed lattice paths are transformed into directed ones (for example, see [21]). This makes it possible to apply appropriate known methods to them.

Another problem considered in the article is the generation of lattice paths. The development of methods for generating various discrete structures is studied in a branch of science called combinatorial generation [22]. The generation problem was also considered for different classes of lattice paths. Since Dyck paths are the most famous and studied class of lattice paths, a variety of combinatorial generation algorithms have been developed for such lattice paths. There are also studies on the development of combinatorial generation algorithms for some generalizations of Dyck paths. For example, ranking and unranking algorithms for a generalized Dyck language that use bijection into Dyck paths were presented in [23]. Additional results related to the development of lexicographic generation algorithms for this generalized Dyck language were obtained in [24]. Another generalized Dyck language and a random generation scheme for it were considered in [25]. There is also an algorithm for random generation of Dyck paths with catastrophes [8]. Dyck paths are used in the development of generation algorithms not only for Dyck languages, but also for other discrete structures. For example, an efficient algorithm for the exhaustive generation of n -node binary trees using Dyck paths is presented in [26]. There are also examples of developing combinatorial generation algorithms for other lattice paths, such as Motzkin and Schroder positive paths [27] and north–east lattice paths with turns [28].

Thus, existing combinatorial generation algorithms mainly solve only the narrow generation problem for a specific combinatorial set. The lack of combinatorial generation algorithms for a wide class of directed lattice paths confirms the relevance of this research. Hence, the purpose of this article is to develop new combinatorial generation algorithms for ranking and unranking directed lattice paths.

The organization of this paper is as follows. In Section 2, we briefly describe the method used to develop new combinatorial generation algorithms for directed lattice paths and specify the main restrictions of its application. Then, in Section 3, we provide a detailed de-

scription of the main stages in the development of combinatorial generation algorithms for directed lattice paths. First, we consider simple directed lattice paths, and then generalize the obtained results to directed lattice paths. A discussion of the obtained results can be found in Section 4.

2. Materials and Methods

There are a large number of combinatorial generation algorithms for different classes of discrete structures. Some of them can be found in [22,29]. There are also several general approaches for developing combinatorial generation algorithms, such as the backtracking method [30], the ECO method [31], the Flajolet method [32] and the method based on permutation languages [33]. However, they mostly aim only at exhaustive generation or consider structures defined by a single parameter (object size). In this research, we study a method for developing combinatorial generation algorithms, which is based on the representation of combinatorial sets in the form of an AND/OR tree structure [34]. An AND/OR tree is a tree structure that contains nodes of two types: OR nodes and AND nodes. This tree structure has a set of its variants, where a variant of an AND/OR tree is a tree structure in which for each OR node only one child node is saved and the remaining child nodes are deleted. Then, the number of variants of an AND/OR tree structure is equal to the number of objects in the corresponding combinatorial set.

The main restriction on the application of this method is that the cardinality function of a combinatorial set can only consist of the following operations and their combinations:

- The use of positive integers, as well as expressions whose calculation result is a positive integer. In an AND/OR tree structure, a positive integer from the cardinality function is represented in the form of an OR node that has k child nodes, where k equals the value of this positive integer and all these child nodes are leaf nodes;
- The use of zero values. In an AND/OR tree structure, a zero value from the cardinality function is represented in the form of an empty node that must be removed from the AND/OR tree along with the subtree containing this node until the closest son of any OR node;
- The use of addition operations. In an AND/OR tree structure, an addition operation from the cardinality function is represented in the form of an OR node that has k child nodes, where k equals the number of summands in this addition and all these child nodes have their own subtree structure defined by the summands' content;
- The use of multiplication operations. In an AND/OR tree structure, a multiplication operation from the cardinality function is represented in the form of an AND node that has k child nodes, where k equals the number of factors in this multiplication and all these child nodes have their own subtree structure defined by the factors' content;
- The use of recursion operations. In an AND/OR tree structure, a recursion operation from the cardinality function is represented in the form of a node that has a subtree structure defined in the same way as the structure of the AND/OR tree;
- The use of cardinality functions of other combinatorial sets that have corresponding AND/OR tree structures. In an AND/OR tree structure, such a cardinality function is represented in the form of a node that has a subtree structure defined by the structure of the corresponding AND/OR tree;
- The use of cardinality functions of other combinatorial sets that have ranking and unranking algorithms. In an AND/OR tree structure, such a cardinality function is represented in the form of an OR node that has k child nodes, where k equals the number of objects in the corresponding combinatorial set and all these child nodes are leaf nodes;
- The use of conditional statements. In an AND/OR tree structure, such conditional statements are represented in the form of a set of AND/OR trees structures, where each AND/OR tree corresponds to one case of the conditional statements.

In contrast to the original method, the presented rules expand the possibilities of constructing AND/OR tree structures for combinatorial sets.

In previous research [35], the method based on AND/OR trees was applied to develop combinatorial generation algorithms for some special cases of lattice paths (north–east lattice paths, Dyck paths, Delannoy paths, Schroder paths, and Motzkin paths). This article continues the study presented in [36], where the possibility of constructing AND/OR tree structures for directed lattice paths was shown.

3. Results

This section provides a detailed description of the main stages in the development of combinatorial generation algorithms for directed lattice paths. First, we consider simple directed lattice paths, and then generalize the obtained results to directed lattice paths.

3.1. Simple Directed Lattice Paths

Theorem 1. *The number of all simple directed lattice paths that begin at $(0,0)$, end at (n,m) and consist of steps in the set $\{(1,b_1), (1,b_2), \dots, (1,b_k)\}$ can be calculated using the following recurrence:*

$$W_n^m = \sum_{i=1}^k W_{n-1}^{m-b_i}, \tag{1}$$

where $W_0^0 = 1$ and $W_0^m = 0$ for $m \neq 0$.

Proof. Let us consider the end point (n,m) of a directed lattice path. Based on the given set of possible steps $S = \{s_1, s_2, \dots, s_k\} = \{(1,b_1), (1,b_2), \dots, (1,b_k)\}$, the end point (n,m) can be reached from the following k points:

1. From $(n-1, m-b_1)$ to (n,m) by performing the step $s_1 = (1,b_1)$;
2. From $(n-1, m-b_2)$ to (n,m) by performing the step $s_2 = (1,b_2)$;
- ⋮
- k . From $(n-1, m-b_k)$ to (n,m) by performing the step $s_k = (1,b_k)$.

Figure 1 demonstrates a set of possible steps $S = \{s_1, s_2, \dots, s_k\}$ and all options for reaching the point (n,m) using one step from S .

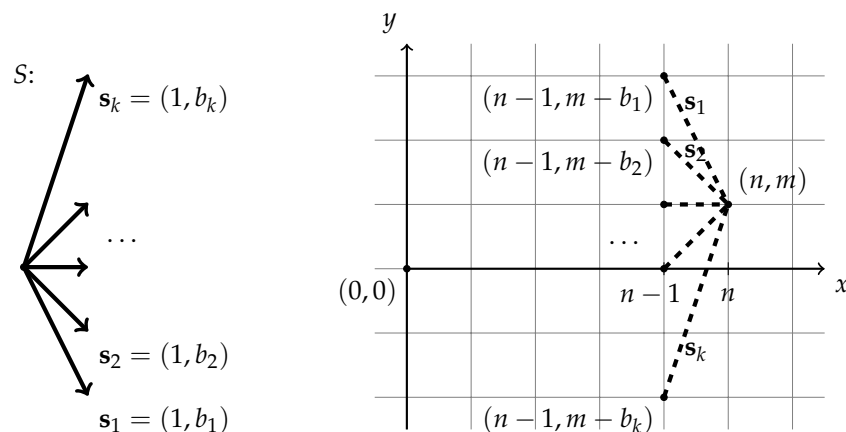


Figure 1. All options for reaching the point (n,m) using one step from $S = \{s_1, s_2, \dots, s_k\}$.

There is only one way to reach the point (n,m) from the point $(n-1, m-b_i)$: to achieve this, it is necessary to perform step $s_i = (1,b_i)$. The number of all simple directed lattice paths that begin at $(0,0)$, end at $(n-1, m-b_i)$ and consist of steps in the set $\{(1,b_1), (1,b_2), \dots, (1,b_k)\}$ is equal to $W_{n-1}^{m-b_i}$. To calculate the number of all simple directed lattice paths that begin at $(0,0)$ and end at (n,m) , it is necessary to sum the number of all simple directed lattice paths from $(0,0)$ to each point $(n-1, m-b_i)$, $i = \overline{1,k}$. Therefore, we obtain the desired recurrence (1).

