Applicable Analysis and Discrete Mathematics
available online at http://pefmath.etf.rs

Appl. Anal. Discrete Math. **x** (xxxx), xxx–xxx.
https://doi.org/10.2298/AADM170829013M

ORBITING TRIANGLE METHOD FOR CONVEX POLYGON TRIANGULATION

Sead H. Mašović, Islam A. Elshaarawy, Predrag S. Stanimirović and Predrag V. Krtolica*

Finding all possible triangulations of convex polygon is a highly time and memory space consuming combinatorial problem. Therefore, it is very important to develop algorithms for generating triangulations as efficiently as possible. This paper presents a new method for generating triangulations of a convex polygon, called *Orbiting triangle method* (OTM). The method is based on using the set of $(n-1)$-gon triangulations during the set of n-gon triangulations creation. The main feature of the OTM algorithm is the use of the Catalan triangle to identify valid triangulations, so that the algorithm spends almost no time to eliminate duplicates. In this way, the method possesses small complexity and saves the computational time required for detecting and eliminating duplicates.

1. INTRODUCTION AND PRELIMINARIES

Triangulation is a fundamental problem in computational geometry, which is mainly initiated by its numerous applications in many important areas, such as the patten recognition, computer graphics, computer aided design, modelling, robotics and many other areas. In [**8**], authors presented an algorithm for the *Flip Distance* problem. A flip of an edge e in a triangulation is the operation of replacing e by another diagonal of a convex quadrilateral formed by the two triangles that share e. The flip distance between two triangulations is defined as the minimum number of edges flipping needed to transform one triangulation to another.

*Corresponding author. Sead H. Mašović
2010 Mathematics Subject Classification. 68U05, 11Y16, 68N15; .
Keywords: Convex polygon triangulation, Catalan numbers, Catalan triangle.

Triangulations of point sets in the plane have been studied in the last three decades as one of the important structures in computational geometry. There main optimality criteria is based on edge length, angles, areas and other elements of the individual triangles in a triangulation. The dynamic programming technique has been considered in the convex case. A dynamic programming algorithm, proposed in [**10**], is usable in finding the optimal triangulation with respect to a number of criteria. An alternative optimality was considered in [**9**], and it is based on solving the MinMax and MaxMin problems. Recall that the MaxMin (resp. MinMax) area triangulation is the triangulation of the polygon that maximizes the area of the smallest triangle (resp. minimizes the area of the largest area triangle) in the triangulation. An interesting combinatorial problem is determining the number of triangulations of a convex k-gon each of whose sides is subdivided by $r-1$ points, where k, r are two natural numbers, $k \geq 3, r \geq 1$. Let $SC(k,r)$ denote a convex k-gon in the plane each of whose sides is subdivided by $r-1$ points and $tr(k,r)$ denotes the number of triangulations of $SC(k,r)$. The number $tr(k,r)$ was investigated in [**1, 2, 7**]. Also, precise asymptotic results for $tr(k,r)$ was considered in [**1, 2**]. Negami in [**14**] showed that any two triangulations of a closed surface can be transformed into each other by means of diagonal flips. Authors in [**21**] presented algorithms for generating strictly binary trees at random based on convex polygon triangulations. Their results show that the numbers of various shapes of strictly binary trees generated are nearly equal. Authors of [**11**] introduced the term dual diameter of triangulations. They presented efficient algorithms for calculating the exact dual diameter of minimum and maximum dual diameter triangulations of convex polygons, which can be obtained by maximizing and minimizing the number of ears of the triangulation, respectively.

In a convex case, the polygon triangulation is the decomposition of the polygon into a set of triangles which are generated by non-intersecting internal diagonals of the polygon.

A convex polygon P_n is determined by its vertices denoted by $(v_1, v_2, \ldots, v_n)$. The n-gon P_n can be decomposed in $n-2$ triangles by its $n-3$ internal diagonals. An internal diagonal of P_n is defined by its ending vertices. It is assumed that a diagonal connecting vertices v_i and v_j is denoted by $\delta_{v_i,v_j} = (v_i, v_j)$.

The set of triangulations of P_n is denoted by

$$\mathcal{T}_n = \{t_n^p \mid p = 1, \ldots, C_{n-2}\}, \tag{1}$$

where

$$C_{n-2} = \frac{1}{n-1}\binom{2n-4}{n-2} = \frac{(2n-4)!}{(n-1)!(n-2)!}, \quad n \geq 3 \tag{2}$$

denotes the cardinality of the set $\mathcal{T}_n$ and C_n represents the nth Catalan number (about the topic see e.g. [**12**]).

Here we state the general rule for the Catalan triangle construction:

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

where n is a nonnegative integer and $k = 0, \ldots, n$. For example, the 8×8 Catalan triangle is presented in Table 1. For more details, see e.g. [**4, 18**].

n\k	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0	1								
1	1	1							
2	1	2	2						
3	1	3	5	5					
4	1	4	9	14	14				
5	1	5	14	28	42	42			
6	1	6	20	48	90	132	132		
7	1	7	27	75	165	297	429	429	
8	1	8	35	110	275	572	1001	1430	1430

Table 1: Some values of Catalan triangle.

The values from the Catalan triangle are used for efficiently detecting and eliminating duplicate triangulation in the OTM method.

The algorithm of the OTM method is presented in Section 2 and it is compared to another method for convex polygon triangulation introduced by Hurtado and Noy in [**6**]. For this reason and the sake of completeness, we restate here Hurtado-Noy algorithm, in the form presented in [**19**].

Algorithm 1 Hurtado-Noy algorithm

Require: Positive integer n and the set $\mathcal{T}_{n-1}$ of P_{n-1} triangulations. Each triangulation is described as a structure containing $2n-5$ vertex pairs presenting P_{n-1} diagonals (here *diagonals* means both internal diagonals and outer face edges).

1: Check the structure containing $2n-5$ vertex pairs looking for pairs $(i_k, n-1)$, $i_k \in \{1, 2, \ldots, n-2\}$, $2 \leq k \leq n-2$, i.e. diagonals incident to vertex $n-1$. The positions of these indices i_k within the structure describing a triangulation should be stored in the array.
2: For every i_k perform the transformation $(i_l, n-1) \to (i_l, n)$, $i_l < i_k$, $0 \leq l \leq n-3$.
3: Insert new pairs (i_k, n) and $(n-1, n)$ into the structure.
4: Take next i_k, if any, and go to Step (2).
5: Continue the above procedure with next $(n-1)$-gon triangulation (i.e. structure with $2n-5$ vertex pairs) if any. Otherwise halt.

