Paulina STACHNIK[1], Michał ŚLIWA[2]

Opiekun naukowy: Stanisław ZAWIŚLAK[3]

PROBLEM WYZNACZENIA CYKLU EULERA W GRAFIE NIESKIEROWANYM

Streszczenie: W artykule omówiono problem odnalezienia cyklu Eulera w grafie nieskierowanych. Napisano własny program komputerowy, który pozwala stworzyć dowolny graf nieskierowany a następnie zmodyfikować go do postaci grafu eulerowskiego. W programie został wykorzystany algorytm Fleury'ego, którego realizuje można prześledzić iteracje po iteracji.

Słowa kluczowe: graf, graf Eulera , cykl Eulera , algorytm Fleury'ego

THE PROBLEM OF FINDING THE EULER CYCLE AND PATH IN AN UNDIRECTED GRAPH

Summary: The article discusses the problem of finding an Euler cycle in a undirected graph. Own computer program have been written which allows to create any undirected graph and then modify it to the Euler graph. In the program was used Fleury's algorithm whose implementation can be traced iterations after iteration.

Keywords: graph, Euler graph , Euler cycle , Fleury algorithm

1. Introduction

The problem of finding an Euler cycle in a graph was the very first problem that started the graph theory [2, 5] in eighteenth century. Namely, it was presented by a Swiss mathematician, Leonhard Euler in 1736, who wanted to solve the problem of Königsberg bridges. Euler was working for many years at the St. Petersburg

[1] University of Bielsko-Biala, Faculty of Mechanical Engineering and Computer Science, Computer Science; Software Engineering and Information Systems;
[2] University of Bielsko-Biala, Faculty of Mechanical Engineering and Computer Science, Computer Science; Software Engineering and Information Systems, email: michalsliwa96@wp.pl;
[3] Dr hab. inż. prof. ATH, University of Bielsko-Biala, Faculty of Mechanical Engineering and Computer Science, email: szawislak@ath.bielsko.pl;

University and died in this town. The historical remarks on the problem are given in [2].

The – so called – Euler cycle is a cycle in the graph G(V,E) that passes through all the edges of the graph (only once), it begins and ends at the same vertex. We consider that the set V in non-empty, $n = |V| \geq 1$. The necessary conditions for a particular graph to have an Euler cycle are: the graph has to be connected and all its vertices have even degrees. Such a graph is called an Euler graph or eulerian graph.

There is also a term an Euler's path which is a path in the graph passing through all its edges but unlikely to the idea of the Euler cycle, its start and end vertices are different. Such a graph is called semi-eulerian graph.

The Fleury's or Hierholzer algorithms can be used to find the cycle and path of the Euler. The program uses the Fleury algorithm. In the paper, the computer program is described which solves the above formulated tasks.

2. Depth-First Search Algorithm for checking graph connectivity

The described program was written by the authors of the paper. The graph for which the task should be performed was generated in two phases. In the first phase an arbitrary graph was generated for an arbitrary *p* entered by a user. Value of *p* should be within the interval (0,1). The greater is number of generated graph edges the greater is the value *p*. In case if the graph is non-eulerian, the repair procedure was triggered.

However, the very first condition for the graph to be Eulerian is that it is connected. In order to examine the connectivity of the graph, the most frequently used algorithms are Depth-First Search algorithm (DFS) and/or Breadth-First Search (BFS) algorithms. The program uses an Depth-First Search algorithm. The operation of the algorithm is presented below.

At the beginning, the start vertex is selected from which the search begins - it is placed on a stack of visited vertices and marked as vertex X (this is the name of the vertex in which the algorithm is currently located). The algorithm selects a random neighbour of vertex X to which it passes. This vertex is added to a stack of visited vertices and marked as X. Then the algorithm passes to any neighbour of vertex X which is not on the stack of visited vertices - this step is repeated until vertex X does not have neighbours to which the algorithm could be passed. This vertex is added to a stack of visited vertices and marked as X. Then the algorithm passes to any neighbour of vertex X which is not on the visited vertex stack - this step is repeated until vertex X has no neighbours to which to pass. Then from vertex X the algorithm returns to the parent vertex (the one from which the algorithm passed to vertex X) marks it as X and search for its not yet visited neighbors if there is no such algorithm returns again to the parent vertex. The algorithm ends when it returns to the start vertex that has no unvisited neighbours. If the number of vertices placed in the stack of vertices visited is equal to the number of vertices of the graph, the graph is connected. If this number is smaller, the graph is disconnected.