Next, we define the initial conditions for the obtained recurrence (1):

1. The value of W_0^0 shows the number of all simple directed lattice paths that begin at $(0,0)$ and end at $(0,0)$. There is only one such lattice path: the empty lattice path that does not contain any performed steps. Therefore, we obtain the first initial condition $W_0^0 = 1$;
2. Since the simple directed lattice path begins at $(0,0)$ and $a_i = 1$ is true for each step $s_i = (a_i, b_i)$, it is impossible to reach the point $(0, m)$, where $m \neq 0$. Therefore, we obtain the second initial condition $W_0^m = 0$ for $m \neq 0$.

Figure 2 demonstrates a set of possible steps $S = \{s_1, s_2, \dots, s_k\}$ and all possible and impossible steps for a simple directed lattice path that begins at $(0,0)$ and ends at $(1, m)$.

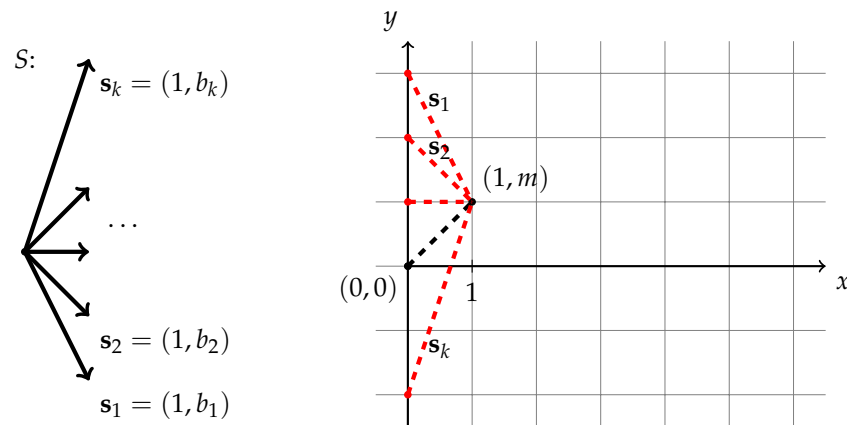


Figure 2. All possible (black line) and impossible (red line) steps for a simple directed lattice path that begins at $(0,0)$ and ends at $(1, m)$.

Hence, the obtained recurrence (1) with the defined initial conditions can be applied for calculating the number of all simple directed lattice paths that begin at $(0,0)$, end at (n, m) and consist of steps in the set $\{(1, b_1), (1, b_2), \dots, (1, b_k)\}$. □

The initial conditions for W_n^m can be slightly improved by adding the following condition that reduces the number of recursive calls when calculating W_n^m :

$$W_n^m = 0 \text{ for } m > n \cdot \max_i b_i \text{ or } m < n \cdot \min_i b_i. \tag{2}$$

In addition, we can use the following initial condition, which also reduces the number of recursive calls when calculating W_n^m :

$$W_n^m = 1 \text{ for } m = n \cdot \max_i b_i \text{ or } m = n \cdot \min_i b_i. \tag{3}$$

Figure 3 demonstrates the set of possible steps $S = \{(1, -1), (1, 0), (1, 1)\}$ and all recursive calls when applying (1) to calculate W_3^1 without (Figure 3a) and with (Figure 3b) the use of (2) and (3).

The obtained recurrence (1) with the use of the initial conditions (2) and (3) satisfies the requirements of the method for developing combinatorial generation algorithms based on AND/OR trees. Figure 4 shows the corresponding AND/OR tree structure for W_n^m . In this tree, the root, labeled W_n^m , is an OR node with k child nodes labeled i ($i = \overline{1, k}$). This structure matches the summation in (1). Each node labeled i has only one child node, labeled $W_{n-1}^{m-b_i}$, that corresponds to the summand in the summation in (1). The subtree of the node labeled $W_{n-1}^{m-b_i}$, which is shown as a node with a triangle, is constructed in a similar way for the new values of parameters n and m ; i.e., a recursive call is needed. Such recursive calls stop when some initial condition for W_n^m are satisfied:

- If we obtain a node where $W_n^m = 1$, then it is a leaf node;
- If we obtain a node where $W_n^m = 0$, then it is necessary to remove this node and its parent node labeled i .

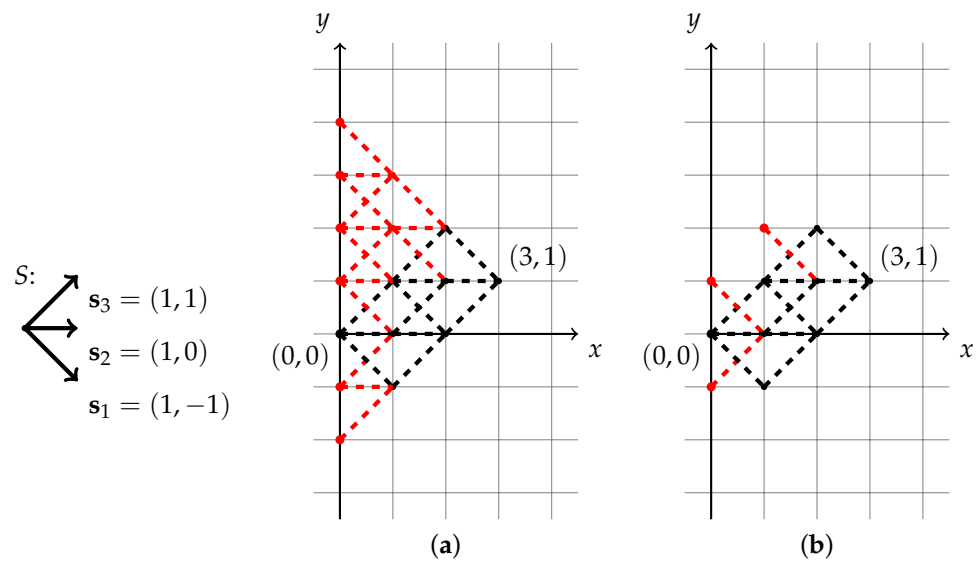


Figure 3. An example of applying (1): (a) without (2) and (3). (b) with (2) and (3).

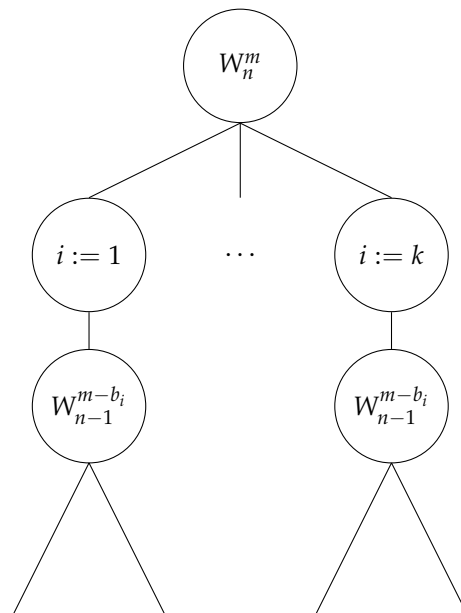


Figure 4. An AND/OR tree structure for W_n^m based on (1).

According to the applied method [34], the total number of variants of the AND/OR tree structure for W_n^m is equal to W_n^m . The constructed tree structure does not contain AND nodes, since there are no multiplications in the recurrence (1). Therefore, a variant of such an AND/OR tree is a path from the root to a leaf node. A bijection between the considered set of all simple directed lattice paths and the set of all variants of the constructed AND/OR tree structure is defined by the following rules:

- Selecting a child of the node labeled W_n^m (each child is labeled with a specific value of the parameter i) corresponds to selecting one of the possible steps s_i when reaching the point (n, m) , i.e., $\mathbf{ps}_n = s_i$. The subtree of the selected child node determines the remaining part of the simple directed lattice path from $(0, 0)$ to $(n - 1, m - b_i)$. It is also important to note that it is necessary to fix some order on the set of possible steps S , because the AND/OR tree is an ordered structure;
- A leaf node labeled W_n^m , where $m = n \cdot \max_i b_i$, means reaching the point (n, m) from $(0, 0)$ by performing n steps of the form $(1, \max_i b_i)$, i.e., $\mathbf{ps}_1 = \dots = \mathbf{ps}_n = (1, \max_i b_i)$;

- A leaf node labeled W_n^m , where $m = n \cdot \min_i b_i$, means reaching the point (n, m) from $(0, 0)$ by performing n steps of the form $(1, \min_i b_i)$, i.e., $\mathbf{ps}_1 = \dots = \mathbf{ps}_n = (1, \min_i b_i)$;
- A leaf node labeled W_0^0 means reaching the starting point $(0, 0)$; i.e., there are no additional steps required.