A number of various algorithms for solving the convex polygon triangulation has been developed. Such an algorithm is the Hurtado-Noy algorithm, proposed by Hurtado and Noy in [**6**]. Authors of [**6**] defined a triangulation tree, where all triangulations $\mathcal{T}_n$ of the polygon P_n are arranged at the nth level of the triangulation

tree. Each triangulation on the nth level has "father" in the $\mathcal{T}_{n-1}$ set and two or more "sons" in the $\mathcal{T}_{n+1}$ set. The *Block method* for generating $\mathcal{T}_n$, proposed in [**19**], is based on the usage of the previously generated triangulations $\mathcal{T}_b$ for polygons P_b, $b < n$ and decomposes the triangulation problem into subproblems which represent smaller instances of the starting problem. A decomposition of the Catalan number ($T_n = C_{n-2}$) into the sum of terms of the form $(2 + i)$, $i \in \{0, \ldots, n-4\}$ was presented in [**20**]. Also, in [**20**], every expression of the form $(2+i)$ was assigned to each edge in the tree of triangulations from [**6**]. As a consequence, the authors of [**20**] introduced the method of generating triangulations $\mathcal{T}_n$ upon already generated triangulations $\mathcal{T}_{n-1}$ using the previously defined weighted tree of triangulations. A method for generating a convex polygon triangulations and their notation, which is based on ballot records, was proposed in [**17**]. Implementations of the Hurtado-Noy algorithm from [**6**], in three programming languages (Java, Python, C++), were described and compared in [**16**].

Recursive convex polygon triangulation algorithms that are based on cutting out one triangle and triangulating the remaining polygon, in general, generate a lot of duplicate triangulations. Moreover, the number of duplicate triangulations rapidly increases with increasing the number of vertices n. One approach in eliminating duplicate triangulations was proposed in [**13**], and it is based on the database usage in order to prevent duplicate triangulations recording. In general, it is known that storing in a database and frequent search in it can slow down the execution. Especially, this drawback is noticeable in large databases produced by a large number of generated triangulations.

In order to remedy the above mentioned drawback and improve the efficacy in eliminating duplicates, we propose a new method, called the OTM method, for the triangulation of P_n. The basic strategy of the OTM method is the usage of the triangulations from $\mathcal{T}_{n-1}$ in order to generate the triangulations in $\mathcal{T}_n$. The main advantage of the OTM method is a non-costly approach for eliminating duplicates. The approach is based on the values included in the Catalan triangle.

The global organization of the rest of the paper is as follows. In Section 2, we present the OTM algorithm and prove its correctness. Complexity of the OTM algorithm is investigated in Section 3. The execution times for the OTM, Block method and Hurtado algorithm are given and compared in Section 4.

2. ALGORITHM FOR ORBITING TRIANGLE METHOD

According to the definition of an *ear* of a simple polygon P stated in [**3**], an ear e of P is a triangle formed by three consecutive vertices such that one of its edges is a diagonal of P and the remaining two edges of e are also edges of P. According to definition of an ear in [**5**] or [**15**], an ear at the vertex v_2 of a convex polygon presents a triangle (v_1, v_2, v_3) made of three consecutive vertices v_1, v_2, v_3. Then the line segment $\delta_{v_1,v_3} = (v_1, v_3)$ between v_1 and v_3 represents a diagonal of the polygon. If we start from the initial ear $(1, 2, 3)$ and turn it around the given

polygon P_n in the clockwise direction in all possible ways (gray areas in Figure 1), we get all possible ears. The number of triangulations of a convex polygon with a prescribed number of ears was defined in [**5**]. Clearly, cutting out any ear from P_n produces the P_{n-1} polygon, as it is shown in Figure 1.

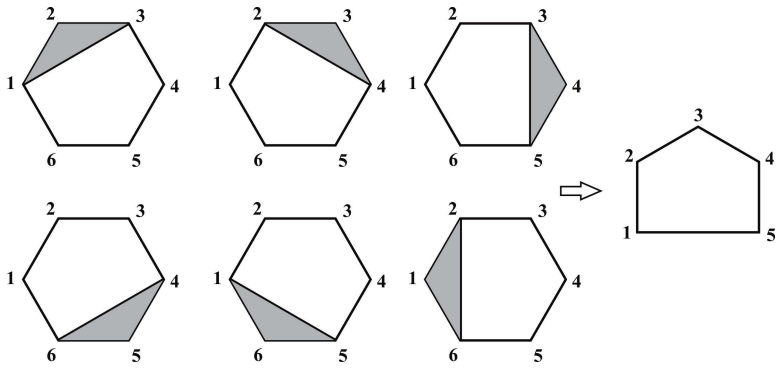


Figure 1: Orbiting polygon P_6 and cutting the ear.

The ears of orbiting the polygon give a background for our method. Obviously, all ears in P_n are defined by the set of ordered triples of vertices, as follows:

$$\mathcal{E} = \{(1,2,3),(2,3,4),\ldots,(n-2,n-1,n),(n-1,n,1),(n,1,2)\}\,. \tag{4}$$

Since each ear (i,j,k) from $\mathcal{E}$ is uniquely determined by a certain internal diagonal $\delta_{i,k} = (i,k)$ of P_n, the result of an ear orbiting the P_n polygon is a set of internal diagonals $\delta_{i,k}$ of generated ears in $\mathcal{E}$, called `OT-list`. More precisely, the `OT-list` of P_n is equal to the following ordered list:

$$\{\delta_{1,3},\delta_{2,4},\ldots,\delta_{n-3,n-1},\delta_{n-2,n},\delta_{n-1,1},\delta_{n,2}\}\,.$$

The OTM method exploits only the first $n-2$ elements from the `OT-list`. The list of these elements is denoted by ℓ_n and possesses the form

$$\ell_n = \{\delta_{1,3},\delta_{2,4},\ldots,\delta_{n-3,n-1},\delta_{n-2,n}\} = \{\delta_{k,k+2} \mid k = 1,\ldots,n-2\}\,. \tag{5}$$

It is very important to preserve the order of the triangulations from $\mathcal{T}_{n-1}$ according to the order of internal diagonals ℓ_n in the process of generating triangulations $\mathcal{T}_n$. More precisely, the triangulations from $\mathcal{T}_{n-1}$ starting with $\delta_{1,3}$ are used first, then the triangulations starting with $\delta_{2,4}$ and finally triangulations starting with $\delta_{n-3,n-1}$.

