VOL.2 NO.2September 2013 - ISSN 2233 – 1859

UOIuBIH
ORSinBIH
Operations Research Society in Bosnia and Herzegovina

Southeast Europe Journal of Soft Computing

Available online: www.scjournal.com.ba

JAVA Implementation for Triangulation of Convex Polygon Based on Lukasiewicz's Algorithm and Binary Trees

Muzafer H. Saračević[1], Sead H. Mašović[2], Danijela G. Milošević[3]

[1]Department of Technical Science, University of Novi Pazar, Dimitrija Tucovića, Serbia, muzafers@uninp.edu.rs
[2] Faculty of Science and Mathematics, University of Nis, Višegradska 33, Niš, Serbia, sead.masovic@pmf.edu.rs
[3]Faculty of Technical Science, University of Kragujevac, Svetog Save 65, Čačak, Serbia, danijela@tfc.kg.ac.rs

Article Info

Article history:
Received 17 Sep.2013
Received in revised form 17 Oct 2013

Keywords:
Triangulation of Polygons,
Lukasiewicz's algorithm,
Binary trees, Java programming.

Abstract

Triangulation of the polygon is one of the fundamental algorithms in computational geometry. This paper describes one method of recording triangulations of a convex polygon. The notation that is obtained is expressed in the form of binary records. A presented method of storing triangulations is based on Lukasiewicz's algorithm and binary trees. Particularly, we present the implementation for creating binary records for triangulation polygons. As evaluation of presented method we provide Java implementation for binary notation for triangulation polygons. Java application displays a correlation between the binary notation and the graphical representation.

1. INTRODUCTION AND PRELIMINARIES

The triangulation of polygons is very important notion in the computer graphics. A triangulation of a simple polygon assumes the decomposition of its interior into triangles, without mutually intersections of internal diagonals. The polygon triangulation is an important aspect of computational geometry. Mainly, triangulation allows a three-dimensional view of objects from the set of points.

It is necessary to provide an efficient method of recording triangulations, which will be used for notation of polygon triangulation, as a substitute for storage of the nodes and distribution of internal polygon diagonals. Efficiency is reflected in the speed of recording triangulation in Java buffer (cache) which is used to store temporary results. In this way better results can be achieved when it comes to the generation speed and drawing triangulations.

The procedure of recording convex polygon triangulations will be connected with the binary trees (Koshy, 2009). The binary tree is a familiar concept in the field of computer sciences and represents a structure intended for data storage.

Its memory units are organized according to the principle of the pyramid. More specifically, each memory unit (node) of the binary tree may point out to more than two elements (its descendants). It is important to note that the tree has only one element towards whom none other element doesn't point out (root). From this element any other element of the tree can be reached.

It is important to note that the binary tree can represent Catalan numbers (see Figure 2).

More specifically, based on these numbers, is performed the process of calculating the number of all triangulations of polygon (Koshy, 2009). The number of triangulations for *n*-gonis equal to

$$C_{n-2} = \frac{2(n-2)}{(n-1)!(n-2)!}, \qquad (1)$$

where n is the number of vertices of a convex polygon (Chazelle, 1991).

In order to obtain for each triangulation of convex polygon the corresponding record, we are going to apply the *Lukasiewicz's* algorithm (Koshy, 2009).

The procedure for touring binary trees by this algorithm is implemented that the nodes are marked with the **0** and the branches with **1**. The obtained record corresponds to the so called *Necklaces notation* which is presented in the form of binary record of movement through the tree (Figure 1).

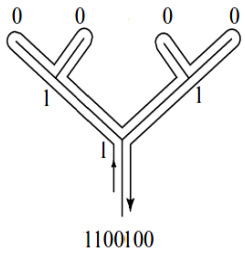


Figure 1: Way of recording a binary tree using *Lukasiewicz's* algorithm

In this way we get the corresponding binary record (notation), which is unique and corresponds to exactly one triangulation for which this binary tree is applicable.