We decided that the disconnected graph is rejected and the next one is generated for a greater value of p, until correct graph is obtained. The obtained connected graph could be non-eulerian then the program makes some routines to convert it into eulerian one.

3. Transformation of an arbitrary graph into the Eulerian graph

The second condition for a graph to be Eulerian is the parity of the degree of all its vertices. For this purpose, an algorithm modifies the graph which has been implemented in the program till this moment.

The algorithm searches all vertices in the graph for odd pairs of degrees – the following cases are possible: (a) all vertices are even – the graph has an eulerian cycle; (b) all vertices are even besides two of them which are odd – eulerian path exists, (c) if there more than 2 odd-degree-vertices then repair is needed. Namely, after finding two vertices with an odd degree, an edge is created between vertices or removed if it already exists. The operation of annihilation of the edge is prohibited if one of end vertices is of degree 1, because resulting graph would be disconnected. Another pairs have to be listed. After performance of this operation, the algorithm continues to search the graph until it checks all vertices.

After searching the graph, it should have all vertices of the even degree. Then the next phase of activities is triggered.

4. Trajan Algorithm

Trajan's algorithm is an algorithm that finds bridges (bridge is an edge in the graph which when removed deprives the graph of connectivity) in the non-directed graph. This algorithm is based on depth-first search algorithm and creating a spanning tree. Fleury's algorithm requires knowledge of bridges in the graph, so the Trajan algorithm has been implemented in the program

The algorithm searches the graph and numbers all visited vertices starting from 1 in succession. This numbering is needed to determine the *Low(v)* parameter.
The *Low(v)* parameter for a given vertex v is the smallest number from the vertex number v assigned to it by the *DFS*, the Low parameters of all its sons in the spanning tree, and the *DFS* numbers of vertices connected to v by secondary edges (i.e. those that were not placed in the spanning tree). If we encounter a vertex v, whose number assigned by the *DFS* is equal to the Low(v) parameter, and the vertex is on the father's spanning tree, then the edge from this father to the vertex v is a bridge.

5. Fleury's Algorithm

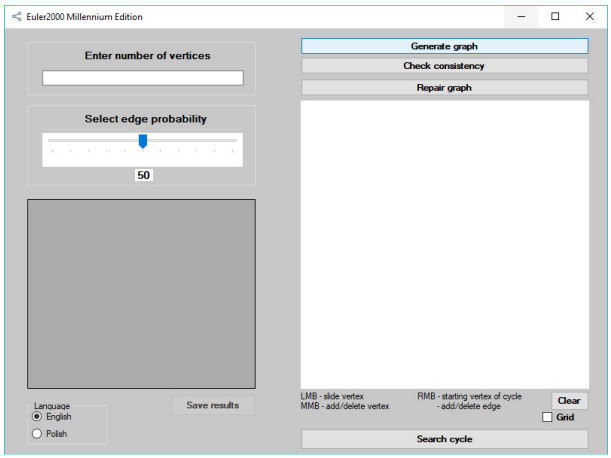
This algorithm is used to find the path or cycle of the Euler in the graph [4]. In the program, the algorithm uses the Trajan algorithm to find bridges.
At the beginning, the starting vertex is selected. Then choose an edge that is not a bridge (otherwise the graph would be disconnected and it would be impossible to find the cycle) unless there is no other choice, i.e. the vertex is connected to the rest of the graph only by an edge-bridge. This edge is saved in the stack. This edge is then passed to the next vertex and the edge is removed from the graph. In the new vertex, the entire procedure is repeated until all available edges have been passed.

If all the edges of the graph are in the saved stack and the final vertex is the same as the initial vertex, it means that the algorithm has found the Euler cycle. If the

start and end vertices are different, the graph is semi-Euler and has an Euler path. If the stack does not contain all the edges, the graph has neither a cycle nor an Euler path.

6. Application interface

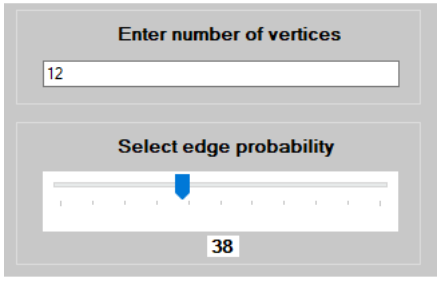
The application was written in C# using WindowsForm technology. The operation of the application can be described in several stages.