Figure 5 shows the corresponding AND/OR tree structure for W_3^1 with the set of possible steps $S = \{\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3\} = \{(1, -1), (1, 0), (1, 1)\}$. This structure is constructed based on (1) with the use of (2) and (3). The total number of its variants is equal to $W_3^1 = 6$, which corresponds to the following simple directed lattice paths: $(\mathbf{s}_3, \mathbf{s}_3, \mathbf{s}_1)$, $(\mathbf{s}_3, \mathbf{s}_2, \mathbf{s}_2)$, $(\mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_2)$, $(\mathbf{s}_3, \mathbf{s}_1, \mathbf{s}_3)$, $(\mathbf{s}_2, \mathbf{s}_2, \mathbf{s}_3)$, $(\mathbf{s}_1, \mathbf{s}_3, \mathbf{s}_3)$.

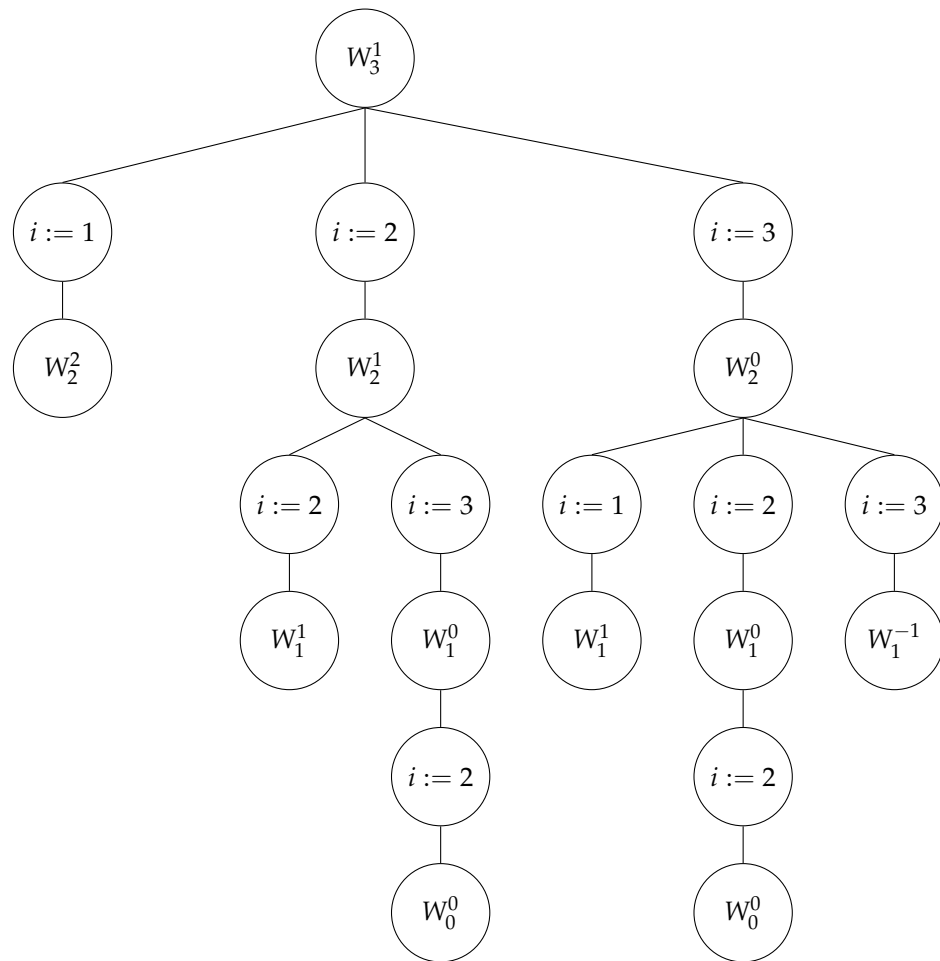


Figure 5. An AND/OR tree structure for W_3^1 with $S = \{(1, -1), (1, 0), (1, 1)\}$.

Since the constructed tree structure does not contain AND nodes and the choice between children of OR nodes is associated with the choice of feasible solutions when constructing lattice paths. This tree structure is similar to the state space tree obtained by applying the backtracking method [30]. Traversing this tree allows for exhaustive generation of the considered simple directed lattice paths. However, in contrast to the backtracking method, the applied method [34] makes it possible to obtain other classes of combinatorial generation algorithms for the set of simple directed lattice paths.

According to the applied method for developing combinatorial generation algorithms based on AND/OR trees, we obtain algorithms for ranking (Algorithm 1) and unranking (Algorithm 2) simple directed lattice paths that begin at $(0, 0)$, end at (n, m) and consist of steps in the set $S = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_k\} = \{(1, b_1), (1, b_2), \dots, (1, b_k)\}$. In these combinatorial generation algorithms, $()$ denotes an empty lattice path and $\text{concat}()$ denotes the concatenation of lattice paths; i.e., $\text{concat}((\mathbf{s}_1, \mathbf{s}_2), (\mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_5))$ returns the lattice path $(\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3, \mathbf{s}_4, \mathbf{s}_5)$.

Algorithm 1: An algorithm for ranking a simple directed lattice path that begins at $(0,0)$, ends at (n,m) and consists of steps in the ordered set $S = \{(1, b_1), (1, b_2), \dots, (1, b_k)\}$.

```

1 Rank_SDLP( $P = (\mathbf{ps}_1, \mathbf{ps}_2, \dots, \mathbf{ps}_n), n, m$ )
2 begin
3   if  $(n = 0$  and  $m = 0)$  or  $m = n \cdot \max_i b_i$  or  $m = n \cdot \min_i b_i$  then  $r := 0$ 
4   else
5      $I :=$  index of  $\mathbf{ps}_n$  in the ordered set  $S$ 
6      $sum := \sum_{i=1}^{I-1} W_{n-1}^{m-b_i}$ 
7      $r := sum +$  Rank_SDLP( $(\mathbf{ps}_1, \mathbf{ps}_2, \dots, \mathbf{ps}_{n-1}), n - 1, m - b_I$ )
8   end
9   return  $r$ 
10 end
```

Algorithm 2: An algorithm for unranking a simple directed lattice path that begins at $(0,0)$, ends at (n,m) and consists of steps in the ordered set $S = \{(1, b_1), (1, b_2), \dots, (1, b_k)\}$.

```

1 Unrank_SDLP( $r, n, m$ )
2 begin
3   if  $n = 0$  and  $m = 0$  then  $P := ()$ 
4   else if  $m = n \cdot \max_i b_i$  then  $P := (\mathbf{ps}_1, \dots, \mathbf{ps}_n) = ((1, \max_i b_i), \dots, (1, \max_i b_i))$ 
5   else if  $m = n \cdot \min_i b_i$  then  $P := (\mathbf{ps}_1, \dots, \mathbf{ps}_n) = ((1, \min_i b_i), \dots, (1, \min_i b_i))$ 
6   else
7      $sum := 0$ 
8     for  $i := 1$  to  $k$  do
9        $s := W_{n-1}^{m-b_i}$ 
10      if  $sum + s > r$  then
11         $r := r - sum$ 
12         $I := i$ 
13        break
14      end
15       $sum := sum + s$ 
16    end
17     $P :=$  concat(Unrank_SDLP( $r, n - 1, m - b_I$ ),  $(\mathbf{s}_I)$ )
18  end
19  return  $P$ 
20 end
```

Algorithms 1 and 2 have at most n recursive calls, where each recursive call requires calculations of W_n^m maximum k times. Hence, Algorithms 1 and 2 have a time complexity of $O(n \cdot k \cdot w(n, k, S))$, where $w(n, k, S)$ is a function that describes the increase in the time of calculating value W_n^m depending on the values of parameters n and m , as well as the set of possible steps S . If we use recurrence (1) for calculating W_n^m , which has time complexity $O(k^n)$, we obtain time complexity $O(nk^n)$ for ranking and unranking algorithms. The time complexity of the developed combinatorial generation algorithms can be improved by using formulas for calculating W_n^m that are more efficient in terms of computational complexity. Such formulas can be found by considering simple directed lattice paths with a specific set of possible steps.