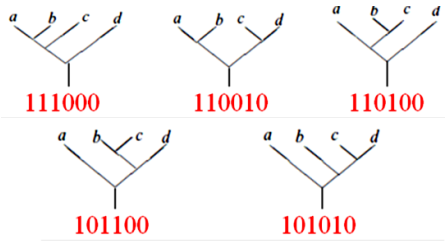


Figure 2: Examples of different binary trees for C_3

Figure 3 presents the procedure how on the basis of a binary tree (more precisely, its binary record), a convex polygon triangulation is formed.

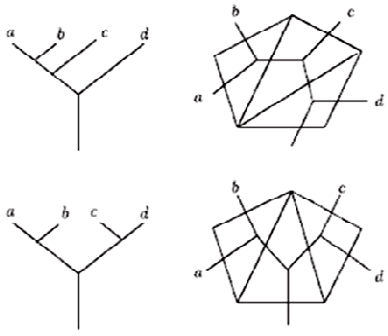


Figure 3: Construction process of triangulation based on a binary tree

2. RELATED WORKS

In our paper (Stanimirovic, 2012), we present the *block method* for convex polygon triangulation. The method is based on the usage of the previously generated triangulations for polygon with smaller number of vertices. In the beginning, we establish some relations between the triangulations of the polygons having the consecutive number of vertices. Using these relations, we decompose the triangulation problem into sub-problems which are themselves smaller instances of the starting problem. The recursion with memoization is used to avoid repeating the calculation of results for previously processed inputs. The corresponding algorithm is developed and implemented.

In order to reduce the memory space requirements, in our paper (Masovic, 2013) we propose a shortened form (so called *AN notation*) for the storage of generated triangulations which is also based on a binary records (bit-strings) and records in the form of *balanced parentheses*. This shortened form represents a unique key for each graph or any combination of triangulations for convex polygons.

In paper (Hurtado, 1999) is described graph of triangulations of a convex polygon and tree of triangulations, while the authors of the paper (Devroye, 1999) give properties of random triangulations and trees. Author of Master's thesis (Fang, 2008) presents the diagonal-flip of convex polygon triangulations and the rotation of binary trees.

3. JAVA IMPLEMENTATION OF LUKASIEWICZ'S ALGORITHM FOR TRIANGULATION OF POLYGON

Java is a programming language with exceptional possibilities when it comes to working with graphics and also when it comes to speed of execution. In order for certain algorithms to work efficiently, they require appropriate data structures for the representation of points, set of points, segments and other geometric objects.

Implementation of notation for convex polygon triangulation using *Lukasiewicz's* algorithm in *Java* consists of following classes: `Triangulation`, `Node`, `LeafNode`, `Point` `GenerateTriangulations`,`PostScriptWriter`.

At runtime of the algorithm we always have two binary records**1100** and **1010**which correspond to binary trees for two triangulations in quadrangle(see the hierarchy of the triangulation in Hurtado, 1999).

Our procedure is based on the complementarities of the two binary records. This idea of taking triangulation from smaller to larger levels resulted from the hierarchy that is presented in the mentioned paper (Hurtado, 1999). In our paper (Saracevic, 2012), we give implementation of the algorithm for generating and displaying triangulations of the convex polygon, given by *Hurtado-Noy*, in three object-oriented programming languages (*Java, Python, C++*).

`Class Node` and `LeafNode` are responsible for working with binary trees and the current formation and storage of triangulations based on them. In the class `Node`are

defined two starting records (corresponds to two triangulations in quadrangle) which are later complemented:

```
privateint base1=1100, base2=1010;
```

From the aspect of writing in the cache, the binary records are expanded, where 1 and 0 are added (for example, from 1100 to 111000 or 1010 in 110100, etc.). The method of class Node for transfer of binary records is `copy()`:

```
publicNode copy(){
return new
Node(this.left.copy(),this.right.copy());}
```

`LeafNode` class has two methods that are responsible for the formation of binary records:

```
publicint leaves() {return 1;}
publicintleftBranch(){return 0;}
```

The line of code from `GenerateTriangulations`class, which corresponds to the transfer of binary records and the formation of descendants are:

```
Node t = (Node)genTriang.get(n).get(j);
Node s = new Node(new LeafNode(),
t.copy());genTriangSum.add(s);
```