First operation used in the OTM method is *complementing triangulation*, which is used for mapping the internal diagonals of P_{n-1} by the rule

$$\mathfrak{C}_{n-1}(\delta_{i,j}) = \delta_{n-i,n-j},\quad i,j \in \{1,\ldots,n-1\},\ |i-j|>1. \tag{6}$$

The complementing triangulation (6) maps the set $\mathcal{T}_{n-1}$ into the set denoted by ${\mathcal{T}_{n-1}}^c = \mathfrak{C}_{n-1}(\mathcal{T}_{n-1})$, where the mapping $\mathfrak{C}_{n-1}$ is element-wise applied on each internal diagonal included in

$$\mathcal{T}_{n-1} = \{t_{n-1}^p \mid \ p = 1, \ldots, C_{n-3}\}. \tag{7}$$

To explain this in more details, the set of internal diagonals which define the pth triangulation t_{n-1}^p is denoted by

$$t_{n-1}^p = \left\{\delta_{i_1,j_1}^p, \ldots, \delta_{i_{n-4},j_{n-4}}^p\right\}. \tag{8}$$

According to (7) and (8), the set of internal diagonals which define all triangulations in $\mathcal{T}_{n-1}$ is defined by ordered $(n-4)$-tuples of internal diagonals, as follows

$$\mathcal{T}_{n-1} = \left\{\left\{\delta_{i_1,j_1}^p, \ldots, \delta_{i_{n-4},j_{n-4}}^p\right\} \mid \ p = 1, \ldots, C_{n-3}\right\}.$$

This implies

$$\begin{aligned}
{\mathcal{T}_{n-1}}^c &= \mathfrak{C}_{n-1}(\mathcal{T}_{n-1}) \\
&= \left\{\left(t_{n-1}^p\right)^c \mid p = 1, \ldots, C_{n-3}\right\} \\
&= \left\{\left\{\left(\delta_{i_1,j_1}^p\right)^c, \ldots, \left(\delta_{i_{n-4},j_{n-4}}^p\right)^c\right\} \mid p = 1, \ldots, C_{n-3}\right\}.
\end{aligned}$$

The second important mapping used in OTM is called *rotating triangulation*. That mapping defines a shifting of all diagonal ending points (describing a particular triangulation) for $l \geq 1$ positions in the clockwise direction. Rotating a diagonal $\delta_{i,j} \in \mathcal{T}_{n-1}$ is defined as the transformation

$$\mathfrak{R}_{l,n-1}(\delta_{i,j}) = \delta_{1+(i+l) \bmod n, 1+(j+l) \bmod n} \in \mathcal{T}_n, \tag{9}$$

where values for l are taken from the set $\{1, \ldots, n-2\}$. The element-wise rotation of the set $\mathcal{T}_{n-1}$ for l positions are denoted by $\mathfrak{R}_{l,n-1}(\mathcal{T}_{n-1}) = {\mathcal{T}_{n-1}}^{r_l}$ and defined by

$$\begin{aligned}
{\mathcal{T}_{n-1}}^{r_l} &= \mathfrak{R}_{l,n-1}(\mathcal{T}_{n-1}) \\
&= \left\{\left(t_{n-1}^p\right)^{r_l} \mid p = 1, \ldots, C_{n-3}\right\} \\
&= \left\{\left\{\left(\delta_{i_1,j_1}^p\right)^{r_l}, \ldots, \left(\delta_{i_{n-4},j_{n-4}}^p\right)^{r_l}\right\} \mid p = 1, \ldots, C_{n-3}\right\}.
\end{aligned}$$

The OTM method is based on the usage of the set
$\mathfrak{R}_{l,n-1}({\mathcal{T}_{n-1}}^c) = {\mathcal{T}_{n-1}}^{c,r_l} = \left\{\left\{\left(\delta_{i_1,j_1}^p\right)^{c,r_l}, \ldots, \left(\delta_{i_{n-4},j_{n-4}}^p\right)^{c,r_l}\right\} \mid p = 1, \ldots, C_{n-3}\right\}$.

In order to better explain the transformation (9), let us note that each P_n can be considered in the form $P_n = P_{n-1} \cup (v_{j-1}, v_j, v_{j+1})$, where

$$\begin{aligned}
P_n &= \{v_1, \ldots, v_{j-1}, v_j, v_{j+1}, \ldots, v_n\}, \\
P_{n-1} &= \{u_1 := v_1, \ldots, u_{j-1} := v_{j-1}, u_j := v_{j+1}, \ldots, u_{n-1} := v_n\}.
\end{aligned} \tag{10}$$

According to (10), each triangulation from $\mathfrak{C}_{n-1}(\mathcal{T}_{n-1})$ is included in P_n with the separated ear (v_{j-1}, v_j, v_{j+1}), where the vertex v_j does not participate in the rotation. In fact, the rotation (9) is applied on u_i vertices and the final result is given in the set of v_i vertices.

In addition to complementing and rotating, it is necessary to concatenate the kth element from ℓ_n, equal to $\delta_{k,k+2}$, with complemented and rotated triangulations included in the set ${\mathcal{T}_{n-1}}^{c,r_k}$, for each $k = 1, \ldots, n-2$. This concatenation will be denoted by $\delta_{k,k+2} \circ {\mathcal{T}_{n-1}}^{c,r_k}$ and defined as

$$\delta_{k,k+2} \circ {\mathcal{T}_{n-1}}^{c,r_k} = \left\{ \delta_{k,k+2} \diamond \left(t_{n-1}^p\right)^{c,r_k} \;\mid\; t_{n-1}^p \in \mathcal{T}_{n-1}, p = 1, \ldots, C_{n-3} \right\},$$

where $\delta_{k,k+2} \diamond \left(t_{n-1}^p\right)^{c,r_k}$ assumes insertion of $\delta_{k,k+2}$ at the beginning of the triangulation $\left(t_{n-1}^p\right)^{c,r_k}$. More precisely, for each element from the set

$$\left(t_{n-1}^p\right)^{c,r_k} = \left\{ \left(\delta_{i_1,j_1}^p\right)^{c,r_k}, \ldots, \left(\delta_{i_{n-4},j_{n-4}}^p\right)^{c,r_k} \right\},$$

the insertion is defined as

$$\delta_{k,k+2} \diamond \left(t_{n-1}^p\right)^{c,r_k} = \left\{ \delta_{k,k+2}, \left(\delta_{i_1,j_1}^p\right)^{c,r_k}, \ldots, \left(\delta_{i_{n-4},j_{n-4}}^p\right)^{c,r_k} \right\}.$$

In this way, we get valid and unique triangulations keeping only the following subset of the set ${\mathcal{T}_{n-1}}^{c,r_k}$:

$$\Upsilon_{n-1}^k = \left\{ \left(t_{n-1}^p\right)^{c,r_k} \in {\mathcal{T}_{n-1}}^{c,r_k} \mid p = C_{n-3} - C(n-3, n-k-2) + 1, \ldots, C_{n-3} \right\}. \tag{11}$$

In this sense, it is necessary to generate the set

$$\delta_{k,k+2} \circ \Upsilon_{n-1}^k = \left\{ \delta_{k,k+2} \diamond \left(t_{n-1}^p\right)^{c,r_k} \mid p = C_{n-3} - C(n-3, n-k-2) + 1, \ldots, C_{n-3} \right\}, \tag{12}$$

for each $k = 1, \ldots, n-2$.