Screenshot 1. Main program window

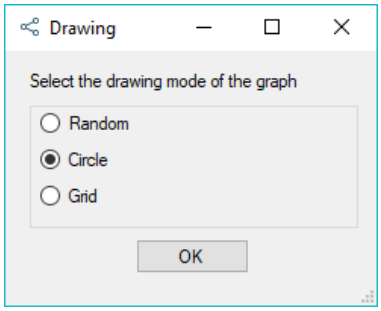
6.1. Graph generation

Two fields have been used to generate the graph (Screen 1). In the first field we enter the number of vertices of the graph, in the second using the slider we set the probability p of occurrence of edges between the two vertices (specified in percent) (Screen 2). After entering this data, you can use the "Generate Graph" button.



Screenshot 2. Data to generate the graph

A window appears (Screen 3) in which the graph drawing mode is selected.

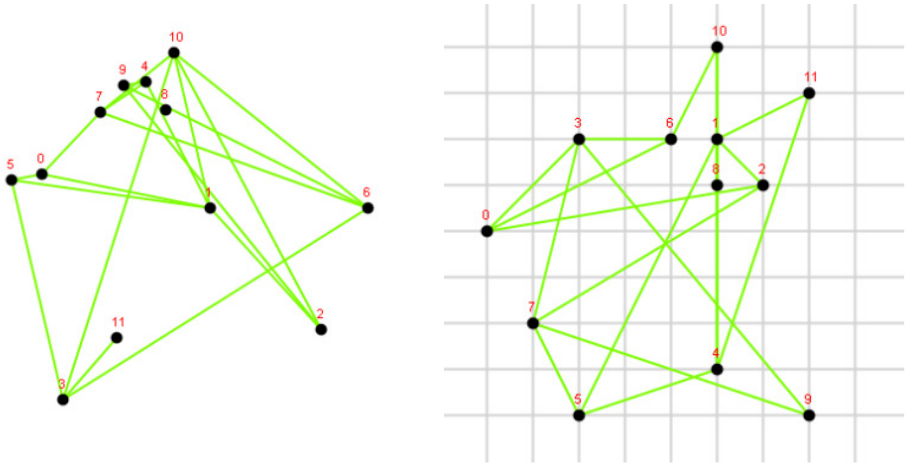


Screenshot 3. Graph drawing modes – user choses

	W0	W1	W2	W3	W
W0	0	1	0	0	1
W1	1	0	1	0	1
W2	0	1	0	1	0
W3	0	0	1	0	0
W4	1	1	0	0	0
W5	1	0	0	1	1
W6	1	0	0	0	0
W7	1	0	0	0	1
W8	0	1	1	0	0
W9	1	1	1	0	0

Screenshot 4. Neighbourhood matrix

A graph is generated, the visualizations of which can be seen on the panel. The window (Screen 4) also lists its neighborhood matrix.

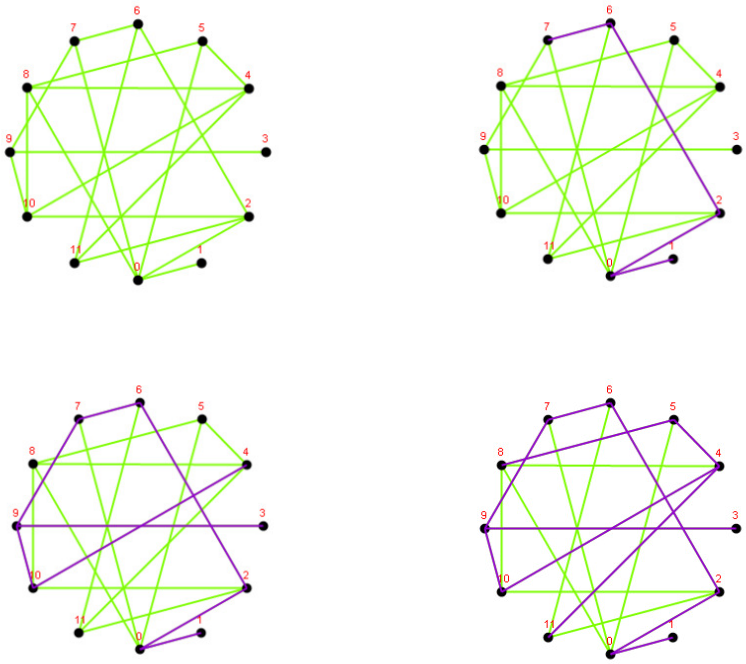


Screenshot 5. Graph visualization

The visualization panel is interactive. Utilizing the mouse it is possible to add new vertices and edges, as well as move vertices around the field. Thanks to this, the whole procedure of generating the graph can be omitted because the user can create any graph with the mouse (Screen 5).