In the class`GenerateTriangulations`, the method `generate()` is defined, which is responsible for the generation of triangulations based on binary records. Triangulation class is responsible for drawing a convex polygon triangulation and the class `PostScriptWriter` for the final review and generating an output file (in *ps* or *pdf* format). In the main class, the process of constructing graphic triangulation is done so that over the instance of class `GenerateTriangulations`, anbinary records is created which is associated with the class Node and thus we get the binary records. `DrawAll` method is called in, of the `Triangulation` class, which on the basis of binary records generates triangulations. A part of the code from the class `GenerateTriangulations` is listed below is responsible for drawing convex polygon triangulations based on the obtained records:

```
GenerateTriangulations app = new
GenerateTriangulations();
Vector<Node>genTriang=app.generate(n-2);
PostScriptWriter writer = new
PostScriptWriter(out);
Triangulation instanceTriangulation = new
Triangulation(p,writer);
instanceTriangulation.DrawAll(genTriang);
```

4. REGULARITY IN THE STORAGE OF BINARY RECORDS FOR TRIANGULATIONS

Now we will analyze the obtained binary records. According to hierarchy from paper (Hurtado, 1999) concatenation of triangulation for*n*-gonis always performed from (*n*-1) level in the *n*-level. Based on these preferences we will investigate whether for their binary records can apply transfer rules, or whether in the next block of triangulation $\mathbf{T_n}$ (sets of binary records for triangulations) are some binary records from $\mathbf{T_{n-1}}$.

Example 1. By testing our *Java* application that records every triangulation formed on the basis of the above described procedure, we get the following matrix with binary records for all triangulations for the pentagon(T_5) and the hexagon(T_6).

$$C_3 = T_5 = \begin{bmatrix} 1&1&1&0&0&0 \\ 1&1&0&1&0&0 \\ 1&0&1&1&0&0 \\ 1&0&1&0&1&0 \\ 1&1&0&0&1&0 \end{bmatrix} \qquad C_4 = T_6 = \begin{bmatrix} 1&1&1&1&0&0&0&0 \\ 1&1&1&0&1&0&0&0 \\ 1&1&0&1&1&0&0&0 \\ 1&1&0&1&0&1&0&0 \\ 1&1&1&0&0&1&0&0 \\ 1&0&1&1&1&0&0&0 \\ 1&0&1&1&0&1&0&0 \\ 1&0&1&0&1&1&0&0 \\ 1&1&0&0&1&1&0&0 \\ 1&1&0&0&1&0&1&0 \\ 1&0&1&0&1&0&1&0 \\ 1&0&1&1&0&0&1&0 \\ 1&1&0&1&0&0&1&0 \\ 1&1&1&0&0&0&1&0 \end{bmatrix}$$

If we take one matrix row of binary records for pentagon (eg. **101010**), we see that the binary record from the pentagon completely states in two records for hexagon triangulation. In this case we can see the existence of two derived triangulation records**10{101010}** and **1{101010}0**.

The results from the examples will be organized in a way as presented in Table 1. It was noted that truly there is valid transfer of the two records from $\mathbf{T_{n-1}}$in $\mathbf{T_n}$.

TABLE1.RATIO OF THE BINARY BLOCKS: T_5IN T_6

T_5	T_6			
	$10T_6$		$1T_60$	
1 1 1 0 0 0	1 0	1 1 1 0 0 0	1	1 1 1 0 0 0	0
1 1 0 1 0 0	1 0	1 1 0 1 0 0	1	1 1 0 1 0 0	0
1 0 1 1 0 0	1 0	1 0 1 1 0 0	1	1 0 1 1 0 0	0
1 0 1 0 1 0	1 0	1 0 1 0 1 0	1	1 0 1 0 1 0	0
1 1 0 0 1 0	1 0	1 1 0 0 1 0	1	1 1 0 0 1 0	0

REST
1 1 0 0 1 0 1 0
1 1 0 0 1 1 0 0
1 1 0 1 0 0 1 0
1 1 1 0 0 0 1 0

Based on this testing and analysis of the obtained results, we see that each triangulation fromT_{n-1}occurs in T_n in two forms, namely: **10{$\mathbf{T_{n-1}}$}**and **1{$\mathbf{T_{n-1}}$}0**.

