FULL LINEARIZATION OF RANKING AND UNRANKING OF CATALAN CIPHER VECTORS USING CATALAN TRIANGLE ABSTRACTION

Adrijan Božinovski[1], Biljana Stojčevska[2]
[1]Faculty of Informatics, American University of Europe, Skopje, Macedonia, adrijan.bozinovski@fon.edu.mk
ORCID 0000-0003-2820-4016
[2]Faculty of Informatics, American University of Europe, Skopje, Macedonia, biljana.stojcevska@fon.edu.mk
ORCID 0009-0001-8915-6516

Original scientific paper
https://doi.org/10.7251/JIT2402136B UDC: 512.642/.644:514.122/.123

Abstract: This paper demonstrates how to abstract a version of the Catalan Triangle necessary to compute the rank value from a given Catalan Cipher Vector, which is a process called ranking, and the process of obtaining a Catalan Cipher Vector from a given rank value, which is a process called unranking. That version of the Catalan Triangle is called the Bottom Ones Catalan Triangle and is not required to be computed in its entirety for the purpose of ranking and unranking, but only elements of it that are required for the current computation. A formula is derived to compute each element of this triangle and it is demonstrated how the processes of both ranking and unranking are fully linear.

Keywords: Bottom Ones Catalan Triangle, abstraction, ranking, unranking, Catalan Cipher Vector

INTRODUCTION

The Catalan Triangle is a number triangle which is commonly used when dealing with problems related to combinatorics, in particular the ones involving counting. It is closely related to the Catalan Numbers [1], since, by default, their sequence is found following the first (i.e., longest) and second diagonal of this triangle. Other diagonals of the Catalan Triangle also produce integer sequences of their own [2, 3, 4, 5, 6, 7]. The *M*-th Catalan Number is computed as

$$C_M = \frac{1}{(M+1)}\binom{2\cdot M}{M} = \frac{(2\cdot M)!}{M!\cdot(M+1)!} \qquad (1)$$

Besides the version of the Catalan Triangle which is considered the default one [8], other versions of it also exist (e.g., [9, 10, 11] etc). One of the features of every Catalan Triangle is the existence of a sequence of elements all having the value of 1, i.e., *the all 1's sequence* [12], alongside the triangle's row, column or diagonal, depending on the version of the triangle. Thus, both the all 1's sequence and the Catalan Numbers sequence are found in every Catalan Triangle. Each of them can be placed horizontally, vertically or diagonally, depending on the version of the Catalan Triangle being used. Notably, the all 1's sequence and the Catalan Numbers sequence are always placed at extremes of the Catalan Triangle, i.e., at the longest diagonal, the leftmost column and/or the bottom row.

Of interest for this paper is a version of the Catalan Triangle where the longest diagonal is removed, thus leaving only one sequence of the Catalan Numbers instead of two. Three variations of such a triangle can be found: a) where the all 1's sequence is located at the leftmost column in the triangle [13, 14]; b) where the all 1's sequence is located at the longest diagonal of the triangle [15, 16]; and c) where the all 1's sequence is located at the bottom row of the triangle [17]. All of these variations are shown in Figure 1.

1				1				14			
1	2			2	1			9	5		
1	3	5		5	4	1		4	3	2	
1	4	9	14	14	14	6	1	1	1	1	1
	a)				b)				c)		

Figure 1. *Three variations of the Catalan Triangle with the first diagonal removed.*

The version of the Catalan Triangle with the first diagonal removed is useful for the purposes of ranking and unranking various enumerations of combinatorial data structures. Of interest for this paper is the ranking of binary trees enumerated by Catalan Cipher Vectors [17]. Catalan combinations [13] and Codewords [18, 19] can also be used as enumerations, as it is possible to directly transform them into Catalan Cipher Vectors and vice versa, as well as each other [17].

The process of obtaining the rank from a given Catalan Cipher Vector, i.e., CCV, is called ranking, whereas the process of obtaining the CCV from a given rank is called unranking. Technically, ranking can be defined as a function by which a number (i.e., integer) is obtained from a vector, and the unranking process is the reverse process. A requirement for both the ranking and unranking processes, i.e., algorithms, of CCVs is that a certain version of the Catalan Triangle – specifically the one displayed in Fig. 1c – be available. This version will be called the **Bottom Ones Catalan Triangle** or **BOCT** in this paper. The general algorithm for obtaining the BOCT, as well as the BOCT itself, have quadratic time and space complexity, respectively [17].