Table 1 presents an example of applying Algorithm 1 for ranking and Algorithm 2 for unranking simple directed lattice paths that begin at $(0, 0)$, end at $(3, 1)$ and consist of steps in the ordered set $S = \{(1, -1), (1, 0), (1, 1)\}$.

Table 1. Ranking and unranking simple directed lattice paths that begin at $(0, 0)$, end at $(3, 1)$ and consist of steps in the ordered set $S = \{s_1, s_2, s_3\} = \{(1, -1), (1, 0), (1, 1)\}$.

Lattice Path P	$\text{Rank}(P, 3, 1)$	Rank r	$\text{Unrank}(r, 3, 1)$	Lattice Path P
(s_3, s_3, s_1)		0		(s_3, s_3, s_1)
(s_3, s_2, s_2)		1		(s_3, s_2, s_2)
(s_2, s_3, s_2)		2		(s_2, s_3, s_2)
(s_3, s_1, s_3)		3		(s_3, s_1, s_3)
(s_2, s_2, s_3)		4		(s_2, s_2, s_3)
(s_1, s_3, s_3)		5		(s_1, s_3, s_3)

Thus, using the developed algorithms, it is possible to rank any simple directed lattice paths, as well as generate a specific lattice path according to a given rank. Moreover, some lattice paths can be transformed to the form of simple directed lattice paths. For example, we can consider north–East lattice paths [37]. A north–east lattice path is a lattice path that begins at $(0, 0)$, ends at (n, m) and consists of steps in the set $S = \{s_1, s_2\} = \{(0, 1), (1, 0)\}$. The step $s_1 = (0, 1)$ is called the north step and the step $s_2 = (1, 0)$ is called the east step. The number of all north–east lattice paths can be calculated using the following explicit formula:

$$L_n^m = \binom{n+m}{m}. \tag{4}$$

North–east lattice paths are not directed lattice paths. If we change the basis and consider another coordinate system (red x^* -axis and y^* -axis in Figure 6), then the north–east lattice paths are transformed into simple directed lattice paths. The obtained simple directed lattice paths begin at $(0, 0)$, end at (n^*, m^*) , and consist of steps in the set $S^* = \{s_1^*, s_2^*\}$ in the new coordinate system. The connection between the new and old coordinate systems is stated in the following formulas:

$$n^* = m + n, \quad m^* = m - n. \tag{5}$$

As a result, we obtain the set of possible steps $S^* = \{s_1^*, s_2^*\} = \{(1, 1), (1, -1)\}$ that satisfy the requirements of simple directed lattice paths.

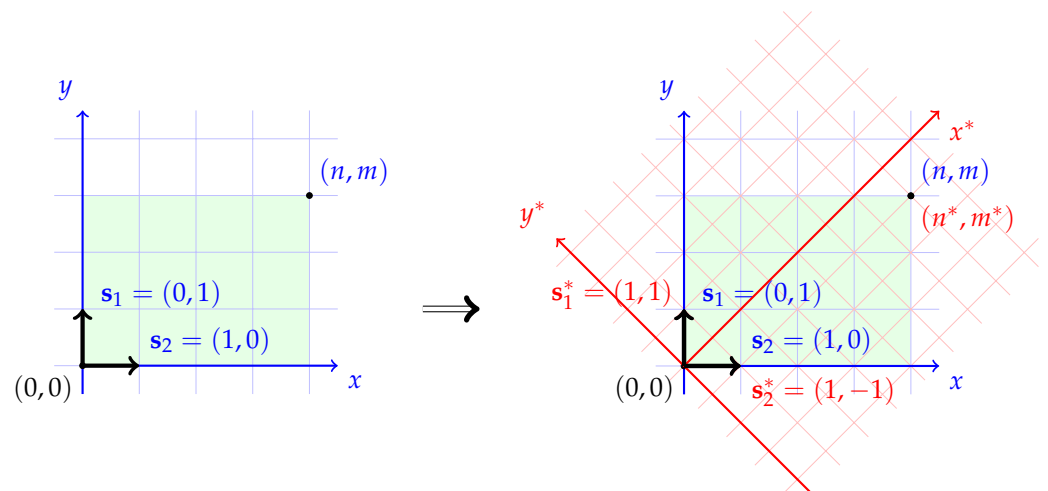


Figure 6. Transformation of north–east lattice paths into simple directed lattice paths.

The number of all simple directed lattice paths that begin at $(0, 0)$, end at (n^*, m^*) and consist of steps in the set $S^* = \{s_1^*, s_2^*\} = \{(1, 1), (1, -1)\}$ is equal to $W_{n^*}^{m^*}$. Hence,

$$L_n^m = W_{n^*}^{m^*} = W_{m+n}^{m-n}. \tag{6}$$

Applying Algorithms 1 and 2 for the considered simple directed lattice paths, we obtain algorithms for ranking and unranking north-east lattice paths. For example, to rank a north-east lattice path $P = (\mathbf{ps}_1, \dots, \mathbf{ps}_n)$, $\mathbf{ps}_i \in S = \{(0, 1), (1, 0)\}$ that begins at $(0, 0)$ and ends at (n, m) , it is necessary to change the basis by using (5), obtain the corresponding simple directed lattice path $P^* = (\mathbf{ps}_1^*, \dots, \mathbf{ps}_n^*)$, $\mathbf{ps}_i^* \in S^* = \{(1, 1), (1, -1)\}$, and execute

$$\text{Rank_SDLP_NE}(P^* = (\mathbf{ps}_1^*, \mathbf{ps}_2^*, \dots, \mathbf{ps}_n^*), m + n, m - n).$$

According to Theorem 1, for the case $S = \{(1, 1), (1, -1)\}$, we obtain

$$W_n^m = \begin{cases} 0, & m > n \text{ or } m < -n; \\ 1, & m = n \text{ or } m = -n; \\ W_{n-1}^{m-1} + W_{n-1}^{m+1}, & \text{otherwise.} \end{cases} \tag{7}$$

If we apply (7) when executing Algorithm 3 or Algorithm 4, we obtain a time complexity of $O(n \cdot 2^n)$. A significant improvement will be obtained if we use explicit formula (4) in (6):

$$W_n^m = \begin{cases} 0, & (n + m) \text{ odd;} \\ L_{\frac{n+m}{2}}^{\frac{n-m}{2}}, & \text{otherwise} \end{cases} = \begin{cases} 0, & (n + m) \text{ odd;} \\ \binom{n}{\frac{n+m}{2}}, & \text{otherwise.} \end{cases} \tag{8}$$

Algorithm 3: An algorithm for ranking a simple directed lattice path that begins at $(0, 0)$, ends at (n, m) and consists of steps in the ordered set $S = \{(1, 1), (1, -1)\}$.

```

1 Rank_SDLP_NE( $P = (\mathbf{ps}_1, \mathbf{ps}_2, \dots, \mathbf{ps}_n)$ ,  $n, m$ )
2 begin
3   if  $m = n$  or  $m = -n$  then  $r := 0$ 
4   else
5     if  $\mathbf{ps}_n = (1, 1)$  then  $r := \text{Rank\_SDLP\_NE}((\mathbf{ps}_1, \mathbf{ps}_2, \dots, \mathbf{ps}_{n-1}), n - 1, m - 1)$ 
6     else  $r := W_{n-1}^{m-1} + \text{Rank\_SDLP\_NE}((\mathbf{ps}_1, \mathbf{ps}_2, \dots, \mathbf{ps}_{n-1}), n - 1, m + 1)$ 
7   end
8   return  $r$ 
9 end
```

Algorithm 4: An algorithm for unranking a simple directed lattice path that begins at $(0, 0)$, ends at (n, m) and consists of steps in the ordered set $S = \{(1, 1), (1, -1)\}$.

```

1 Unrank_SDLP_NE( $r, n, m$ )
2 begin
3   if  $n = 0$  and  $m = 0$  then  $P := ()$ 
4   else if  $m = n$  then  $P := (\mathbf{ps}_1, \dots, \mathbf{ps}_n) = ((1, 1), \dots, (1, 1))$ 
5   else if  $m = -n$  then  $P := (\mathbf{ps}_1, \dots, \mathbf{ps}_n) = ((1, -1), \dots, (1, -1))$ 
6   else
7     if  $r < W_{n-1}^{m-1}$  then  $P := \text{concat}(\text{Unrank\_SDLP\_NE}(r, n - 1, m - 1), ((1, 1)))$ 
8     else  $P := \text{concat}(\text{Unrank\_SDLP\_NE}(r - W_{n-1}^{m-1}, n - 1, m + 1), ((1, -1)))$ 
9   end
10  return  $P$ 
11 end
```

Then, we obtain a polynomial time complexity $O(n^2)$ for Algorithms 3 and 4. The results of computational experiments using software implementations for Algorithms 3 and 4 confirm the theoretical time complexity.