Clearly, it is necessary to rotate only these parts of the set $\mathcal{T}_{n-1}$ which are determined by the formula (11). The previous statement is the very key observation and it gives the strength to our method: the method eliminates the duplicates efficiently and significantly decreases the computational cost. The pseudocode of the OTM method is described in Algorithm 2.

Algorithm 2 Orbiting Triangle Method.

Require: A positive integer n and the set $\mathcal{T}_{n-1}$ of P_{n-1} triangulations (described by internal diagonal of the form of $\delta_{i,j}$).

1: Produce the list ℓ_n.

2: Read $\mathcal{T}_{n-1}$ triangulations from the input.

3: Generate triangulations $\mathcal{T}_n$ using the following three steps:
(a) Perform *complementing triangulation* on $\mathcal{T}_{n-1}$ and get $\mathfrak{C}_{n-1}(\mathcal{T}_{n-1})$.
(b) Generate the set of triangulations (11) from ${\mathcal{T}_{n-1}}^{c,r_k}$, for each $k = 1, \ldots, n-2$.
(c) Generate the set Υ_{n-1}^{k} according to (11) and generate the set $\delta_{k,k+2} \circ \Upsilon_{n-1}^{k}$, for each $k = 1, \ldots, n-2$.

4: Return the set of triangulations
$\mathcal{T}_n = \left\{\delta_{k,k+2} \circ \Upsilon_{n-1}^{k},\ k = 1, \ldots, n-2\right\}$.

Remark 1. *The ℓ_n is always of the form* (5) *and can be available by default.*

Example 1. Let us illustrate Algorithm 2 on generating hexagon triangulations using known pentagon triangulations. The list ℓ_6 is produced in Step 1 as

$$\ell_6 = (\delta_{1,3}, \delta_{2,4}, \delta_{3,5}, \delta_{4,6}).$$

The set $\mathcal{T}_5$ of pentagon triangulations is read in Step 2 of Algorithm 2, and it is given by

$$\mathcal{T}_5 = \{\{\delta_{1,3}, \delta_{1,4}\}, \{\delta_{1,3}, \delta_{5,3}\}, \{\delta_{2,4}, \delta_{2,5}\}, \{\delta_{2,4}, \delta_{1,4}\}, \{\delta_{3,5}, \delta_{2,5}\}\}.$$

Note that the triangulation ordering in $\mathcal{T}_5$ is initiated by the sequence

$$\delta_{1,3}, \delta_{2,4}, \delta_{3,5}$$

imposed by the first three elements of ℓ_6.

According to Step 3(a), it is necessary to complement diagonals from $\mathcal{T}_5$ making the set $\mathfrak{C}_5(\mathcal{T}_5) = {\mathcal{T}_5}^c$, which is presented in Table 2.

$\mathcal{T}_5$	$\delta_{i,j} \to \delta_{(6-i),(6-j)}$	${\mathcal{T}_5}^c$
$\{\delta_{1,3}, \delta_{1,4}\}$	$\{\delta_{(6-1),(6-3)}, \delta_{(6-1),(6-4)}\}$	$\{\delta_{5,3}, \delta_{5,2}\}$
$\{\delta_{1,3}, \delta_{5,3}\}$	$\{\delta_{(6-1),(6-3)}, \delta_{(6-5),(6-3)}\}$	$\{\delta_{5,3}, \delta_{1,3}\}$
$\{\delta_{2,4}, \delta_{2,5}\}$	$\{\delta_{(6-2),(6-4)}, \delta_{(6-2),(6-5)}\}$	$\{\delta_{4,2}, \delta_{4,1}\}$
$\{\delta_{2,4}, \delta_{1,4}\}$	$\{\delta_{(6-2),(6-4)}, \delta_{(6-1),(6-4)}\}$	$\{\delta_{4,2}, \delta_{5,2}\}$
$\{\delta_{3,5}, \delta_{2,5}\}$	$\{\delta_{(6-3),(6-5)}, \delta_{(6-2),(6-5)}\}$	$\{\delta_{3,1}, \delta_{4,1}\}$

Table 2: Set of pentagon triangulations and their complements.

According to Step 3(b), it is necessary to rotate the triangulations from ${\mathcal{T}_5}^c$ for $1, 2, 3$, and 4 positions. Of course, we rotate only the subset of ${\mathcal{T}_5}^c$ which

is defined by (11). Particulary, we will take only 5, 5, 3, and 1 bottom aligned triangulations from the columns 1, 2, 3, and 4, respectively. This is illustrated in Table 3.

Υ^1_{n-1}	Υ^2_{n-1}	Υ^3_{n-1}	Υ^4_{n-1}
$\{\delta_{5,3},\delta_{5,2}\}\to\{\delta_{1,5},\delta_{1,4}\}$	$\{\delta_{5,3},\delta_{5,2}\}\to\{\delta_{2,6},\delta_{2,5}\}$		
$\{\delta_{5,3},\delta_{1,3}\}\to\{\delta_{1,5},\delta_{3,5}\}$	$\{\delta_{5,3},\delta_{1,3}\}\to\{\delta_{2,6},\delta_{4,6}\}$		
$\{\delta_{4,2},\delta_{4,1}\}\to\{\delta_{6,4},\delta_{6,3}\}$	$\{\delta_{4,2},\delta_{4,1}\}\to\{\delta_{1,5},\delta_{1,4}\}$	$\{\delta_{4,2},\delta_{4,1}\}\to\{\delta_{2,6},\delta_{2,5}\}$	
$\{\delta_{4,2},\delta_{5,2}\}\to\{\delta_{6,4},\delta_{1,4}\}$	$\{\delta_{4,2},\delta_{5,2}\}\to\{\delta_{1,5},\delta_{2,5}\}$	$\{\delta_{4,2},\delta_{5,2}\}\to\{\delta_{2,6},\delta_{3,6}\}$	
$\{\delta_{3,1},\delta_{4,1}\}\to\{\delta_{5,3},\delta_{6,3}\}$	$\{\delta_{3,1},\delta_{4,1}\}\to\{\delta_{6,4},\delta_{1,4}\}$	$\{\delta_{3,1},\delta_{4,1}\}\to\{\delta_{1,5},\delta_{2,5}\}$	$\{\delta_{3,1},\delta_{4,1}\}\to\{\delta_{2,6},\delta_{3,6}\}$

Table 3: Rotating triangulation in $\mathfrak{C}_{n-1}(\mathcal{T}_5)$ to fit in P_6.