Example 2. Now we will use the obtained records for $T_6$for deriving of binary triangulations in T_7. In the first step, the transfer of two blocks is done in the form: **10{$\mathbf{T_6}$}**and **1{$\mathbf{T_6}$}0**.

Final results for the ratio of the binary blocks T_6 in T_7 are shown in Table 2.

TABLE2.RATIO OF THE BINARY BLOCKS:T_6IN T_7

T_6	T_7	
	$10T_6$	$1T_60$
1 1 1 1 0 0 0 0	1 0 1 1 1 1 0 0 0 0	1 1 1 1 1 0 0 0 0 0
1 1 1 0 1 0 0 0	1 0 1 1 1 0 1 0 0 0	1 1 1 1 0 1 0 0 0 0
1 1 0 1 1 0 0 0	1 0 1 1 0 1 1 0 0 0	1 1 1 0 1 1 0 0 0 0
1 1 0 1 0 1 0 0	1 0 1 1 0 1 0 1 0 0	1 1 1 0 1 0 1 0 0 0
1 1 1 0 0 1 0 0	1 0 1 1 1 0 0 1 0 0	1 1 1 1 0 0 1 0 0 0
1 0 1 1 1 0 0 0	1 0 1 0 1 1 1 0 0 0	1 1 0 1 1 1 0 0 0 0
1 0 1 1 0 1 0 0	1 0 1 0 1 1 0 1 0 0	1 1 0 1 1 0 1 0 0 0
1 0 1 0 1 1 0 0	1 0 1 0 1 0 1 1 0 0	1 1 0 1 0 1 1 0 0 0
1 0 1 0 1 0 1 0	1 0 1 1 0 0 1 1 0 0	1 1 1 0 0 1 1 0 0 0
1 0 1 1 0 0 1 0	1 0 1 0 1 1 0 0 1 0	1 1 1 0 0 1 0 1 0 0
1 1 0 0 1 0 1 0	1 0 1 1 0 1 0 0 1 0	1 1 0 1 0 1 0 1 0 0
1 1 0 0 1 1 0 0	1 0 1 1 1 0 0 0 1 0	1 1 0 1 1 0 0 1 0 0
1 1 0 1 0 0 1 0	1 0 1 1 0 0 1 0 1 0	1 1 1 0 1 0 0 1 0 0
1 1 1 0 0 0 1 0	1 0 1 0 1 0 1 0 1 0	1 1 1 1 0 0 0 1 0 0

REST
1 1 0 0 1 0 1 0 1 0
1 1 0 0 1 0 1 1 0 0
1 1 0 0 1 1 0 0 1 0
1 1 0 0 1 1 0 1 0 0
1 1 0 0 1 1 1 0 0 0
1 1 0 1 0 0 1 0 1 0
1 1 0 1 0 0 1 1 0 0
1 1 0 1 0 1 0 0 1 0
1 1 0 1 1 0 0 0 1 0
1 1 1 0 0 0 1 0 1 0
1 1 1 0 0 0 1 1 0 0
1 1 1 0 0 1 0 0 1 0
1 1 1 0 1 0 0 0 1 0
1 1 1 1 0 0 0 0 1 0

Based on the above tests, in the general case we can define the relation: $\mathbf{T_n = 10\{T_{n-1}\}+1\{T_{n-1}\}0+\{REST\}}$,where is T_n,T_{n-1} and RESTare sets of binary records for triangulations.

These established relationships can provide the separation of complete blocks T_{n-1}on the basis of the received block T_n. More specifically, based on a larger block of T_n can be seen lower T_{n-1} blocks (Table 3).

TABLE3.
NUMBER OF SMALLER BINARY BLOCKS INVOLVED IN BIGGER BINARY BLOCKS

T_{n-1}	T_n				
	12	**11**	**10**	**9**	**8**
12	1	x	x	x	x
11	2	1	x	x	x
10	5	2	1	x	x
9	14	5	2	1	x
8	42	14	5	2	1
7	132	42	14	5	2
6	429	132	42	14	5
5	1430	429	132	42	14
4	4862	1430	429	132	42

Figure 4 shows the separation of T_5 triangulation block from T_7 (left) and extraction of the T_6 triangulation block of T_7 (right).

