IMPLEMENTATION OF ALGORITHM ON GRAPHS FOR TRIANGULATIONS OF A CONVEX POLYGON

G. Himanshu, K. Avnish, S. Divyanshu

1. Abstract

Calculating the number of triangulations of an n-sided regular convex polygon has been one of those areas of immense interest for some of the greatest mathematicians of the 18th and 19th century, like Euler, Segner, Lame, Liouville and Catalan. Once posted as a thought out of Euler's curiosity in 1751, the problem of triangulation eventually came up as an open challenge in the late 1830s by Liouville, which was finally solved by as late as 1838 by Gabriel Lame. Lame gave a "formula" for computing the number of triangulations of an n-sided polygon. The formula provided for computing the number of triangulations led to the next problem of plotting the graphs of the triangulations to identify similarities (isomorphism) to apply this feature for in various sectors including that of Graph Theory. This led to development of algorithms for these graphs in the recent times. The paper approaches the algorithm by Hurtado and Nay but their exists other methods such as Solving the Triangulation Problem using Cones by Felix Schwenninger. This paper briefly talks about Lame's sight at the problem, and highlights his brilliance by discussing his vision about the problem and his clever deduction of the formula. Later, this paper displays Hurtado and Nay's algorithm to compute triangulations, and the authors' implementation on MATLAB.

2. Introduction

A triangulation of a polygon is polygon, together with some of its diagonals, such that the entire polygon is divided into triangles, where no two diagonals cut each other. The polygon talked about hereafter is a regular convex one.

A triangulation of an n-sided polygon thus can be visualized by cutting a polygon into triangles, conditioned that one is only allowed to cut it along diagonals. Therefore, triangulation of a polygon is the number of all possible distinct individual triangulation of a polygon. For a triangle, it is one. The sequence of triangulations of polygons, starting from the most basic three sided polygon, is 1, 2, 5, 14, 42, 132, 429, 1430, and so on. These numbers are often called Catalan numbers, named after Eugene Charles Catalan for first describing them as the number of ways n+1 factors can be completely parenthesized by n pairs of parenthesis. These Catalan numbers are of much use in combinatorial mathematics, and occur in various counting problems involving recursive objects, like counting Dyck words, counting the number of full binary trees with n+1 leaves, number of stack-sortable permutations of {1 … n}, number of rooted binary trees with n internal vertices and triangulations of an n-gon.

The triangulation problem can be stated as follows:

"Given a convex n-sided polygon, divide it into triangles by drawing non-intersecting diagonals connecting some of the vertices of the polygon."

Euler calculated the number, Pn , of distinct triangulations of a convex n-gon for the first few values of n, and conjectured a formula for Pn based on an empirical study of the ratios Pn+1 /Pn . Lamé was one of the first to provide the details for a combinatorial proof of Euler's conjectured result for Pn+1 /Pn.

Computationally, a triangulation problem becomes a problem of graph theory, where any triangulation is represented in the form of an adjacency matrix. In the further methods in the paper ahead, the authors consider a triangulation as a matrix in MATLAB, and a recursive approach to compute triangulations of n+1 sides from the previously computed of triangulations of n sided polygon, by carrying out required transformations in the corresponding adjacency matrices.

3. MATHEMATICAL ANALYSIS

The concept of deducing the triangulation graphs by the approach taken by F. Hurtado and M. Noy [1999] relies on the data structure known as rooted binary trees where every node is a triangulation of a polygon.

The level of the node represents the triangulations for the corresponding n-polygon defined by the sequence known as catalan numbers which is nothing but the count of the number of possible triangulations for a n sided convex polygon in this case.

The deduction of the Euler's formula for counting triangulations by Lame is provided below as an excerpt from the paper sent as a letter to Liouville on the question.

Let ABCDEF . . . be a convex polygon of n+1 sides, and denote by the symbol P_k the total number of decompositions of a polygon of k sides into triangles. An arbitrary side AB of ABCDEF . . . serves as the base of a triangle, in each of the P_{n+1} decompositions of the polygon, and the triangle will have its vertex at C, or D, or F . . . ; to the triangle CBA there will correspond P_n different decompositions; to DBA another group of decompositions, represented by the product $P_3 P_{n-1}$; to EBA the group $P_n P_{n-2}$; to FBA, $P_5 P_{n-3}$; and so forth, until the triangle ZAB, which will belong to a final group Pn.

Now, all these groups are completely distinct: their sum therefore gives P_{n+1}. Thus one has

$$P_{n+1} = P_n + P_3 P_{n-1} + P_4 P_{n-2} + P_5 P_{n-3} + \cdots + P_{n-3} P_5 + P_{n-2} P_4 + P_{n-1} P_3 + P_n. \qquad (1)$$

Since, each of the partial decompositions of the total group P_n is repeated $2n - 6$ times in $n*(P_3 P_{n-1} + P_4 P_{n-2} + \cdots + P_{n-2} P_4 + P_{n-1} P_3)$, one obtains P_n upon dividing this sum by $2n - 6$.