Complementing and rotating of a triangulation are illustrated in Figure 2. Figure 2(a) shows a pentagon triangulation $t^1_5=\{\delta_{1,3},\delta_{1,4}\}$. The corresponding complemented triangulation $\left(t^1_5\right)^c$ is presented in Figure 2(b). Figure 2(c) illustrates the fitting of the complemented triangulation $\left(t^1_5\right)^c$ in a hexagon with an ear (v_1,v_2,v_3). Finally, the triangulation $\left(t^1_5\right)^c$ rotated for one position, denoted by $\left(t^1_5\right)^{c,r_1}$, is presented in Figure 2(d).

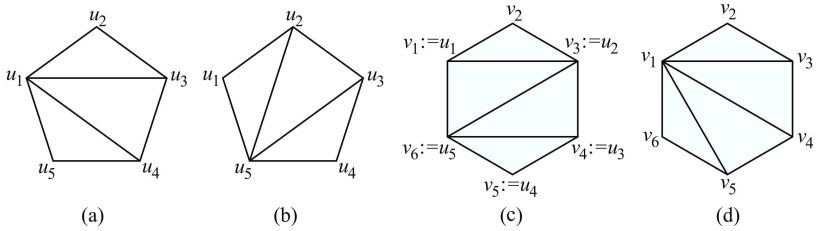


Figure 2: Triangulation complementing and rotating.

Let us explain Figure 2(c) in a few more words. According to (10), a hexagon $P_6=(v_1,v_2,v_3,v_4,v_5,v_6)$ can be considered as the union of the pentagon $P_5=(u_1,u_2,u_3,u_4,u_5)$ and the ear $(v_1:=1,v_2:=2,v_3:=3)$. This means that P_5 can be derived from P_6 according to rules:

$$P_5=\{u_1:=v_1,u_2:=v_3,u_3:=v_4,u_4:=v_5,u_5:=v_6\}.$$

Particulary, the ear tip $v_2:=2$ does not participate in the rotation. So, the diagonal rotation for $l=1$ position is performed in the following way:

$$\delta_{5,2}\to\delta_{\mathbf{1,4}}\Longleftrightarrow\delta_{u_5:=v_6,u_2:=v_3}\to\delta_{u_1:=v_1,u_3:=v_4},$$
$$\delta_{5,3}\to\delta_{\mathbf{1,5}}\Longleftrightarrow\delta_{u_5:=v_6,u_3:=v_4}\to\delta_{u_1:=v_1,u_4:=v_5}.$$

Details about this diagonal rotation are presented in Table 4.

$\delta_{5,2} \to \delta_{\mathbf{1,4}}$	
$u_5 \to 1+(5+1) \bmod 6 = \mathbf{1} = v_1$	$u_2 \to 1+(2+1) \bmod 6 = \mathbf{4} = v_4$
$\delta_{5,3} \to \delta_{\mathbf{1,5}}$	
$u_5 \to 1+(5+1) \bmod 6 = \mathbf{1} = v_1$	$u_3 \to 1+(3+1) \bmod 6 = \mathbf{5} = v_5$

Table 4: Rotating diagonals according to equation (9) for $l = 1$.

Having in mind (10), we perform the rotation of the diagonals as they are the diagonals of P_{n-1}, but we place them in P_n with a specified ear, taking node marks as they are in P_n. As a consequence, the rotation is performed by simple vertices shifting with the same computational cost which does not depend on the number of shifts l.

The subsequent Step 3(c) is the unification of ℓ_6 and Table 3, and it is illustrated in Table 5.

$\delta_{1,3} \circ \Upsilon^1_{n-1}$	$\delta_{2,4} \circ \Upsilon^2_{n-1}$	$\delta_{3,5}\Upsilon^3_{n-1}$	$\delta_{4,6} \circ \Upsilon^4_{n-1}$
$\{\delta_{1,3}, \delta_{1,5}, \delta_{1,4}\}$	$\{\delta_{2,4}, \delta_{2,6}, \delta_{2,5}\}$		
$\{\delta_{1,3}, \delta_{1,5}, \delta_{3,5}\}$	$\{\delta_{2,4}, \delta_{2,6}, \delta_{4,6}\}$		
$\{\delta_{1,3}, \delta_{6,4}, \delta_{6,3}\}$	$\{\delta_{2,4}, \delta_{1,5}, \delta_{1,4}\}$	$\{\delta_{3,5}, \delta_{2,6}, \delta_{2,5}\}$	
$\{\delta_{1,3}, \delta_{6,4}, \delta_{1,4}\}$	$\{\delta_{2,4}, \delta_{1,5}, \delta_{2,5}\}$	$\{\delta_{3,5}, \delta_{2,6}, \delta_{3,6}\}$	
$\{\delta_{1,3}, \delta_{5,3}, \delta_{6,3}\}$	$\{\delta_{2,4}, \delta_{6,4}, \delta_{1,4}\}$	$\{\delta_{3,5}, \delta_{1,5}, \delta_{2,5}\}$	$\{\delta_{4,6}, \delta_{2,6}, \delta_{3,6}\}$

Table 5: Updating blocks for hexagon.

As a result, the triangulations of $\mathcal{T}_6$ are defined as

$$\begin{aligned}\mathcal{T}_6 &= \left\{\delta_{1,3} \circ \Upsilon^1_{n-1}, \delta_{2,4} \circ \Upsilon^2_{n-1}, \delta_{3,5}\Upsilon^3_{n-1}, \delta_{4,6} \circ \Upsilon^4_{n-1}\right\} \\ &= \{\{\delta_{1,3}, \delta_{1,5}, \delta_{1,4}\}, \{\delta_{1,3}, \delta_{1,5}, \delta_{3,5}\}, \{\delta_{1,3}, \delta_{6,4}, \delta_{6,3}\}, \{\delta_{1,3}, \delta_{6,4}, \delta_{1,4}\}, \{\delta_{1,3}, \delta_{5,3}, \delta_{6,3}\}, \\ &\quad \{\delta_{2,4}, \delta_{2,6}, \delta_{2,5}\}, \{\delta_{2,4}, \delta_{2,6}, \delta_{4,6}\}, \{\delta_{2,4}, \delta_{1,5}, \delta_{1,4}\}, \{\delta_{2,4}, \delta_{1,5}, \delta_{2,5}\}, \{\delta_{2,4}, \delta_{6,4}, \delta_{1,4}\} \\ &\quad \{\delta_{3,5}, \delta_{2,6}, \delta_{2,5}\}, \{\delta_{3,5}, \delta_{2,6}, \delta_{3,6}\}, \{\delta_{3,5}, \delta_{1,5}, \delta_{2,5}\} \\ &\quad \{\delta_{4,6}, \delta_{2,6}, \delta_{3,6}\}\}.\end{aligned}$$

Example 2. The key part of the OTM method is a low-cost duplicate triangulations exclusion according to the values of the Catalan triangle. For a more detailed explanation, let us illustrate (in Table 6) details about generating triangulations of P_6 using both complementing and rotating, as stated in Algorithm 2. Repeated triangulations are stricken through.