The tables in the Figure 4 aretriangulations represented by internal diagonals (marked with$\delta_{p,q}$).

i	D1	D2	D3	D4
1	$\delta_{1,3}$	$\delta_{1,4}$	$\delta_{1,5}$	$\delta_{1,6}$
2	$\delta_{1,3}$	$\delta_{3,5}$	$\delta_{1,5}$	$\delta_{1,6}$
3	$\delta_{2,4}$	$\delta_{1,4}$	$\delta_{1,5}$	$\delta_{1,6}$
4	$\delta_{2,4}$	$\delta_{2,5}$	$\delta_{1,5}$	$\delta_{1,6}$
5	$\delta_{2,5}$	$\delta_{3,5}$	$\delta_{1,5}$	$\delta_{1,6}$
6	$\delta_{1,3}$	$\delta_{1,4}$	$\delta_{4,6}$	$\delta_{1,6}$
7	$\delta_{1,3}$	$\delta_{3,5}$	$\delta_{3,6}$	$\delta_{1,6}$
8	$\delta_{2,4}$	$\delta_{1,4}$	$\delta_{4,6}$	$\delta_{1,6}$
9	$\delta_{2,4}$	$\delta_{2,5}$	$\delta_{2,6}$	$\delta_{1,6}$
10	$\delta_{2,5}$	$\delta_{3,5}$	$\delta_{2,6}$	$\delta_{1,6}$
11	$\delta_{2,6}$	$\delta_{3,6}$	$\delta_{4,6}$	$\delta_{1,6}$
12	$\delta_{2,6}$	$\delta_{3,6}$	$\delta_{3,5}$	$\delta_{1,6}$
13	$\delta_{2,6}$	$\delta_{4,6}$	$\delta_{2,4}$	$\delta_{2,6}$
14	$\delta_{3,6}$	$\delta_{4,6}$	$\delta_{1,3}$	$\delta_{4,6}$
15	$\delta_{1,3}$	$\delta_{1,4}$	$\delta_{1,5}$	$\delta_{5,7}$
16	$\delta_{1,3}$	$\delta_{3,5}$	$\delta_{1,5}$	$\delta_{5,7}$
17	$\delta_{2,4}$	$\delta_{1,4}$	$\delta_{1,5}$	$\delta_{5,7}$
18	$\delta_{2,4}$	$\delta_{2,5}$	$\delta_{1,5}$	$\delta_{5,7}$
19	$\delta_{2,5}$	$\delta_{3,5}$	$\delta_{1,5}$	$\delta_{5,7}$
20	$\delta_{1,3}$	$\delta_{1,4}$	$\delta_{4,6}$	$\delta_{5,7}$
21	$\delta_{1,3}$	$\delta_{3,5}$	$\delta_{3,6}$	$\delta_{5,7}$
22	$\delta_{2,4}$	$\delta_{1,4}$	$\delta_{4,6}$	$\delta_{5,7}$
23	$\delta_{2,4}$	$\delta_{2,5}$	$\delta_{2,6}$	$\delta_{5,7}$
24	$\delta_{2,5}$	$\delta_{3,5}$	$\delta_{2,6}$	$\delta_{5,7}$
25	$\delta_{2,6}$	$\delta_{3,6}$	$\delta_{4,6}$	$\delta_{2,7}$
26	$\delta_{2,6}$	$\delta_{3,6}$	$\delta_{3,5}$	$\delta_{2,7}$
27	$\delta_{2,6}$	$\delta_{4,6}$	$\delta_{2,4}$	$\delta_{2,7}$
28	$\delta_{3,6}$	$\delta_{4,6}$	$\delta_{1,3}$	$\delta_{3,7}$
29	$\delta_{2,7}$	$\delta_{3,7}$	$\delta_{4,7}$	$\delta_{5,7}$
30	$\delta_{2,7}$	$\delta_{3,7}$	$\delta_{4,7}$	$\delta_{4,6}$
31	$\delta_{2,7}$	$\delta_{4,7}$	$\delta_{5,7}$	$\delta_{2,4}$
32	$\delta_{2,7}$	$\delta_{3,7}$	$\delta_{5,7}$	$\delta_{3,5}$
33	$\delta_{3,7}$	$\delta_{4,7}$	$\delta_{5,7}$	$\delta_{1,3}$
34	$\delta_{2,7}$	$\delta_{3,7}$	$\delta_{3,6}$	$\delta_{3,5}$
35	$\delta_{2,7}$	$\delta_{3,7}$	$\delta_{3,6}$	$\delta_{4,6}$
36	$\delta_{2,7}$	$\delta_{4,7}$	$\delta_{2,4}$	$\delta_{4,6}$
37	$\delta_{3,7}$	$\delta_{4,7}$	$\delta_{1,3}$	$\delta_{4,6}$
38	$\delta_{4,7}$	$\delta_{5,7}$	$\delta_{1,3}$	$\delta_{1,4}$
39	$\delta_{3,7}$	$\delta_{5,7}$	$\delta_{1,3}$	$\delta_{3,5}$
40	$\delta_{4,7}$	$\delta_{5,7}$	$\delta_{2,4}$	$\delta_{1,4}$
41	$\delta_{2,7}$	$\delta_{5,7}$	$\delta_{2,4}$	$\delta_{2,5}$
42	$\delta_{2,7}$	$\delta_{5,7}$	$\delta_{2,5}$	$\delta_{3,5}$