3.2. Simple Directed Lattice Paths with Restrictions

Various restrictions can be applied to the generated lattice paths. There may be restrictions of the following type: fixing the end point of the lattice path, prohibiting going beyond certain boundaries, calculating some statistics of the lattice path, etc. For example, the following four types of directed lattice paths are distinguished in [17]:

1. *Walk*: a directed lattice path that ends at (n, m) , where m is not specified;
2. *Bridge*: a directed lattice path that ends at $(n, 0)$;
3. *Meander*: a directed lattice path that ends at (n, m) , where m is not specified and never falls below the x -axis;
4. *Excursion*: a directed lattice path that ends at $(n, 0)$ and never falls below the x -axis.

Next, we consider these types of lattice paths for simple directed lattice paths. Simple walks and bridges are the special cases of the simple directed lattice paths from Theorem 1. Therefore, we can calculate the number of all simple walks or bridges through W_n^m .

Corollary 1. *The number of all simple walks that begin at $(0, 0)$, end at (n, m) , where m is not specified, and consist of steps in the set $\{(1, b_1), (1, b_2), \dots, (1, b_k)\}$ can be calculated using the following formula:*

$$W_n = \sum_{m=n \cdot \min b_i}^{n \cdot \max b_i} W_n^m. \tag{9}$$

Proof. Simple walks differ from the simple directed lattice paths from Theorem 1 in that m is not specified. Therefore, we need to consider all possible values of m and sum up the number of simple directed lattice paths for each of them. The minimum possible value of m will be obtained if all n steps of the form $(1, b_{\min})$ are performed in a simple directed lattice path, where $b_{\min} = \min_i b_i$. The maximum possible value of m will be obtained if all n steps of the form $(1, b_{\max})$ are performed in a simple directed lattice path, where $b_{\max} = \max_i b_i$. Combining all the conditions, we obtain the desired formula (9). \square

Corollary 2. *The number of all simple bridges that begin at $(0, 0)$, end at $(n, 0)$ and consist of steps in the set $\{(1, b_1), (1, b_2), \dots, (1, b_k)\}$ can be calculated using the following formula:*

$$B_n = W_n^0.$$

To calculate the number of simple meanders and excursions, it is necessary to prevent them from falling below the x -axis. Therefore, we add the appropriate initial condition to recurrence (1) and obtain the following theorem:

Theorem 2. *The number of all simple directed lattice paths that begin at $(0, 0)$, end at (n, m) , consist of steps in the set $\{(1, b_1), (1, b_2), \dots, (1, b_k)\}$ and never fall below the x -axis can be calculated using the following recurrence:*

$$M_n^m = \sum_{i=1}^k M_{n-1}^{m-b_i},$$

where $M_0^0 = 1$, $M_0^m = 0$ for $m \neq 0$ and $M_n^m = 0$ for $m < 0$.

Proof. In contrast to Theorem 1, for the considered simple directed lattice paths, it is impossible to reach the point $(0, m)$, where $m < 0$. Therefore, we obtain an initial condition that $M_n^m = 0$ for $m < 0$. \square

Since the considered simple directed lattice paths never fall below the x -axis, only the following additional initial conditions can be added to M_n^m :

$$M_n^m = 0 \text{ for } m > n \cdot \max_i b_i,$$

$$M_n^m = 1 \text{ for } m = n \cdot \max_i b_i.$$

According to the applied method for developing combinatorial generation algorithms based on AND/OR trees, we obtain algorithms for ranking (Algorithm 5) and unranking (Algorithm 6) simple directed lattice paths that begin at $(0, 0)$, end at (n, m) , consist of steps in the set $S = \{s_1, s_2, \dots, s_k\} = \{(1, b_1), (1, b_2), \dots, (1, b_k)\}$ and never fall below the x -axis.

Algorithm 5: An algorithm for ranking a simple directed lattice path that begins at $(0, 0)$, ends at (n, m) , consists of steps in the ordered set $S = \{(1, b_1), (1, b_2), \dots, (1, b_k)\}$ and never fall below the x -axis.

```

1 Rank_SDLPx( $P = (\mathbf{ps}_1, \mathbf{ps}_2, \dots, \mathbf{ps}_n), n, m$ )
2 begin
3   if ( $n = 0$  and  $m = 0$ ) or  $m = n \cdot \max_i b_i$  then  $r := 0$ 
4   else
5      $I :=$  index of  $\mathbf{ps}_n$  in the ordered set  $S$ 
6      $sum := \sum_{i=1}^{I-1} M_{n-1}^{m-b_i}$ 
7      $r := sum + \text{Rank\_SDLPx}((\mathbf{ps}_1, \mathbf{ps}_2, \dots, \mathbf{ps}_{n-1}), n-1, m-b_I)$ 
8   end
9   return  $r$ 
10 end
```

Algorithm 6: An algorithm for unranking a simple directed lattice path that begins at $(0, 0)$, ends at (n, m) , consists of steps in the ordered set $S = \{(1, b_1), (1, b_2), \dots, (1, b_k)\}$ and never fall below the x -axis.

```

1 Unrank_SDLPx( $r, n, m$ )
2 begin
3   if  $n = 0$  and  $m = 0$  then  $P := ()$ 
4   else if  $m = n \cdot \max_i b_i$  then  $P := (\mathbf{ps}_1, \dots, \mathbf{ps}_n) = ((1, \max_i b_i), \dots, (1, \max_i b_i))$ 
5   else
6      $sum := 0$ 
7     for  $i := 1$  to  $k$  do
8        $s := M_{n-1}^{m-b_i}$ 
9       if  $sum + s > r$  then
10         $r := r - sum$ 
11         $I := i$ 
12        break
13      end
14       $sum := sum + s$ 
15    end
16     $P := \text{concat}(\text{Unrank\_SDLPx}(r, n-1, m-b_I), (s_I))$ 
17  end
18  return  $P$ 
19 end
```

Simple meanders and excursions are the special cases of the simple directed lattice paths from Theorem 2. Therefore, we can calculate the number of all simple meanders and excursions through M_n^m .

Corollary 3. *The number of all simple meanders that begin at $(0,0)$, end at (n,m) , where m is not specified, consist of steps in the set $\{(1,b_1), (1,b_2), \dots, (1,b_k)\}$ and never fall below the x -axis can be calculated using the following formula:*

$$M_n = \sum_{m=0}^{n \cdot \max b_i} M_n^m.$$

Proof. The proof is similar to the proof of Corollary 1. \square

Corollary 4. *The number of all simple excursions that begin at $(0,0)$, end at $(n,0)$, consist of steps in the set $\{(1,b_1), (1,b_2), \dots, (1,b_k)\}$ and never fall below the x -axis can be calculated using the following formula:*

$$E_n = M_n^0.$$

As an example, we consider the applications of Algorithms 5 and 6 for Dyck paths [7]. A Dyck n -path is a lattice path that begins at $(0,0)$, ends at (n,n) , consists of steps in the set $S = \{s_1, s_2\} = \{(0,1), (1,0)\}$ and never rises above the diagonal $y = x$. The number of all Dyck n -paths is equal to the Catalan numbers C_n (the sequence A000108 in the OEIS [38]).

Dyck paths are not directed lattice paths. If we change the basis and consider another coordinate system (red x^* -axis and y^* -axis in Figure 7), then the Dyck paths are transformed into simple directed lattice paths. The obtained simple directed lattice paths begin at $(0,0)$, end at $(2n,0)$, consist of steps in the set $S^* = \{s_1^*, s_2^*\} = \{(1,-1), (1,1)\}$ and never fall below the x^* -axis in the new coordinate system.