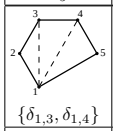
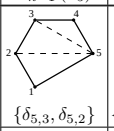
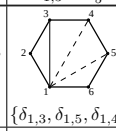
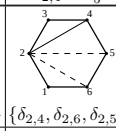
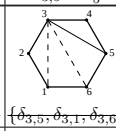
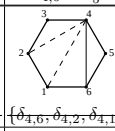
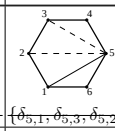
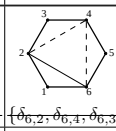
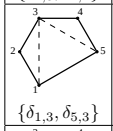
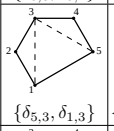
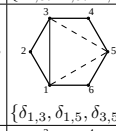
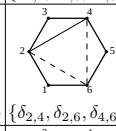
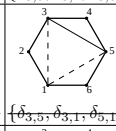
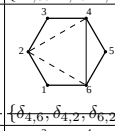
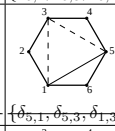
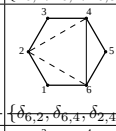
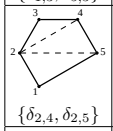
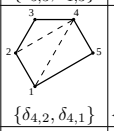
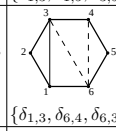
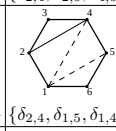
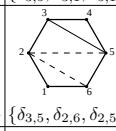
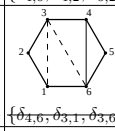
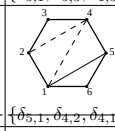
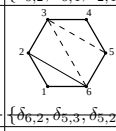
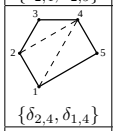
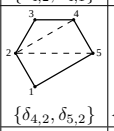
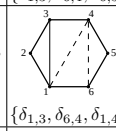
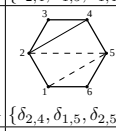
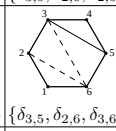
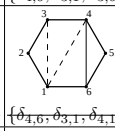
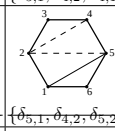
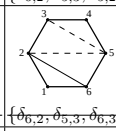
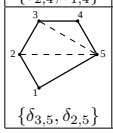
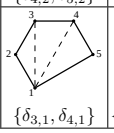
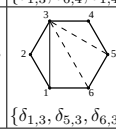
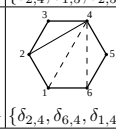
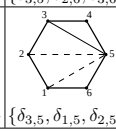
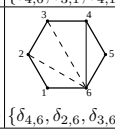
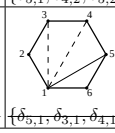
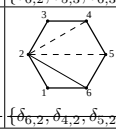
		Generating $\ell_6 = (\delta_{1,3}, \delta_{2,4}, \delta_{3,5}, \delta_{4,6})$					
$\mathcal{T}_5$	$\mathfrak{C}_{n-1}(\mathcal{T}_5)$	$\delta_{1,3} \circ \Upsilon_5^1$	$\delta_{2,4} \circ \Upsilon_5^2$	$\delta_{3,5} \circ \Upsilon_5^3$	$\delta_{4,6} \circ \Upsilon_5^4$		
							
$\{\delta_{1,3}, \delta_{1,4}\}$	$\{\delta_{5,3}, \delta_{5,2}\}$	$\{\delta_{1,3}, \delta_{1,5}, \delta_{1,4}\}$	$\{\delta_{2,4}, \delta_{2,6}, \delta_{2,5}\}$	~~$\{\delta_{3,5}, \delta_{3,1}, \delta_{3,6}\}$~~	~~$\{\delta_{4,6}, \delta_{4,2}, \delta_{4,1}\}$~~	~~$\{\delta_{5,1}, \delta_{5,3}, \delta_{5,2}\}$~~	~~$\{\delta_{6,2}, \delta_{6,4}, \delta_{6,3}\}$~~
							
$\{\delta_{1,3}, \delta_{5,3}\}$	$\{\delta_{5,3}, \delta_{1,3}\}$	$\{\delta_{1,3}, \delta_{1,5}, \delta_{3,5}\}$	$\{\delta_{2,4}, \delta_{2,6}, \delta_{4,6}\}$	~~$\{\delta_{3,5}, \delta_{3,1}, \delta_{5,1}\}$~~	~~$\{\delta_{4,6}, \delta_{4,2}, \delta_{6,2}\}$~~	~~$\{\delta_{5,1}, \delta_{5,3}, \delta_{1,3}\}$~~	~~$\{\delta_{6,2}, \delta_{6,4}, \delta_{2,4}\}$~~
							
$\{\delta_{2,4}, \delta_{2,5}\}$	$\{\delta_{4,2}, \delta_{4,1}\}$	$\{\delta_{1,3}, \delta_{6,4}, \delta_{6,3}\}$	$\{\delta_{2,4}, \delta_{1,5}, \delta_{1,4}\}$	$\{\delta_{3,5}, \delta_{2,6}, \delta_{2,5}\}$	~~$\{\delta_{4,6}, \delta_{3,1}, \delta_{3,6}\}$~~	~~$\{\delta_{5,1}, \delta_{4,2}, \delta_{4,1}\}$~~	~~$\{\delta_{6,2}, \delta_{5,3}, \delta_{5,2}\}$~~
							