However, the entire BOCT is not required in order to obtain the rank of a CCV, but only elements of it which are linearly dependent on the CCV for which the rank is computed. The same is true for the unranking process as well: only the values of the BOCT that are necessary to generate the CCV from a given rank are needed to be used, depending on the rank value that needs to be unranked. Therefore, the entire BOCT would be abstracted if it were possible to compute only the values of the required elements of the BOCT as they become needed. The purpose of this paper is to demonstrate such an approach.

The side of the Catalan Triangle which contains the all 1's sequence will be referred to as the Edge of the Catalan Triangle, and the side that contains the Catalan Numbers sequence will be referred to as the Diagonal of the Catalan Triangle. This will pertain to both the default Catalan Triangle and the BOCT.

Converting the Default Catalan Triangle into the Bottom Ones Catalan Triangle

Figure 2 shows the default Catalan Triangle and the BOCT compared to each other, both with the size *N*=4. The row and column indices for the default Catalan Triangle and the BOCT are stated as (*n*, *k*) and (*r*, *c*), respectively.

n\k	*0*	*1*	*2*	*3*	*4*
0	1				
1	1	1			
2	1	2	2		
3	1	3	5	5	
4	1	4	9	14	14

r\c	*0*	*1*	*2*	*3*
0	14			
1	9	5		
2	4	3	2	
3	1	1	1	1

Figure 2. *The default Catalan Triangle and the BOCT with N=4.*

Equation (2) shows the formula for calculating the value of the element with indices (*n*, *k*) in the default Catalan Triangle [20]:

$$C(n,k) = \frac{n-k+1}{n+1}\binom{n+k}{n} \tag{2}$$

The BOCT is obtained from the default Catalan Triangle by: 1) omitting the longest diagonal from it; 2) transposing it; and 3) inverting it alongside the horizontal. Following are insights in order to be able to perform such a conversion.

In the default Catalan Triangle, on the row with index *n*=0 there is just one element, which is a part of the longest diagonal. Thus, if the starting rows index in the default Catalan Triangle is set to be 1, this will have the effect of removing the longest diagonal from the default Catalan Triangle. This is the transform that is required to be applied as far as the rows are concerned.

If the starting index of the columns in the default Catalan Triangle is taken to be 1, this has the effect of accessing elements in the subsequent (i.e., „to the right") column to the one of interest, when using (2). One of the features of the default Catalan Triangle is that each internal element (i.e., element not on the Edge or the Diagonal) is a sum of the element to the left of it and above it. In other words, C(*n*, *k*) = C(*n*, *k*-1) + C(*n*-1, *k*). Placing *k*+1 instead of *k* results in C(*n*, *k*+1) = C(*n*, *k*) + C(*n*-1, *k*+1), or, stated differently,

$$C(n,k) = C(n,k+1) - C(n-1,k+1) \tag{3}$$

However, if *k* starts from 1 instead of 0, then *k*+1 becomes *k* in the right-hand side of (3). So, if the *k*+1 on the right-hand side of (3) is replaced with *k*, it will be a valid substitution, provided that 1 is used for the starting index for the columns of the default Catalan Triangle instead of 0.

Restating (3) by replacing k+1 with k, implementing (2) into it and using the factorial representation of the combination term produces

$$C(n,k) = \frac{(n-k+2)\cdot(n+k-1)!}{(k-1)!\cdot(n+1)!} \tag{4}$$

Thus, it is possible to compute the value of an element in the Catalan Triangle using the values of the elements in the adjacent column. So, if the starting column of the default Catalan Triangle is 1 instead of 0, (4) should be used instead of (2) to compute the value of each element of the triangle. This is the transform that is required to be applied as far as the columns are concerned.

This way, all of the requirements can be met: 1) the effect of omitting the first diagonal from the Catalan Triangle can be achieved by having the starting index of the default Catalan Triangle be (1, 1) instead of (0, 0), provided that (4) is used to compute the value of the required element; 2) the transposition effect can be achieved by exchanging the values for the row and column in (4); and 3) the inversion effect can be achieved when substituting $n = N - r$ and $k = N - c$. After implementing these transformations, the final formula for calculating the value of the element in the BOCT with indices for the row and column (r, c) becomes

$$BOCT(r,c) = \frac{(r-c+2)\cdot(2\cdot N-r-c-1)!}{(N-r-1)!\cdot(N-c+1)!} \tag{5}$$