Therefore one has $P_n = n*(P_3 P_{n-1} + P_4 P_{n-2} + \cdots + P_{n-2} P_4 + P_{n-1} P_3) / (2n - 6)$. (2)

Combining (1) and (2) we have, $P_{n+1} = \frac{4n-6}{n} P_n$.

The final equation with recursive relation comes out as $P_{n+1} = \frac{1}{D2} \binom{2n-2}{n-1} P_2$, where D_2 is some integer .

4. ALGORITHM

As justified by F. Hurtado and M. Noy [1999], triangulations of an n-sided polygon can be viewed in the form of a graph, where each node is a triangulation of a polygon. The tree has a root as a triangle, and each further level of the tree has P_n number of nodes, where P_n is the number of triangulations of n sided polygon. The algorithm demonstrated by Hurtado and Noy computes the triangulations of the polygon, starting from a triangle as the root of the tree, and continuously computes the polygons based on three rules, thus forming a hierarchy of triangulations. It is proved that the three rules are sufficient in generating all possible triangulations of P_n from all given triangulations $P_{n-1.}$

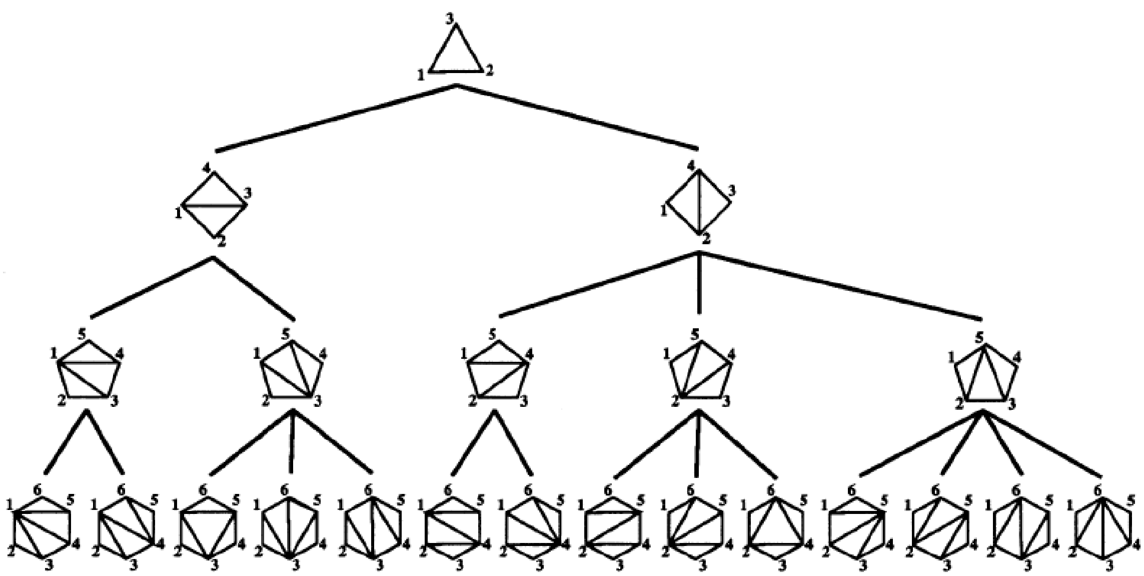
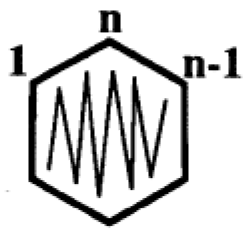


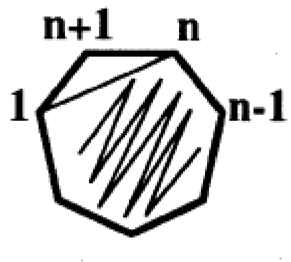
FIG. 3.1 A rooted binary tree with all triangulations for a n-gon.

First rule

Given a triangulation of n-1sides (father), its n sided triangulation counterpart (son) can be computed by adding the n^{th} vertex and joining it to the 1^{st} and n^{th} vertex, shown as an example in figure.



Father



Son

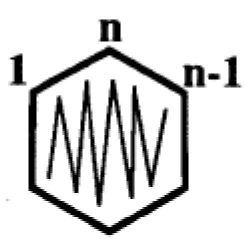
FIG. 3.2

FIG. 3.3

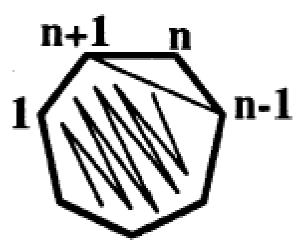
Note: According to first rule, a node will have one and only one son.