i	D1	D2	D3	D4
1	$\delta_{1,3}$	$\delta_{1,4}$	$\delta_{1,5}$	$\delta_{1,6}$
2	$\delta_{1,3}$	$\delta_{3,5}$	$\delta_{1,5}$	$\delta_{1,6}$
3	$\delta_{2,4}$	$\delta_{1,4}$	$\delta_{1,5}$	$\delta_{1,6}$
4	$\delta_{2,4}$	$\delta_{2,5}$	$\delta_{1,5}$	$\delta_{1,6}$
5	$\delta_{2,5}$	$\delta_{3,5}$	$\delta_{1,5}$	$\delta_{1,6}$
6	$\delta_{1,3}$	$\delta_{1,4}$	$\delta_{4,6}$	$\delta_{1,6}$
7	$\delta_{1,3}$	$\delta_{3,5}$	$\delta_{3,6}$	$\delta_{1,6}$
8	$\delta_{2,4}$	$\delta_{1,4}$	$\delta_{4,6}$	$\delta_{1,6}$
9	$\delta_{2,4}$	$\delta_{2,5}$	$\delta_{2,6}$	$\delta_{1,6}$
10	$\delta_{2,5}$	$\delta_{3,5}$	$\delta_{2,6}$	$\delta_{1,6}$
11	$\delta_{2,6}$	$\delta_{3,6}$	$\delta_{4,6}$	$\delta_{1,6}$
12	$\delta_{2,6}$	$\delta_{3,6}$	$\delta_{3,5}$	$\delta_{1,6}$
13	$\delta_{2,6}$	$\delta_{4,6}$	$\delta_{2,4}$	$\delta_{2,6}$
14	$\delta_{3,6}$	$\delta_{4,6}$	$\delta_{1,3}$	$\delta_{4,6}$
15	$\delta_{1,3}$	$\delta_{1,4}$	$\delta_{1,5}$	$\delta_{5,7}$
16	$\delta_{1,3}$	$\delta_{3,5}$	$\delta_{1,5}$	$\delta_{5,7}$
17	$\delta_{2,4}$	$\delta_{1,4}$	$\delta_{1,5}$	$\delta_{5,7}$
18	$\delta_{2,4}$	$\delta_{2,5}$	$\delta_{1,5}$	$\delta_{5,7}$
19	$\delta_{2,5}$	$\delta_{3,5}$	$\delta_{1,5}$	$\delta_{5,7}$
20	$\delta_{1,3}$	$\delta_{1,4}$	$\delta_{4,6}$	$\delta_{5,7}$
21	$\delta_{1,3}$	$\delta_{3,5}$	$\delta_{3,6}$	$\delta_{5,7}$
22	$\delta_{2,4}$	$\delta_{1,4}$	$\delta_{4,6}$	$\delta_{5,7}$
23	$\delta_{2,4}$	$\delta_{2,5}$	$\delta_{2,6}$	$\delta_{5,7}$
24	$\delta_{2,5}$	$\delta_{3,5}$	$\delta_{2,6}$	$\delta_{5,7}$
25	$\delta_{2,6}$	$\delta_{3,6}$	$\delta_{4,6}$	$\delta_{2,7}$
26	$\delta_{2,6}$	$\delta_{3,6}$	$\delta_{3,5}$	$\delta_{2,7}$
27	$\delta_{2,6}$	$\delta_{4,6}$	$\delta_{2,4}$	$\delta_{2,7}$
28	$\delta_{3,6}$	$\delta_{4,6}$	$\delta_{1,3}$	$\delta_{3,7}$
29	$\delta_{2,7}$	$\delta_{3,7}$	$\delta_{4,7}$	$\delta_{5,7}$
30	$\delta_{2,7}$	$\delta_{3,7}$	$\delta_{4,7}$	$\delta_{4,6}$
31	$\delta_{2,7}$	$\delta_{4,7}$	$\delta_{5,7}$	$\delta_{2,4}$
32	$\delta_{2,7}$	$\delta_{3,7}$	$\delta_{5,7}$	$\delta_{3,5}$
33	$\delta_{3,7}$	$\delta_{4,7}$	$\delta_{5,7}$	$\delta_{1,3}$
34	$\delta_{2,7}$	$\delta_{3,7}$	$\delta_{3,6}$	$\delta_{3,5}$
35	$\delta_{2,7}$	$\delta_{3,7}$	$\delta_{3,6}$	$\delta_{4,6}$
36	$\delta_{2,7}$	$\delta_{4,7}$	$\delta_{2,4}$	$\delta_{4,6}$
37	$\delta_{3,7}$	$\delta_{4,7}$	$\delta_{1,3}$	$\delta_{4,6}$
38	$\delta_{4,7}$	$\delta_{5,7}$	$\delta_{1,3}$	$\delta_{1,4}$
39	$\delta_{3,7}$	$\delta_{5,7}$	$\delta_{1,3}$	$\delta_{3,5}$
40	$\delta_{4,7}$	$\delta_{5,7}$	$\delta_{2,4}$	$\delta_{1,4}$
41	$\delta_{2,7}$	$\delta_{5,7}$	$\delta_{2,4}$	$\delta_{2,5}$
42	$\delta_{2,7}$	$\delta_{5,7}$	$\delta_{2,5}$	$\delta_{3,5}$