It is now possible to display an example of a BOCT and compare its indices with those of a default Catalan Triangle. Figure 3 shows the case for the size N=4. The indices of the BOCT range from (0, 0) to (3, 3), whereas those of the default Catalan Triangle, because of the inversion transformation, range from (4, 4) to (1, 1). The transposition is demonstrated by the switched indices in the default Catalan Triangle.

k\n		4	3	2	1
	r\c	0	1	2	3
4	0	14			
3	1	9	5		
2	2	4	3	2	
1	3	1	1	1	1

Figure 3. *A BOCT with size N=4, superimposed with the corresponding indices of the default Catalan Triangle of the same size.*

For r=c=d, where $0 \le d \le N-1$, (5) transforms into

$$BOCT(d,d) = \frac{2\cdot(2\cdot(N-d)-1)!}{((N-d)-1)!\cdot((N-d)+1)!} \tag{6}$$

which is the formula for obtaining the values of the elements along the diagonal of the BOCT (which is the reason for the bounds $0 \le d \le N-1$). Those elements are the Catalan Numbers, and substituting $M = N-d$ gives

$$C_M = \frac{2\cdot(2\cdot M-1)!}{(M-1)!\cdot(M+1)!} \tag{7}$$

which is another formula for computing the Catalan Numbers. Multiplying by M in the numerator and denominator on the right hand side of (7) transforms it into (1).

Abstraction of the Bottom Ones Catalan Triangle

As can be seen in (5), the factorial function needs to be invoked three times in order to compute the value of the BOCT element with indices (r, c). The algorithms for ranking and unranking are linear, provided that the BOCT had already been generated and its elements are available for access [17]. Since the factorial is a compounding function and is linear by itself as well, the goal becomes to make it possible to access the factorial of a given number in constant time, so that the ranking and unranking algorithms remain linear.

An analysis of (5) shows that the largest factorial term is $(2\cdot N-1)!$, which is found in the numerator, and is obtained for r=c=0; the factorial terms in the denominator thus become $(N-1)!$ and $(N+1)!$ respectively. Given that the definition of the factorial is $N! = N\times (N-1)!$, where $0! = 1$, it follows that it is necessary to compute the value of $(N-1)!$ in order to compute the value of $(N+1)!$, which in turn needs to be computed in order to compute the value of $(2\cdot N-1)!$. The aforementioned holds true for $N \ge 2$, which is a necessary requirement for a Catalan Triangle to exist (a Catalan Triangle of just one element is essentially just the number 1).

Thus, if the values of the factorials of all positive integers up to and including $(2\cdot N-1)!$ are available to be used on demand, the computation of (5) is done in constant time. This can be achieved using a linear data structure which can access the values of its elements using indexing, e.g., an array. Given that $0! = 1$, the value of the element with index 0 is initially set to be 1, whereas the value of each subsequent element is computed as per the factorial function definition. Both the process of populating such a structure and the structure itself have linear time and space complexities. Figure 4 shows an example of such a structure for $N = 4$.

f[0]	f[1]	f[2]	f[3]	f[4]	f[5]	f[6]	f[7]
1	1	2	6	24	120	720	5040

Figure 4. *An example for N = 4: an array of all factorials up to and including (2×4 – 1)! = 7! would be sufficient to compute the values of all the elements of the BOCT required to rank and/or unrank any CCV of size N = 4.*

Fully Linear Algorithms for Ranking and Unranking of Catalan Cipher Vectors

The following algorithms are modifications to the ranking and unranking algorithms found in [17], which are shown to be linear there. The algorithms in this paper focus on the ranking and unranking processes instead on the full generation of binary trees using the CCVs. They also utilize the abstraction approach explained in the previous sections. They are presented as functions and pseudocode (which resembles the C++ language) is given for each of them.

During the ranking and unranking processes, the elements in a Catalan Cipher Vector are sequentially read and updated, respectively. In both cases, the size of the CCV is taken to be N, and this value is assumed to be known and stored in the variable N. The array variable *CCV*, of size N, which stores the elements of the CCV currently being processed, is also assumed to be initialized and populated with valid values, as defined in [17].

The Bottom Ones Catalan Triangle Element Value Computation Function

Pseudocode 1 presents the function for computing the value of the BOCT element that is needed for the ongoing ranking or unranking process. This function is auxiliary but is called during both the ranking and unranking process. The parameters are: N (the size of the CCV, as previously introduced); r – the row index of the element in the BOCT being accessed; and c – the column index of the element in the BOCT being accessed.