$\{\delta_{2,4}, \delta_{1,4}\}$	$\{\delta_{4,2}, \delta_{5,2}\}$	$\{\delta_{1,3}, \delta_{6,4}, \delta_{1,4}\}$	$\{\delta_{2,4}, \delta_{1,5}, \delta_{2,5}\}$	$\{\delta_{3,5}, \delta_{2,6}, \delta_{3,6}\}$	~~$\{\delta_{4,6}, \delta_{3,1}, \delta_{4,1}\}$~~	~~$\{\delta_{5,1}, \delta_{4,2}, \delta_{5,2}\}$~~	~~$\{\delta_{6,2}, \delta_{5,3}, \delta_{6,3}\}$~~
							
$\{\delta_{3,5}, \delta_{2,5}\}$	$\{\delta_{3,1}, \delta_{4,1}\}$	$\{\delta_{1,3}, \delta_{5,3}, \delta_{6,3}\}$	$\{\delta_{2,4}, \delta_{6,4}, \delta_{1,4}\}$	$\{\delta_{3,5}, \delta_{1,5}, \delta_{2,5}\}$	$\{\delta_{4,6}, \delta_{2,6}, \delta_{3,6}\}$	~~$\{\delta_{5,1}, \delta_{3,1}, \delta_{4,1}\}$~~	~~$\{\delta_{6,2}, \delta_{4,2}, \delta_{5,2}\}$~~

Table 6: Generating triangulation of $\mathcal{T}_6$ by OTM method.

It is important to mention that duplicate triangulations in Table 6 appear sequentially and emerge on the top. In accordance with this observation, the OTM method discovers unique triangulations in exact order as values of the Catalan triangle, which represents background for the elimination of duplicates.

Remark 2. *As stated before, the algorithms similar to Algorithm 2, which are based on the usage of the triangulations included in $\mathcal{T}_{n-1}$ in order to generate the triangulations of $\mathcal{T}_n$, make a lot of duplicated triangulations. Their elimination is tedious and time costly. One distinctive appearance of this difficulty appears in the Block method for convex polygon triangulation, which was introduced in* [**19**]. *The recursion with memoization was used in* [**19**] *to avoid recomputations of previously processed triangulations, caused by overlapping of initiated subproblems in the Block method. A suitable file storage, which is aimed to store the previously derived $(n-1)$-gon triangulations and use them later in generating the n-gon triangulations, was used in* [**13**].

The crucial observation in the proposed method is the fact that it requires only a subset Υ_{n-1}^k of the form (11) *in each column. In this way, we get proper n-gon triangulations with far lower computational effort.*

In Theorem 1 we show correctness of Algorithm 2.

Theorem 1. *The set of triangulations $\mathcal{T}_n = \left\{\delta_{k,k+2} \circ \Upsilon_{n-1}^k, k = 1, \ldots, n-2\right\}$ generated by Algorithm 2 is valid.*

Proof. Obviously, the cardinality of $\mathcal{T}_n$ is equal to C_{n-2}, since the number of elements in $\mathcal{T}_n$ is equal to the sum of elements of the corresponding row $(n-3)$ of the Catalan triangle. Later, we have to verify that $\mathcal{T}_n$ does not contain duplicates. The proof proceeds with the mathematical induction. Indeed, assume that the set of triangulations $\mathcal{T}_{n-1}$ is valid, i.e. without duplicates. This implies that the set ${\mathcal{T}_{n-1}}^{c,r_k}$, generated in Step 3(b) of Algorithm 2, is duplicate free. Further, the set Υ_{n-1}^k is also without duplicates and finally the set $\delta_{k,k+2} \circ \Upsilon_{n-1}^k$ is without duplicates for each $k = 1, \ldots, n-2$. This implies that all elements in the arbitrary kth column are without duplicates.

Moreover, let us consider triangulations included in the kth column, $k \in \{1, \ldots, n-2\}$. These triangulations are of the form $\delta_{k,k+2} \circ \Upsilon_{n-1}^k$ and start with the diagonal $\delta_{k,k+2}$, which is unique for this column. The arbitrary different column j, $j \neq k$, includes triangulations of the form $\delta_{j,j+2} \circ \Upsilon_{n-1}^j$, which contain the first diagonal equal to $\delta_{j,j+2}, \neq \delta_{k,k+2}$. In this way, the triangulations $\delta_{k,k+2} \circ \Upsilon_{n-1}^k$ and $\delta_{j,j+2} \circ \Upsilon_{n-1}^j$ are surely different for each $j \neq k$. So, elements in the jth and kth column are mutually different. According to the previous statement, all triangulations $\delta_{k,k+2} \circ \Upsilon_{n-1}^k$ included in an arbitrary kth column are without duplicates, so the proof is completed. □

3. ALGORITHM COMPLEXITY

Let us discuss the complexity of Algorithm 2. In other words, we want to determine how many operations are needed in order to produce one triangulation of P_n in Algorithm 2.

The operation of the diagonal complementing requires performing $2(n-4) = 2n-8$ subtractions. Indeed, each triangulation from $\mathcal{T}_{n-1}$ is defined by $n-4$ diagonals and we need two subtractions to complement every diagonal in the triangulation, as defined in (6).

Further, for the rotation of a diagonal, we should shift its vertex marks for k positions, $k \in \{1, \ldots, n-2\}$ in the clockwise direction, as it is defined in (9). So, it makes always two additions. But, the number of diagonals in a triangulation is again $n-4$ and the total number of additions required to perform the rotation is $2n-8$.

Now, it is necessary to perform one concatenation to produce an element of the set (12). Altogether, it makes $4n-15$ operations. We conclude that Algorithm 2 exhibits the linear complexity.

Of course, the generation of the whole set $\mathcal{T}_n$ would be more complex because of the non-linearity of Catalan numbers. As the production of each triangulation has complexity $\mathcal{O}(n)$, the total number of operations needed for $\mathcal{T}_n$ generation is equal to

$N_{OTM} = (2n-8)C_{n-3} + (2n-8)C_{n-2} + C_{n-2} = (2n-8)C_{n-3} + (2n-7)C_{n-2}.$

This number can be compared to the number of operations needed for the Hurtado method:

$$N_H = (2n-5)C_{n-3} + (2n-3)C_{n-2}$$

It is easy to calculate that

$$N_H - N_{OTM} = 3C_{n-3} + 4C_{n-2} \geq 0, \quad n > 4.$$

4. COMPARATIVE ANALYSIS AND EXPERIMENTAL RESULTS

For the implementation of Block method, Hurtado-Noy and OTM method we have used NetBeans IDE environment for Java.

The execution times for all three algorithms are presented in Table 7. The table column "Speedup" is presented in two form, comparing execution time of Block method vs OTM method and Hurtado-Noy method vs OTM method.