Second Rule

According to this rule, a son can be computed from a father by replacing the n^{th} vertex with the $n+1^{th}$ vertex, and the n^{th} vertex is placed with connections only to the $n-1^{th}$ vertex and $n+1^{th}$ vertex. This rule, like the first rule also yeilds in exactly one triangulation. The second rule can be understood from the following figure.



Father



Son

FIG. 3.4

FIG. 3.5

Third Rule

While the first and second rules always yield to a child of a node, the third rule may yield sons. The number of sons is determined by the number of diagonals with one vertex as the n^{th} vertex. The graph is dissected along each of such diagonals, as shown in the figure, and $n+1^{th}$ vertex is accommodated, thus resulting in a new triangulation for each diagonal.

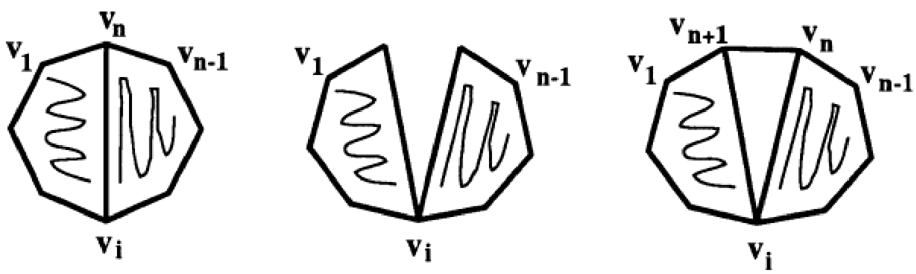


FIG. 3.6

5. MATLAB IMPLEMENTATION AND RESULTS

The algorithm's implementation on MATLAB works as follows: During any iteration, there exists an n x n x P_n dimensional triangulation matrix, which contains the adjacency matrices of all the triangulations of n sided polygon. The script starts with a 3 x 3 x 1 matrix, and computes its children according to the above three rules, to get 4 x 4 x 2 matrix, which serves as the input for next iteration. This algorithm's MATLAB implementation however has memory constraints, which does not allow computations for polygons of more than 15 sides. This can be viewed as a consequence of the large size of the triangulation matrix (15 x 15 x 742900). The following are a few of the computed graphs for a triangulation of a given n-gon.