Pseudocode 1. *The function for the calculation of the BOCT element given the size of the BOCT N, the row of the element r and the column of the element c.*

```
function BOCTel(N, r, c)
{
    num = (r-c+2) * f[2*N-r-c-1];
    den = f[N-r-1] * f[N-c+1];
    return num / den;
}
```

The Ranking Function

Pseudocode 2 shows the ranking function for a given array parameter *CCV*. Variables additional to the ones already known are: v – the index in the *CCV* array; and *rank* – the value of the rank being computed during the ranking process.

Pseudocode 2. *The ranking function.*

```
function Rank(CCV)
{
    r = 0;
    c = 0;
    rank = 0;
    for(v = 0; v < N; ++v)
    {
        while (CCV[v] > (r + c))
            rank += BOCTel(N, r, c++);
        ++r;
    }
    return rank;
}
```

The Unranking Function

Pseudocode 3 shows the unranking function for a given parameter *rank*.

Pseudocode 3. *The unranking function.*

```
function Unrank(rank)
{
    r = 1;
    c = 0;
    v = 0;
    CCV = new array[N];
    CCV[v++] = 0;
    while(v < N)
    {
        while ($BOCTel(N, r, c)$ <= $rank$)
            $rank$ -= $BOCTel(N, r, c$++);
        $CCV[v$++] = $r$++ + $c$;
    }
    return $CCV$;
}
```

Analysis of the Algorithms

Both the ranking and unranking algorithms traverse the CCV linearly and they access only the elements of the BOCT that are required during the corresponding process [17]. For each element of the CCV being processed, a corresponding element in each of the rows on the BOCT is accessed. Depending on the value of the element in the CCV, there is either no displacement from the left vertical (i.e., the Edge) of the BOCT, or there is some, up to at most the value of the index of the row (i.e., the element on the Diagonal). However, once that displacement occurs when processing an element of the CCV, the displacement is not reset in the subsequent row.

This means that the required elements in the BOCT are traversed in a single line, without segments of the line being repeated or intersected. This ensures that, in the best case, only the elements of the leftmost column will be traversed, so the time complexity of both the ranking and unranking algorithm will be W(N). In the worst case, the full lengths of both the column and the bottom row will be traversed, so the time complexity for both algorithms will be O($2 \cdot N$) = O(N). Thus, the time complexity of both algorithms is Q(N), even though that may not be apparent upon initial overviews of their respective pseudocodes. Given that the preliminary process of generating the array of factorials for positive integers from 0 to $2 \cdot N-1$ is also linear, this means that the full processes of both ranking and unranking, alongside all of their necessary algorithms and data structures, are linear for both time and space complexity.

Conclusion

The process of obtaining an integer for a given Catalan Cipher Vector is called ranking, and the reverse process is called unranking. This paper presents the principle and algorithms for abstraction of the Bottom Ones Catalan Triangle, which is a version of the Catalan Triangle that is used for the processes of ranking and unranking of Catalan Cipher Vectors. A formula is presented that computes the values of the elements of the Bottom Ones Catalan Triangle as they become necessary during the ranking and unranking processes. This way, the triangle doesn't need to be generated fully and is therefore abstracted during the ranking and unranking processes, thus circumventing the default quadratic time and space requirements for its computation. Instead, a linear data structure, such as an array, is used for storing values of factorials necessary for such computations, thus making both the ranking and unranking processes fully linear for both time and space complexity.

Acknowledgment

Professor Veno Pačovski, who passed away in 2020, was the third author in the paper that has introduced the Catalan Cipher Vectors in 2013. The authors of this paper would like to remember him as a person full of vitality, energy and drive to promote and improve Computer Science, like he used to be back then.

References

[1] L. Comtet, *Advanced Combinatorics,* Reidel, 1974.

[2] K. H. Kim, D. G. Rogers and F. W. Roush, "Similarity relations and semiorders", Proceedings of the Tenth Southeastern Conference on Combinatorics, Graph Theory and Computing, Boca Raton, FL, pp. 577-594, 1979.

[3] R Sedgewick and P. Flajolet, *Analysis of Algorithms*, Addison Wesley, 1996.

[4] C. Krishnamachary and M. Bheemasena Rao, "Determinants Whose Elements are Eulerian, Prepared Bernoullian and Other Numbers", J. Indian Math. Soc., Vol. 14, 1922.