Figure 4:Selection of smaller blocks T_5 and T_6 in T_7

5. TESTING AND EXPERIMENTAL RESULTS

This section presents experimental results of testing *Java* application. When testing Java application for *Hurtado-Noy* algorithm, we find a problem in the speed of generating a triangulation of polygons where the number of triangulations is greater than several tens of thousands of triangulations, see the experimental results of our work (Saracevic, 2012).

Table 4 present the testing results of the methodthat usebinary records for triangulation, values *n = 5,6, ... ,17*, which correspond to the number of combinations from 5 to 9.694.845. The table provides two criteria test:

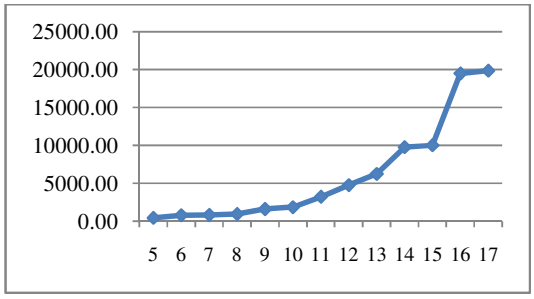
- *A: total time of generating binary records for all triangulations,*
- *B: number of triangulation records in a second.*

TABLE4.RESULTS OF TESTING METHODS WITH BINARY RECORDS

n	Number of triangulations	The method of binary records A	B
5	5	0.012	416.67
6	14	0.018	777.78
7	42	0.051	823.53
8	132	0.142	929.58
9	429	0.270	1588.89
10	1,430	0.781	1833.33
11	4,862	1.520	3198.68
12	16,796	3.541	4744.63
13	58,786	9.463	6214.16
14	208,012	21.33	9752.09
15	742,900	74.27	10002.69
16	2,674,440	137.21	19491.58
17	9,694,845	488.56	19843.71

Testing was done on the computer the following performances: CPU Intel(R) Core(TM)2Duo CPU, T7700, 2.40 GHz, L2 Cache 4 MB, RAM Memory - 2 Gb, Graphic card - NVIDIA GeForce 8600M GS.

We use the *NetBeans Profiler module*. Testing for each value of *n* was done in triplicate and the average values were taken.It can be observed that increasing values of *n* the *Java* application increases the number of generated triangulations per second (Graph 1).



Graph 1: Increasing the number of generated triangulations per second

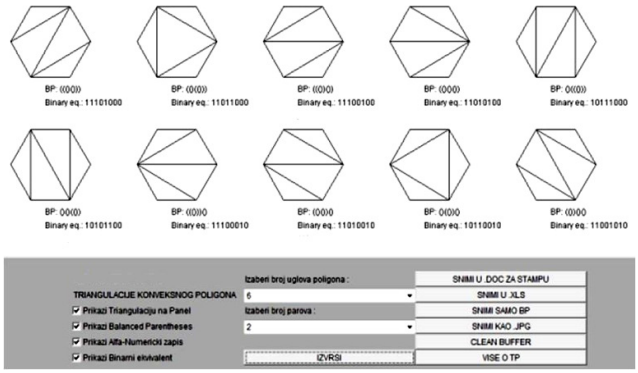
6. CONCLUSION