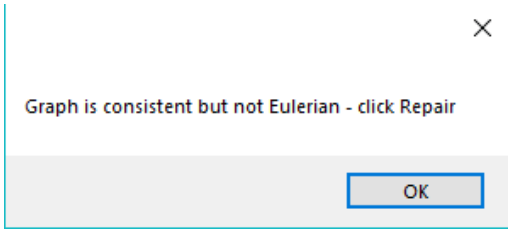
6.2. Connectivity check

The next step the user should take is to check the connectivity of the graph (Screen 6). This step can be omitted if the user knows that the graph is connected.



Screenshot 6. Graph connectivity – searching procedure vertex by vertex, the tree is spanned on all graph vertices

The button "Check connectivity" is used for this purpose - it runs a search algorithm whose steps can be traced in the animation. When the algorithm is finished, a window appears (Screen 7), informing if the graph is connected.



Screenshot 7. Connectivity information – here named as 'consistent'

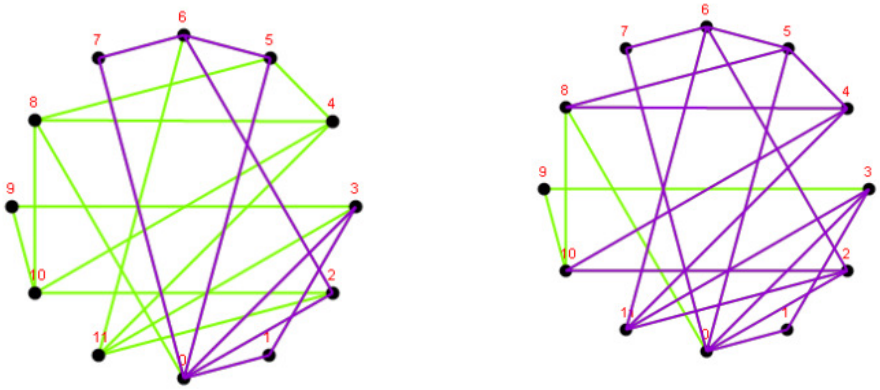
6.3. Graph Repair

To transform a particular graph into an Euler graph, use the "Repair Graph" button. It uses the algorithm described above in the article. After the algorithm is executed, the graph is drawn again together with the new edges.

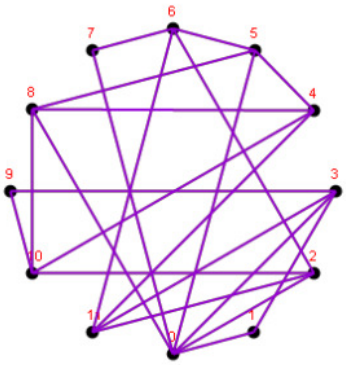
6.4 Euler cycle

The "Search cycle" button launches the Fleury algorithm - this algorithm can find the Euler cycle in the Euler graph or the Euler path in the semi-Euler graph. Otherwise, the algorithm will result in information about the lack of cycle.

The steps of the algorithm can be traced on the panel by means of animation, as it was during the search into the graph (Screen 8). As we can see in the upper window of the screen – every edge is passed only one time, vertices are visited more times depending on their degree – visiting means entering and going out of the vertex, therefore even degree is essentially needed. In Table 1, the Euler cycle (Screen 10) is listed edge by edge. Graph could also be semi-eulerian which is also detected by the discussed computer program.



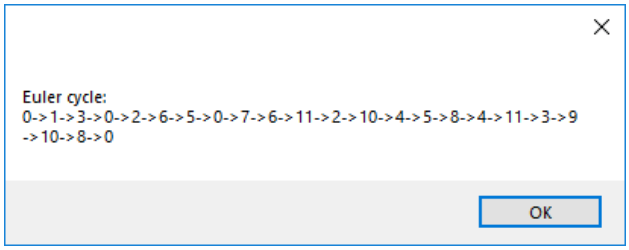
Screenshot 8. Euler cycle – searching procedure



Screenshot 9. Euler cycle - every edge is passed only once

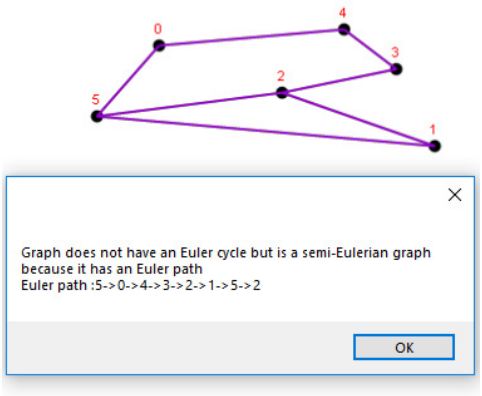
Table 1. Euler cycle for graph from Screenshot 9