[5] N. J. A. Sloane and S. Plouffe, *The Encyclopedia of Integer Sequences*, Academic Press, 1995.

[6] D. E. Davenport, L. W. Shapiro, L. K. Pudwell and L. C. Woodson, "The Boundary of Ordered Trees", J. Integer Seq., Vol. 18, 2015.

[7] H. H. Gudmundsson, "Dyck paths, standard Young tableaux, and pattern avoiding permutations", Pure Mathematics and Applications, Vol. 21, No.2, pp. 265-284, 2010.

[8] A. Proskurowski and E. Laiman, "Fast enumeration, ranking, and unranking of binary trees", Proceedings of the Thirteenth Southeastern Conference on Combinatorics, Graph Theory and Computing, Boca Raton, FL, pp. 401-413, 1982.

[9] Y. Shablya, D. Kruchinin and V. Kruchinin, "Method for Developing Combinatorial Generation Algorithms Based on AND/OR Trees and Its Application", Mathematics, Vol. 8, No. 6, ar. 962, 2020.

[10] The On-Line Encyclopedia of Integer Sequences, published electronically at https://oeis.org, Sequence A181645, 2010.

[11] R. K. Guy, "Catwalks, Sandsteps and Pascal Pyramids", J. Integer Sequences, Vol. 3, 2000.

[12] A. M. Hinz, S. Klavžar, U. Milutinović, and C. Petr, *The Tower of Hanoi - Myths and Maths*, Birkhäuser 2013.

[13] M. Črepinšek and L. Mernik, “An efficient representation for solving Catalan Number related problems”, Int. Jour. Pure Appl. Math., vol. 56, no. 4, pp. 589–604, 2009.
[14] B. Derrida, E. Domany and D. Mukamel, “An Exact Solution of a One-Dimensional Asymmetric Exclusion Model with Open Boundaries”, J. Stat. Phys., Vol. 69, pp. 667-687, 1992.
[15] L. W. Shapiro, “A Catalan Triangle”, Discr. Math. 14, pp. 83–90, 1976.
[16] B. A. Bondarenko, *Generalized Pascal Triangles and Pyramids, Their Fractals, Graphs and Applications*, The Fibonacci Association, 1990.
[17] A. Božinovski, B. Stojčevska and V. Pačovski, “Enumeration, Ranking and Generation of Binary Trees Based on Level-Order Traversal Using Catalan Cipher Vectors”, Jour. Inf. Tech. Appl., vol. 3, no. 2, pp. 78–86, 2013.
[18] D. Roelants van Baronaigen, “A loopless algorithm for generating binary tree sequences”, Inf. Proc. Lett. 39, pp. 189–194, 1991.
[19] D. Zerling, “Generating Binary Trees Using Rotations”, Jour. ACM, vol, 27, pp. 694–701, 1985.
[20] D. F. Bailey, “Counting Arrangements of 1’s and -1’s”, Mathematics Magazine, vol. 69, no. 2, pp. 128–131, 1996.

Received: November 13, 2024
Accepted: November 23, 2024

About the authors

Adrijan Božinovski works as a Full Professor at the Faculty of Informatics at the American University of Europe. He obtained his BSc from University “Ss. Cyril and Methodius” in Skopje, Macedonia, and his MSc and PhD from University of Zagreb, Croatia. His research interests include Robotics, Artificial Intelligence, Biomedical Engineering, Data Structures and Algorithms.

Biljana Stojcevska works as a Full Professor at the American University of Europe in Skopje. She received her PhD degree in Computer Science at the Institute of Informatics, Faculty of Natural Sciences and Mathematics, at “Ss. Cyril and Methodius” University in Skopje. The areas of her interest are computer networks, network congestion management, operating systems and programming languages.

For citation

Adrijan Božinovski, Biljana Stojčevska, Full Linearization of Ranking and Unranking of Catalan Cipher Vectors Using Catalan Triangle Abstraction, *JITA – Journal of Information Technology and Applications, Banja Luka*, Pan-Europien University APEIRON, Banja Luka, Republika Srpska, Bosna i Hercegovina, JITA 14(2024)2:136-141, (UDC: 512.642/.644:514.122/.123), (DOI: 10.7251/JIT2402136B, Volume 14, Number 2, Banja Luka, December (89-188), ISSN 2232-9625 (print), ISSN 2233-0194 (online), UDC 004