The proposed application allows efficient representations of the convex polygon triangulations, both in the graphical form as well as in the terms of appropriate records stored in the database. We have listed the efficient method for storing results in the form of *binary* notation. This record form presents a unique key for each graph or any combination of triangulations for convex polygons.

The application in conjunction with a graphical display of triangulations can be used to detect some regularities in generating the triangulations of larger polygons.

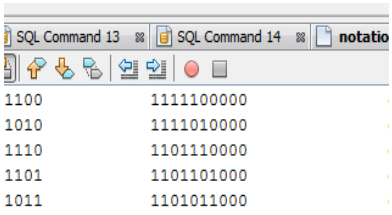
APPENDIX

A) Java application for generating triangulations of a convex polygon(**panel for graphic display).**



Complete *Java* application can be downloaded from: http://muzafers.uninp.edu.rs/triangulacija.html

B) Binary records for triangulation of convex polygons(**content of the output file).**



REFERENCES

Chazelle B. (1991) Triangulation a Simple Polygon in Linear Time. Discrete Computational Geometry 6: 485-524.

Devroye L, Flajolet P, Hurtado F, Noy M, Steiger W (1999) Properties of random triangulations and trees, Discrete & computational geometry 22(1): 105-117.

Fang Z S (2008) The Diagonal-flip of Convex Polygon Triangulations and the Rotation of Binary Trees. Dalian Maritime University, Master's thesis.

Hurtado F, Noy M (1999) Graph of Triangulations of a Convex Polygon and Tree of Triangulations. Computational Geometry 13: 179–188.

Koshy T (2009) Catalan Numbers with Applications, Oxford University Press, New York.

Masovic S, Saracevic M, Stanimirovic PS (2013) Alpha-Numeric notation for one DataStructure in Software Engineering. ActaP.Hungarica: Journal of Applied Sciences (accepted).

Saracevic M, Stanimirovic PS, Masovic S, Bisevac E (2012) Implementation of the convex polygon triangulation algorithm. FactaUniversitatis, series: Mathematics and Informatics27(2): 213–228.

Stanimirovic PS,Krtolica PV, Saracevic M, Masovic S (2012) Block Method for Triangulation Convex Polygon. ROMJIST - Journal of Information Science and Technology15(4): 344–354.