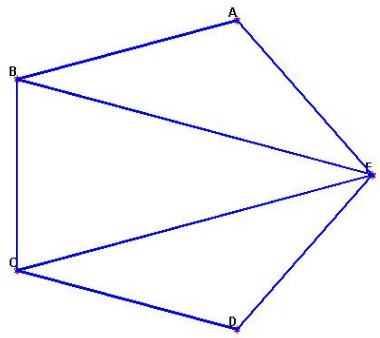


Fig. 5.1 A triangulation for a pentagon

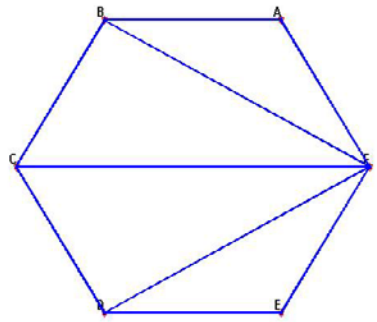


Fig. 5.2 A triangulation for a Hexagon

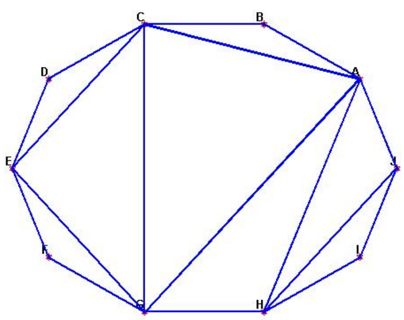


Fig. 5.3 A triangulation for a Decagon

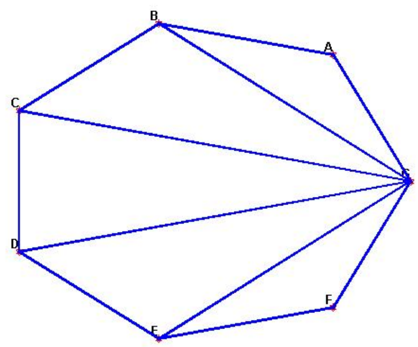


Fig. 5.4 A triangulation for a Heptagon

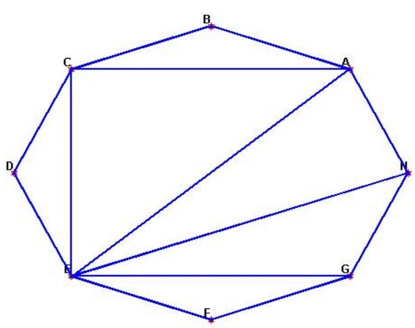


Fig. 5.5 A triangulation for a Octagon

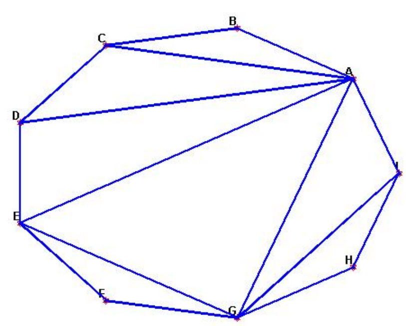


Fig. 5.6 A triangulation for a Nonagon

6. CONCLUSION

The presented results in the above section provides an insight into the matlab implementation of the algorithm proposed by F. Hurtado and M. Noy [1999]. The computational efficiency and other statistics have been observed by Muzafer Saraˇcevi´c, Predrag Stanimirovi´c, Sead Maˇsovi´c, Enver Biˇsevac in their paper for the platforms Java, Python and C++. The matlab platform provides the user with richer visual experience to analyze the triangulations and its combinations. The drawback of the program is the increase in the computational complexity of the program leading to a limit in triangulation possibility to 15 sided polygon. The triangulation matrix comes out to be of the order of (15 x 15 x 742900) for the subsequent polygon leading to memory constraints on the platform.

The platform MATLAB confirms to the validity of the F. Hurtado and M. Noy [1999] algorithm and eventually the Lame counting of Triangulation deduction. The computational limits of matlab and the program restricts us to verifiy the results upto 15 sided polygons and conclude on that basis only.

7. REFERENCES

1. *Jerry Lodder, Gabriel Lam´e's Counting of Triangulations*
2. *Tom Davis, Catalan Numbers, 2006*
3. *F. Hurtado , M. Noy, Graph of triangulations of a convex polygon and tree of Triangulations, 1998*
4. *Muzafer Saraˇcevi´c, Predrag Stanimirovi´c, Sead Maˇsovi´c, Enver Biˇsevac, Implementation of the convex polygon triangulation algorithm, 2012*
5. *Gabriel Lam´e's Counting of Triangulations, An Historical Project*
6. *Felix Schwenninger, Solving the Triangulation Problem using Cones*

GABRIEL LAME'S COUNTING OF TRIANGULATION

Table of Contents

This script uses Lame's modern method to get the number of triangulations of a regular polygon, and the uses an algorithm to draw all those possible triangulations. For simplicity in terms of exactness and execution time and memory constraints, the algorithm is not feeded with a polygon of more than 15 sides.

```
clear all;
clc;
close all;
```

Taking a valid input

```
n = input('Enter the number of sides of the polygon: ');

if (mod(n,1)) ~= 0
    error('Number of sides should be integral!');
elseif (n < 3)
    error('Number of sides should be positive, greater than 3!');
elseif (length(n) ~= 1)
    error('Number of sides should be scalar, recieved vector/matrix');
end
```

Simplicity conditions

```
if n>15
    disp('A smaller number ( < 15 ) recommended! ');
    input('Hit return to restart.');
    lame;
end
```

Generating Adjacency matrix of the regular polygon of n sides

```
adjacency = get_adj_regular(n);
```

Returning the number of triangulations

```
Pn = get_catalan(n-2);
disp('Number of triangulations: ');
disp(Pn);
```

Computing triangulations

```
previous = [0 1 1;1 0 1;1 1 0];

catalans = zeros(1,n-2);
for i = 3:n
    catalans(i-2) = get_catalan(i-2);
end


for count = 4:n
% one cycle of this loop should return triangulations of count sided
% polygon, which should be equal to catalans(count-2)
    new = zeros(count,count,catalans(count-2));
    iterator = 1;
    for cat_count = 1:catalans(count-3)
    % one cycle of this loop should give all children of one of the
    % triangulation of count-1 sided polygon
        temp_adj = previous(:,:,cat_count);

        new(:,:,iterator) = tri_gen_1(temp_adj);
        iterator = iterator + 1;

        [number,mat] = tri_gen_3(temp_adj);
        for i = 1:number
           new(:,:,iterator) = mat(:,:,i);
           iterator = iterator+1;
        end

        new(:,:,iterator) = tri_gen_2(temp_adj);
        iterator = iterator + 1;


    end
    previous = new;
end

triangulations = previous;
```

Displaying the results

```
if n == 3
    graph_plotter(triangulations);
    set(findobj(gcf, 'type','axes'), 'Visible','off');
end
```

```
for i = 1:length(triangulations)
    graph_plotter(triangulations(:,:,i));
    set(findobj(gcf, 'type','axes'), 'Visible','off');
    pause(1);
end
```

Published with MATLAB® R2013a