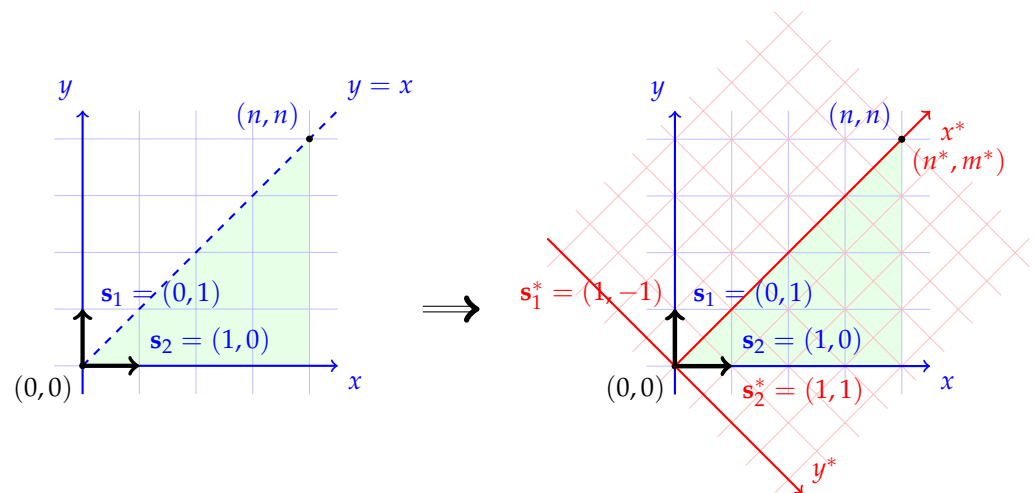


Figure 7. Transformation of Dyck paths into simple directed lattice paths.

The number of all simple directed lattice paths that begin at $(0,0)$, end at $(2n,0)$, consist of steps in the set $S^* = \{(1,-1), (1,1)\}$ and never fall below the x^* -axis is equal to M_{2n}^0 . Hence, $C_n = M_{2n}^0 = E_{2n}$. Applying Algorithms 5 and 6 for the considered simple directed lattice paths, we obtain algorithms for ranking and unranking Dyck paths. For example, to rank a Dyck path $P = (\mathbf{ps}_1, \dots, \mathbf{ps}_n)$, $\mathbf{ps}_i \in S = \{(0,1), (1,0)\}$, that begins at $(0,0)$ and ends at (n,n) , it is necessary to change the basis, obtain the corresponding simple directed lattice path $P^* = (\mathbf{ps}_1^*, \dots, \mathbf{ps}_n^*)$, $\mathbf{ps}_i^* \in S^* = \{(1,-1), (1,1)\}$, and execute

$$\text{Rank_SDLPx}(P^* = (\mathbf{ps}_1^*, \mathbf{ps}_2^*, \dots, \mathbf{ps}_n^*), 2n, 0).$$

3.3. Directed Lattice Paths

All obtained results on recurrent formulas and combinatorial generation algorithms for simple directed lattice paths can be generalized to the case of directed lattice paths. Similar results in obtaining recurrences for directed lattice paths were obtained in [39] (Example 8).

Theorem 3. *The number of all directed lattice paths that begin at $(0,0)$, end at (n,m) and consist of steps in the set $\{(a_1, b_1), (a_2, b_2), \dots, (a_k, b_k)\}$ can be calculated using the following recurrence:*

$$W_n^m = \begin{cases} 0, & n = 0 \text{ and } m \neq 0 \text{ or } n < 0; \\ 1, & n = m = 0; \\ \sum_{i=1}^k W_{n-a_i}^{m-b_i}, & \text{otherwise.} \end{cases} \tag{10}$$

Proof. The proof is similar to the proof of Theorem 1. \square

Theorem 4. *The number of all directed lattice paths that begin at $(0,0)$, end at (n,m) , consist of steps in the set $\{(a_1, b_1), (a_2, b_2), \dots, (a_k, b_k)\}$ and never fall below the x -axis can be calculated using the following recurrence:*

$$M_n^m = \begin{cases} 0, & n = 0 \text{ and } m \neq 0 \text{ or } n < 0 \text{ or } m < 0; \\ 1, & n = m = 0; \\ \sum_{i=1}^k M_{n-a_i}^{m-b_i}, & \text{otherwise.} \end{cases} \tag{11}$$

Proof. The proof is similar to the proof of Theorem 2. \square

The obtained recurrences (10) and (11) also satisfy the requirements of the method for developing combinatorial generation algorithms based on AND/OR trees. Thus, this makes it possible to obtain combinatorial generation algorithms for directed lattice paths. Algorithms 7 and 8 present the obtained algorithms for ranking and unranking directed lattice paths from Theorem 3. Moreover, if a lattice path can be transformed into a directed lattice path, then combinatorial generation algorithms can also be obtained for it. For example, next, we consider the transformation of several well-known lattice paths into directed lattice paths.

Algorithm 7: An algorithm for ranking a directed lattice path that begins at $(0,0)$, ends at (n,m) and consists of steps in the ordered set $S = \{(a_1, b_1), (a_2, b_2), \dots, (a_k, b_k)\}$.

```

1 Rank_DLP( $P = (\mathbf{ps}_1, \mathbf{ps}_2, \dots, \mathbf{ps}_l), n, m$ )
2 begin
3   if  $n = 0$  and  $m = 0$  then  $r := 0$ 
4   else
5      $I :=$  index of  $\mathbf{ps}_l$  in the ordered set  $S$ 
6      $sum := \sum_{i=1}^{I-1} W_{n-a_i}^{m-b_i}$ 
7      $r := sum + \text{Rank\_DLP}((\mathbf{ps}_1, \mathbf{ps}_2, \dots, \mathbf{ps}_{l-1}), n - a_I, m - b_I)$ 
8   end
9   return  $r$ 
10 end
```

Algorithm 8: An algorithm for unranking a directed lattice path that begins at $(0,0)$, ends at (n,m) and consists of steps in the ordered set $S = \{(a_1, b_1), (a_2, b_2), \dots, (a_k, b_k)\}$.

```

1 Unrank_DLP( $r, n, m$ )
2 begin
3   if  $n = 0$  and  $m = 0$  then  $P := ()$ 
4   else
5      $sum := 0$ 
6     for  $i := 1$  to  $k$  do
7        $s := W_{n-a_i}^{m-b_i}$ 
8       if  $sum + s > r$  then
9          $r := r - sum$ 
10         $I := i$ 
11        break
12      end
13       $sum := sum + s$ 
14    end
15     $P := \text{concat}(\text{Unrank\_DLP}(r, n - a_I, m - b_I), (s_I))$ 
16  end
17  return  $P$ 
18 end

```

A Delannoy path is a lattice path that begins at $(0,0)$, ends at (n,m) and consists of steps in the set $S = \{(0,1), (1,0), (1,1)\}$. The number of all Delannoy paths is equal to the Delannoy numbers D_n^m (sequence A008288 in the OEIS [38]). Delannoy paths are not directed lattice paths. If we change the basis and consider another coordinate system (red x^* -axis and y^* -axis in Figure 8), then the Delannoy paths are transformed into directed lattice paths. The obtained directed lattice paths begin at $(0,0)$, end at $(m+n, m-n)$ and consist of steps in the set $S^* = \{(1,1), (1,-1), (2,0)\}$ in the new coordinate system.