Edge	
Vertex A	Vertex B
0	1
1	3
3	0
0	2
2	6
6	5
5	0
0	7
7	6
6	11
11	2
2	10
10	4
4	5
5	8
8	4
4	11
11	3
3	9
9	10
10	8
8	0



Screenshot 10. Information about Euler cycle

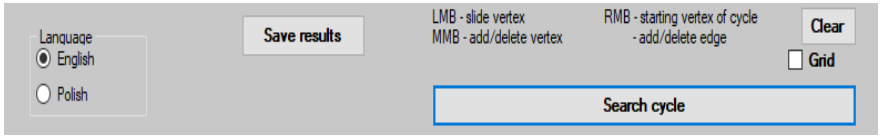
The discussed computer program could be used in the didactics of the subjects related to graph theory. The visual window shows the results of calculations in the way which helps the user to understand the algorithms. Usage of colors allows for observation of running of the program in step by step manner. The graph after repair procedure – in the proposed version – could be eulerian or semi-eulerian. Both cases are considered and solved. Eulerian cycle and eulerian path are presented as an image and as a path in the following form: 0→1→3 … 9→10→8→0 (see Screens 10 and 11). Eulerian cycle starts and ends in the same vertex i.e.: "0". Furthermore, eulerian path starts and ends in different vertices of odd degree (Screen 11).



Screenshot 11. Example of semi-Eulerian graph – having two vertices of odd degree and all other of even degree

6.5 Additional functions

At the end (Screen 12) it is possible to save the program result to a txt file - the "Save result" (Screen 13) button is used for this purpose.



Screenshot 12. Additional functions

The application also supports two languages - Polish and English, which we signal by means of radiobuttons. It is also possible to draw a grid on the panel - thanks to checkbox selection - "Grid". The "Clear" button removes the drawn graph.

7. Final remarks

The software dedicated for visualisation of finding an eulerian cycle in the simple graph is described in the present paper. The problem is considered as classical in graph theory. It has versatile applications (e.g. [1]) and is still under development [3].

```
ATH - 50 lat tradycji
Data i godzina: 14.02.2019 11:53:01
Zespół :
Paulina Stachnik
Michał Śliwa
Dane:
Ilość wierzchołków : 12
Prawdopodobienstwo : 38%
Wygenerowana macierz grafu:

                W0              W1              W2              W3
W0              0               1               1               0
W1              1               0               1               0
W2              1               1               0               1
W3              0               0               1               0
W4              1               1               0               0
W5              1               0               0               1
W6              1               0               0               0
W7              1               0               0               0
W8              0               1               1               0
W9              1               1               1               0
W10             0               0               1               1
W11             1               1               0               1
deg             8               6               6               4

Euler cycle: 0->1->2->0->4->1->8->2->3->5->0->6->7->0->9->1->11->3->10-
```

Screenshot 13. Fragment of the resulting file – the line named 'deg' shows degrees of the graph vertices, in this case all the values are even

In the described software, a user has a possibility to generate a graph, checking its connectivity as well as checking the necessary condition for existence of the considered cycle. In the case if the considered graph is non-eulerian, the procedure could be triggered which repairs the graph. After creation of eulerian graph the cycle could be distinguished. The advantage of the program is that all the functionalities are visualized on the computer screen, so it could be used for didactics of graph theory.

REFERENCES

1. DICKMAN P.: Incremental, distributed orphan detection and actor garbage collection using graph partitioning and Euler cycles. In: *International Workshop on Distributed Algorithms*. Springer, Berlin, Heidelberg (1996). 141-158.
2. ECKMANN B.: The Euler characteristic—a few highlights in its long history. *Mathematical Survey Lectures 1943–2004*, (2006) 177-188.
3. NEUMANN F.: Expected runtimes of evolutionary algorithms for the Eulerian cycle problem. In: *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*. IEEE (2004) 904-910.
4. WAŁASZEK J.: Algorytmy i struktury danych.
5. *https://eduinf.waw.pl/inf/alg/001_search/0130a.php#P2*
6. *https://eduinf.waw.pl/inf/alg/001_search/0135.php#P1*. (open: 10.03.2019).
7. WILSON R.: Wprowadzenie do teorii grafów, PWN, Warszawa (2019).