The testing is performed in NetBeans testing module "Profile Main Project / CPU Analyze Performance" in configuration*: *CPU - Inter(R) Core(TM) i5-4210U CPU @ 1.70GHz 2.40GHz, RAM memory 8GB, Graphic card: NVIDIA GeForce 820M.*

n	Number of triangulations	Block method	H.N. method	OTM method	Speedup Block vs OTM	Speedup H.N. vs OTM
5	5	0.000	0.001	0.000	-	-
6	14	0.001	0.002	0.001	1.00	2.00
7	42	0.001	0.002	0.001	1.00	2.00
8	132	0.001	0.002	0.001	1.00	2.00
9	429	0.002	0.003	0.002	1.03	1.50
10	1,430	0.004	0.006	0.003	1.33	2.00
11	4,862	0.010	0.014	0.008	1.25	1.75
12	16,796	0.015	0.033	0.013	1.15	2.54
13	58,786	0.047	0.055	0.038	1.24	1.45
14	208,012	0.140	0.143	0.114	1.23	1.25
15	742,900	0.396	0.512	0.313	1.27	1.64
16	2,674,440	1.761	2.536	1.293	1.36	1.96
17	9,694,845	8.443	10.098	6.728	1.25	1.50
18	35,357,670	106.782	151.273	49.688	2.15	3.04

Table 7: The execution times (in seconds): Block method [**19**] vs. OTM method & Hurtado-Noy method vs. OTM method.

The results arranged in Table 7 confirm the efficacy of the OTM method in generating triangulations and eliminating duplicates. More precisely, the OTM method is fastest compared to both Block method and Hurtado-Noy method.

5. CONCLUSION

Convex polygon triangulation algorithms that are based on cutting out one triangle and triangulating the remaining polygon have a flaw which is reflected in producing more and more duplicates with the increase of the polygon size. One approach tries to use database in order to prevent duplicate triangulations recording (see [**13**]). Regardless of the fact that the presented OTM can be, in a certain way, placed in such a group of algorithms, it has a unique feature which is a non-costly approach in eliminating duplicate triangulations. The approach is simply based on the values arranged in the Catalan triangle. This achievement gives a significant speedup compared to the performance of Block method and Hurtado-Noy method.

Acknowledgements. Predrag Stanimirović and Predrag Krtolica gratefully acknowledge support from the Research Project 174013 of the Serbian Ministry of Science.

REFERENCES

1. A. Asinowski, C. Krattenthaler, T. Mansour: *Counting triangulations of balanced subdivisions of convex polygons*. Electronic Notes in Discrete Mathematics **54**(2016), 73–78.
2. A. Asinowski, C. Krattenthaler, T. Mansour: *Counting triangulations of some classes of subdivided convex polygons*. European Journal of Combinatorics **62**(2017), 92–114.
3. H. ElGindy, H. Everett, G. Toussaint: *Slicing an ear using prune-and-search*, Pattern Recognition Letters **14**(1993), 719–722.
4. D.F. Bailey: *Counting Arrangements of 1's and -1's*. Mathematics Magazine, **69** (1996), 128–131.
5. F. Hurtado and M. Noy: *Ears of triangulations and Catalan numbers*. Discrete Mathematics, **149**(1996), 319–324.
6. F. Hurtado and M. Noy: *Graph of triangulations of a convex polygon and tree of triangulations*. Computational Geometry, **13**(1999), 179–188.
7. F. Hurtado, M. Noy: *Counting triangulations of almost-convex polygons*. Ars Combinatoria **45**(1997), 169–179.
8. I. Kanj, E. Sedgwick, G. Xia: *Computing the flip distance between triangulations*. Discrete Compu. Geom. **58**(2017), 313–344.
9. J. M. Keil, T. S. Vassilev: *Algorithms for optimal area triangulations of a convex polygon*. Computational Geometry **35**(2006), 173–187.
10. G.T. Klincsek: *Minimal triangulations of polygonal domains*. Annals of Discrete Mathematics **9**(1980), 121–123.
11. M. Korman, S,. Langerman, W. Mulzer, A. Pilz, M. Saumell, B. Vogtenhuber: *The dual diameter of triangulations*. Computational Geometry: Theory and Applications **68**(2018), 243–252.

12. T. Koshy: *Catalan Numbers with Applications*. Oxford University Press, London, UK (2009) .

13. P. Krtolica, P. Stanimirović, M. Tasić and S. Pepić: *Triangulation of Convex Polygon with Storage Support*. Facta Universitatis, Series: Mathematics and Informatics, **29**(2)(2014), 189–208.

14. S. Negami: *Diagonal flips in triangulations of surfaces*. Discrete Mathematics **135**(1994), 225–232.

15. J. O'Rourke: *Art gallery theorems and algorithms*. Oxford Univ. Press, Oxford (1987).

16. M. Saračević, P.S. Stanimirović, S. Mašović and E. Biševac: *Implementation of the convex polygon triangulation algorithm*. Facta Universitatis, Series: Mathematics and Informatics, **27**(2012), 213–228.

17. M. Saračević, P.S. Stanimirović, P. Krtolica and S. Mašović: *Construction and notation of convex polygon triangulation based on ballot problem*. Romanian Journal of Information Science and Technology, **17**(2014), 237–251.

18. L.W. Shapiro: *A Catalan triangle*. Discrete Mathematics, **14**(1976), 83-90.

19. P.S. Stanimirović, P. Krtolica, M. Saračević and S. Mašović: *Block Method for Convex Polygon Triangulation*. Romanian Journal of Information Science and Technology, **15**(4)(2012), 344–354.

20. P.S. Stanimirović, P. Krtolica, M. Saračević and S. Mašović: *Decomposition of Catalan numbers and convex polygon triangulations*. International Journal of Computer Mathematics, **91**(2014), 1315–1328.

21. J. Yu, Z.-Li Tang: *Generating strictly binary trees at random based on convex polygon triangulations*. International Journal Of Computer Mathematics **93**(2016), 445–452.

Sead Mašovic
Faculty of Science and Mathematics,
University of Niš
4 July 93, 36300 Novi Pazar
Serbia
E-mail: *sead.masovic@pmf.edu.rs*

Islam Elshaarawy
Faculty of Engineering (at Shoubra),
Benha University, Cairo
Egypt
E-mail: *islam.elshaarawy@feng.bu.edu.eg*

Predrag Stanimirovic
Faculty of Science and Mathematics,
University of Niš
Niš
Serbia
E-mail: *pecko@pmf.ni.ac.rs*

Predrag Krtolica
Faculty of Science and Mathematics,
University of Niš
Serbia
E-mail: *krca@pmf.ni.ac.rs*

(Received 29.08.2017)
(Revised 04.07.2018)