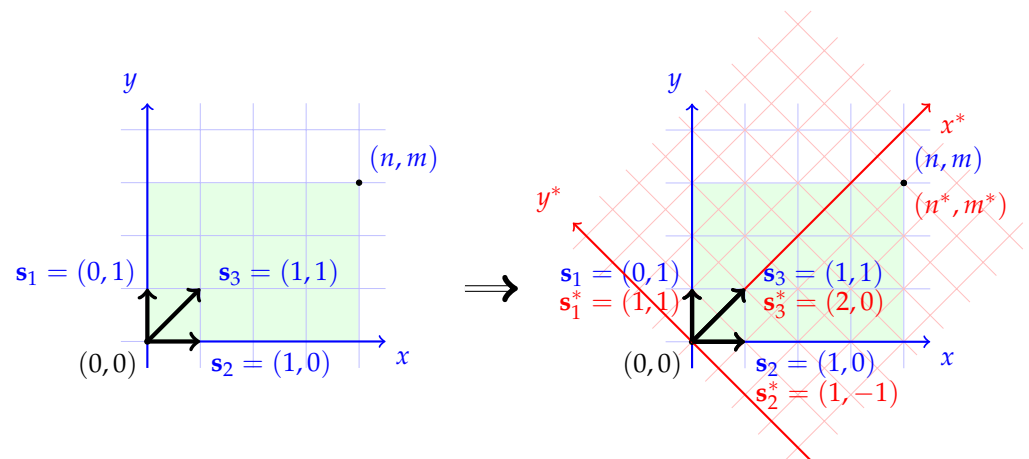


Figure 8. Transformation of Delannoy paths into directed lattice paths.

The number of all directed lattice paths that begin at $(0,0)$, end at $(m+n, m-n)$ and consist of steps in the set $S^* = \{(1,1), (1,-1), (2,0)\}$ is equal to W_{m+n}^{m-n} . Hence, $D_n^m = W_{m+n}^{m-n}$. Applying ranking and unranking algorithms for the considered directed lattice paths, we obtain appropriate algorithms for Delannoy paths.

A Schroder path is a lattice path that begins at $(0,0)$, ends at (n,n) , consists of steps in the set $S = \{(0,1), (1,0), (1,1)\}$ and never rises above the diagonal $y = x$. The number of all Schroder paths is equal to the Schroder numbers S_n (sequence A006318 in the OEIS [38]). Schroder paths are not directed lattice paths. If we change the basis and consider an-

other coordinate system (red x^* -axis and y^* -axis in Figure 9), then the Schroder paths are transformed into directed lattice paths. The obtained directed lattice paths begin at $(0,0)$, end at $(2n,0)$, consist of steps in the set $S^* = \{(1, -1), (1, 1), (2, 0)\}$ and never fall below the x^* -axis in the new coordinate system.

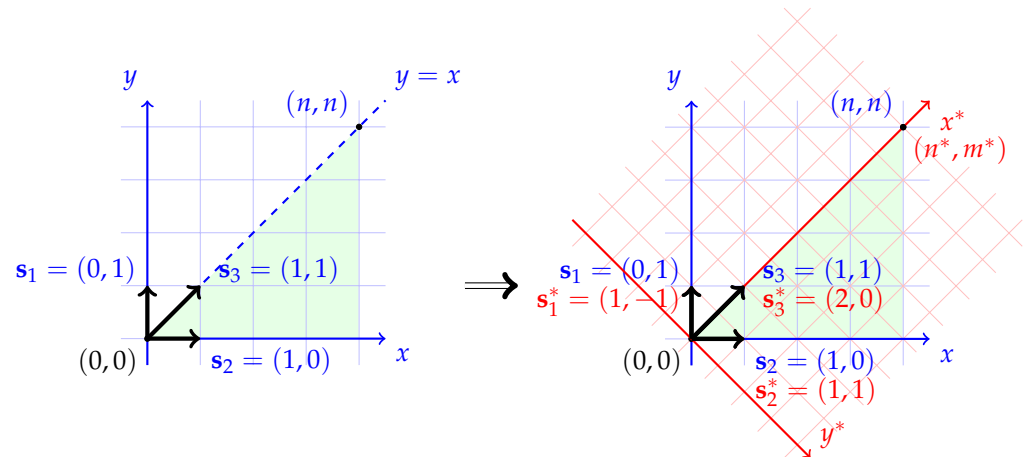


Figure 9. Transformation of Schroder paths into directed lattice paths.

The number of all directed lattice paths that begin at $(0,0)$, end at $(2n,0)$, consist of steps in the set $S^* = \{(1, -1), (1, 1), (2, 0)\}$ and never fall below the x -axis is equal to M_{2n}^0 . Hence, $S_n = M_{2n}^0$. Applying ranking and unranking algorithms for the considered directed lattice paths, we obtain appropriate algorithms for Schroder paths.

A Motzkin path is a lattice path that begins at $(0,0)$, ends at (n,n) , consists of steps in the set $S = \{(0,2), (2,0), (1,1)\}$ and never rises above the diagonal $y = x$. The number of all Motzkin paths is equal to the Motzkin numbers (sequence A001006 in the OEIS [38]). Motzkin paths are not directed lattice paths. If we change the basis and consider another coordinate system (red x^* -axis and y^* -axis in Figure 10), then the Motzkin paths are transformed into directed lattice paths. The obtained directed lattice paths begin at $(0,0)$, end at $(n,0)$, consist of steps in the set $S^* = \{(1, -1), (1, 1), (1, 0)\}$ and never fall below the x^* -axis in the new coordinate system. Applying ranking and unranking algorithms for the considered directed lattice paths, we obtain appropriate algorithms for Motzkin paths.

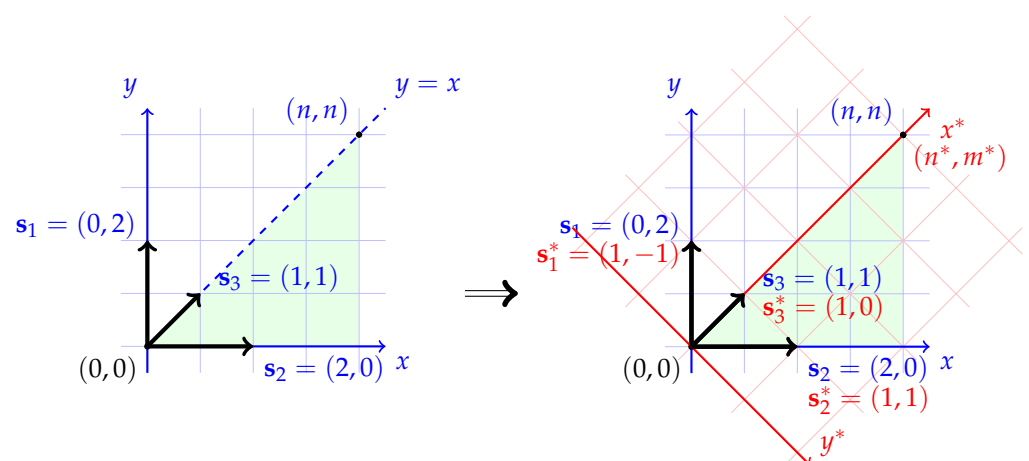


Figure 10. Transformation of Motzkin paths into directed lattice paths.

4. Discussion

In this article, we have demonstrated the possibilities of applying a method for developing combinatorial generation algorithms, which is based on the representation of combinatorial sets in the form of an AND/OR tree structure. In contrast to the original

method, the rules presented in Section 2 expand the possibilities of constructing AND/OR tree structures for combinatorial sets based on their cardinality functions.

To develop new combinatorial generation algorithms, the following broad class of combinatorial sets was chosen: directed lattice paths. The lack of such algorithms confirms the relevance of this research. For example, lattice paths are used in bioinformatics to describe molecular structures such as RNA and DNA. Lattice paths can also be used in the field of physics to describe the motion of particles or in the field of finance to predict changes in various financial indicators. In addition, there are many other discrete structures that have practical applications and can be described by lattice paths. Then, the developed combinatorial generation algorithms can be used to model such discrete structures or to test systems that work with them.

The main result of this article is that we have found recurrence relations for enumerating simple directed lattice paths that satisfy the requirements of the applied method. Recurrence (1) from Theorem 1 can be used to calculate the number of all simple directed lattice paths without restrictions. Theorem 2 and the corollaries of these theorems allow us to calculate the number of simple directed lattice paths that have restrictions, such as fixing the end point of the lattice path or prohibiting falling below the x -axis. Based on these recurrences, the corresponding AND/OR tree structures have been constructed. Applying the constructed AND/OR tree structure, we have developed new algorithms for ranking and unranking simple directed lattice paths. To make the developed algorithms efficient, it is advisable to use other formulas for calculating the values of W_n^m and M_n^m that have an explicit form with a polynomial computational complexity.

Additionally, the obtained results were generalized to the case of directed lattice paths. Thus, this makes it possible to obtain combinatorial generation algorithms for any directed lattice path. The results of computational experiments using software implementations of developed combinatorial generation algorithms have confirmed their correctness. Note that the obtained recurrences correspond to the idea of the backtracking method. Thus, this approach can be generalized to the case of any lattice path (not just directed lattice paths). At the same time, constructing an appropriate AND/OR tree structure allows us to develop ranking and unranking algorithms for such lattice paths.

Author Contributions: Methodology, Y.S.; software, Y.S.; investigation, Y.S., A.M. and D.K.; writing—original draft preparation, Y.S.; supervision, D.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Russian Science Foundation, grant number 22-71-10052.

Data Availability Statement: The source code can be made available on request.

Acknowledgments: The author would like to thank the referees for their helpful comments and suggestions.

Conflicts of Interest: The author declare no conflicts of interest.

References

- Humphreys, K. A history and a survey of lattice path enumeration. *J. Statist. Plann. Inference* **2010**, *140*, 2237–2254. [[CrossRef](#)]
- Stanley, R.P. *Enumerative Combinatorics: Volume 1*, 2nd ed.; Cambridge University Press: New York, NY, USA, 2012.
- Krattenthaler, C. Lattice path enumeration. In *Handbook of Enumerative Combinatorics*; Bona, M., Ed.; CRC Press: New York, NY, USA, 2015; pp. 589–678.
- Wallner, M. *Combinatorics of Lattice Paths and Tree-Like Structures*. Ph.D. Thesis, Institute of Discrete Mathematics and Geometry, Vienna University of Technology, Vienna, Austria, 2016.
- Deutsch, E. Dyck path enumeration. *Discret. Math.* **1999**, *204*, 167–202. [[CrossRef](#)]
- Merlini, D.; Sprugnoli, R.; Verri, M.C. Some statistics on Dyck paths. *J. Statist. Plann. Inference* **2002**, *101*, 211–227. [[CrossRef](#)]
- Mansour, T. Statistics on Dyck paths. *J. Integer Seq.* **2005**, *9*, 06.1.5.
- Banderier, C.; Wallner, M. Lattice paths with catastrophes. *Electron. Notes Discret. Math.* **2017**, *59*, 131–146. [[CrossRef](#)]
- Prodinger, H. Skew Dyck paths with catastrophes. *Discret. Math. Lett.* **2022**, *10*, 9–13.
- Baril, J.-L.; Kirgizov, S.; Marechal, R.; Vajnovszki, V. Enumeration of Dyck paths with air pockets. *J. Integer Seq.* **2023**, *26*, 23.3.2.
- Shablya, Y.; Kruchinin, D. Euler–Catalan’s number triangle and its application. *Symmetry* **2020**, *12*, 600. [[CrossRef](#)]

12. Banderier, C.; Schwer, S. Why Delannoy numbers? *J. Statist. Plann. Inference* **2005**, *135*, 40–54. [[CrossRef](#)]
13. Shapiro, L.W.; Sulanke, R.A. Bijections for the Schroder numbers. *Math. Mag.* **2000**, *73*, 369–376. [[CrossRef](#)]
14. Oste, R.; Van der Jeugt, J. Motzkin paths, Motzkin polynomials and recurrence relations. *Electron. J. Combin.* **2015**, *22*, P2.8. [[CrossRef](#)]
15. Chen, W.Y.C.; Deng, E.Y.P.; Yang, L.L.M. Riordan paths and derangements. *Discret. Math.* **2008**, *308*, 2222–2227. [[CrossRef](#)]
16. Baril, J.-L.; Kirgizov, S.; Petrossian, A. Enumeration of Lukasiewicz paths modulo some patterns. *Discret. Math.* **2019**, *342*, 997–1005. [[CrossRef](#)]
17. Banderier, C.; Flajolet, P. Basic analytic combinatorics of directed lattice paths. *Theoret. Comput. Sci.* **2002**, *281*, 37–80. [[CrossRef](#)]
18. Banderier, C.; Gittenberger, B. Analytic combinatorics of lattice paths: Enumeration and asymptotics for the area. *Discrete Math. Theoret. Comput. Sci.* **2006**, *AG*, 345–356. [[CrossRef](#)]
19. Banderier, C.; Krattenthaler, C.; Krinik, A.; Kruchinin, D.; Kruchinin, V.; Nguyen, D.; Wallner, M. Explicit formulas for enumeration of lattice paths: Basketball and the kernel method. In *Lattice Path Combinatorics and Applications*; Andrews, G.E., Krattenthaler, C., Krinik, A., Eds.; Springer: Cham, Switzerland, 2019; pp. 78–118.
20. Asinowski, A.; Bacher, A.; Banderier, C.; Gittenberger, B. Analytic combinatorics of lattice paths with forbidden patterns, the vectorial kernel method, and generating functions for pushdown automata. *Algorithmica* **2020**, *82*, 386–428. [[CrossRef](#)]
21. Dziemianczuk, M. On directed lattice paths with vertical steps. *Discret. Math.* **2016**, *339*, 1116–1139. [[CrossRef](#)]
22. Ruskey, F. Combinatorial Generation. Available online: <https://page.math.tu-berlin.de/~felsner/SemWS17-18/Ruskey-Comb-Gen.pdf> (accessed on 1 May 2023).
23. Liebehenschel, J. Ranking and unranking of a generalized Dyck language and the application to the generation of random trees. *Sem. Lothar. Combin.* **2021**, *43*, B43d.
24. Liebehenschel, J. Lexicographical generation of a generalized Dyck language. *SIAM J. Comput.* **2003**, *32*, 880–903. [[CrossRef](#)]
25. Duchon, P. On the enumeration and generation of generalized Dyck words. *Discret. Math.* **2000**, *225*, 121–135. [[CrossRef](#)]
26. Parque, V.; Miyashita, T. An efficient scheme for the generation of ordered trees in constant amortized time. In Proceedings of the 15th International Conference on Ubiquitous Information Management and Communication (IMCOM), Seoul, Republic of Korea, 4–6 January 2021.
27. Barucci, E.; Bernini, A.; Pinzani, R. Exhaustive generation of some lattice paths and their prefixes. *Theoret. Comput. Sci.* **2021**, *878–879*, 47–52. [[CrossRef](#)]
28. Kuo, T. From enumerating to generating: A linear time algorithm for generating 2D lattice paths with a given number of turns. *Algorithms* **2015**, *8*, 190–208. [[CrossRef](#)]
29. Knuth, D.E. *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1*; Addison-Wesley Professional: Boston, MA, USA, 2011.
30. Kreher, D.L.; Stinson, D.R. *Combinatorial Algorithms: Generation, Enumeration, and Search*; CRC Press: Boca Raton, FL, USA, 1999.
31. Bacchelli, S.; Barucci, E.; Grazzini, E.; Pergola, E. Exhaustive generation of combinatorial objects by ECO. *Acta Inform.* **2004**, *40*, 585–602. [[CrossRef](#)]
32. Flajolet, P.; Zimmerman, P.; Cutsem, B. A calculus for the random generation of combinatorial structures. *Theoret. Comput. Sci.* **1994**, *132*, 1–35. [[CrossRef](#)]
33. Hartung, E.; Hoang, H.; Mutze, T.; Williams, A. Combinatorial generation via permutation languages. I. Fundamentals. *Trans. Amer. Math. Soc.* **2022**, *375*, 2255–2291. [[CrossRef](#)]
34. Shablya, Y.; Kruchinin, D.; Kruchinin, V. Method for developing combinatorial generation algorithms based on AND/OR trees and its application. *Mathematics* **2020**, *8*, 962. [[CrossRef](#)]
35. Shablya, Y. Combinatorial generation algorithms for some lattice paths using the method based on AND/OR trees. *Algorithms* **2023**, *16*, 266. [[CrossRef](#)]
36. Shablya, Y.; Merinov, A. Bijection between simple directed lattice paths and AND/OR tree structures. In Proceedings of the 6th Mediterranean International Conference of Pure & Applied Mathematics and Related Areas (MICOPAM), Paris, France, 23–27 August 2023.
37. Mohanty, G. *Lattice Path Counting and Applications*; Academic Press: New York, NY, USA, 1979.
38. The On-Line Encyclopedia of Integer Sequences. Available online: <https://oeis.org/> (accessed on 1 January 2024).
39. Bousquet-Melou, M.; Petkovsek, M. Linear recurrences with constant coefficients: The multivariate case. *Discret. Math.* **2000**, *225*, 51–75. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.