https://www.sworldjournal.com/index.php/swj/article/view/swj30-02-013
DOI: 10.30888/2663-5712.2025-30-02-013

UDC 519.177

POSSIBILITY OF MATRIX REPRESENTATION OF BASIC UNARY OPERATIONS ON GRAPHS

Yakimova Nataliya Anatoliivna
c. t. s., as. prof.
Odessa I.I. Mechnikov National University,
Odessa, str. Dvoryanska, 2, 65029, Ukraine

Abstract. *Today, science is faced with an increasing number of problems related to computerization. It is the needs of computer information processing that require the study of the possibilities of matrix representation of graphs and, as a result, matrix representation of all possible transformations of graphs. The matrix implementation of operations on graphs has its own characteristics depending on the type of each specific graph. Because of this, obviously, the procedure for their implementation is not universal. A graph can be fully characterized by the adjacency matrix. Therefore, for the matrix implementation of the considered basic unary operations on graphs, it is necessary to use the apparatus of ordinary arithmetic matrix operations described in linear algebra.*

Keywords: *directed and undirected graph, adjacency matrix, operation on graphs, arithmetic operations on matrices.*

Introduction.

When studying a system of objects connected by some arbitrary types of relationships, both directed and undirected graphs can be used. Each such system is an ordered collection of elements with which certain changes can occur. Each such specific system can be represented graphically as a graph or in digital format as an adjacency matrix or incidence matrix of such a graph. n any case, changes in the elements of the system are reflected by unary operations on the vertices or edges of the corresponding graph. The graphical implementation of such operations has already been well studied and described [1]. But computer processing of information involves its digital representation in matrix form. Algebraic matrix apparatus is also widely represented in mathematical research [2]. This article aims to establish a correspondence between known operations on ordinary matrices and unary operations on elements of an arbitrary graph. In this way, a transition from a graphical to a digital method of not only representing, but also processing various information can be made.

Main text.

Let’s consider an directed graph G_1 (Fig. 1).

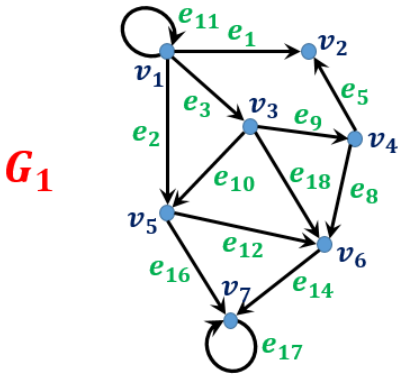

Figure 1 – Directed graph G_1

Author's development

The adjacency matrix corresponding to this graph will be as follows (in tabular or in the usual matrix form):

$A(G_1) =$

Initial \ Final	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	1	1	1	0	1	0	0
v_2	0	0	0	0	0	0	0
v_3	0	0	0	1	1	1	0
v_4	0	1	0	0	0	1	0
v_5	0	0	0	0	0	1	1
v_6	0	0	0	0	0	0	1
v_7	0	0	0	0	0	0	1

$$A(G_1) = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

<u>*Deleting a vertex*</u> from a graph entails deleting all edges incident to it, which results in deleting all connections of this object or node to other objects or nodes. This means that when deleting a vertex v_i from the adjacency matrix, the i-th row and i-th column must be deleted. In this regard, the algorithm for performing the operation of deleting a vertex v_i from a graph in a matrix representation is similar to the algorithm for constructing a minor M_{ii} for the adjacency matrix of this graph. For example, suppose that we want to delete vertex v_4 from graph G_1. This is equivalent to constructing a minor M_{44} for matrix $A(G_1)$:

$$M_{44} = \begin{vmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{vmatrix}.$$

Therefore, the new graph $G_1' = G_1 \backslash \{v_4\}$ will have the adjacency matrix corresponding to it

$$A(G_1') = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The algorithm for deleting rows and columns from a matrix has already been computerized. In this case, the software implementation will involve performing two shifts: for rows and columns. For a graph without isolated vertices, it is advisable to perform this operation using the adjacency matrix.

In the adjacency matrix, the sign of an isolated vertex is the presence of a zero row and column of the same name. In the incidence matrix, the sign of an isolated vertex is the presence of a zero row. Therefore, if the vertex to be deleting from the graph is isolated, then in this unique case it is more convenient to perform this operation using the incidence matrix. When a zero row is deleting from the incidence matrix, no connection between other vertices (objects or nodes) is broken. Therefore, there is no need to track the appearance of columns that, after deleting this row, will contain only one unit, which is not permissible for the incidence matrix [3]. Using the incidence matrix when deleting an isolated vertex halves the program implementation (there is no need to delete a column at the same time, and there will also be only one shift).

The operation of deleting one edge involves the disappearance of a single edge (connection or relation) from the graph while preserving all vertices (objects or nodes) and the remaining connections between objects [4]. Thus, only those elements in the adjacency matrix that correspond to the edges that are being deleting are changed. The remaining elements of the adjacency matrix retain their values. The order of the matrix also remains the same, because no vertex disappears from the graph as a result of deleting an edge. When a directed edge is deleting, only one element of the adjacency matrix is reducing. When an undirected edge is deleting, a pair of symmetric elements of the adjacency matrix undergo the same changes.

Let it be necessary to delete a certain number of l_{ij}^{-} edges between vertices v_i and

v_j. Therefore, to perform this operation, it is necessary to perform the arithmetic subtraction of the matrix A_e^- from the adjacency matrix of the initial graph. In the subtractor matrix A_e^-, the elements corresponding to the edges being deleted will be equal to l_{ij}^-. The remaining elements of this matrix will be zero.

Consider, for example, the graph G_2,

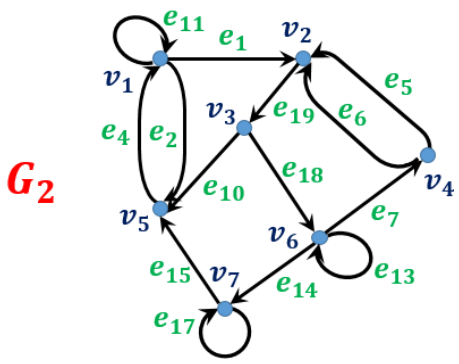

Figure 2 – Directed graph G_2

Author's development

to which the adjacency matrix corresponds (in tabular or in the usual matrix form)

$A(G_2) =$

Initial \ Final	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	1	1	0	0	1	0	0
v_2	0	0	1	0	0	0	0
v_3	0	0	0	0	1	1	0
v_4	0	2	0	0	0	0	0
v_5	1	0	0	0	0	0	0
v_6	0	0	0	1	0	1	1
v_7	0	0	0	0	1	0	1

$$A(G_2) = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

In this matrix there is a pair of symmetric elements a_{15}=a_{51}=1. These elements correspond to a pair of oppositely directed edges, which can be replaced by a single undirected edge. Suppose that this undirected edge needs to be deleted from this graph. Suppose that in addition, one of the strictly parallel directed edges from vertex v_4 to vertex v_2 needs to be deleted from this graph. This means that in the matrix A_e^- only the elements l_{42}^-=1 and l_{15}=l_{51}=1 will be non-zero. Therefore, to calculate the adjacency matrix of the graph

$$(G_2)' = (G_2)\backslash\{\vec{e}(v_4, v_2)\}\backslash\{e(v_1, v_5)\} =$$

$$= (G_2)\backslash\{\vec{e}(v_4, v_2)\}\backslash\{\vec{e}(v_1, v_5)\}\{\vec{e}(v_5, v_1)\}$$

(the notation $\vec{e}$ means that this edge is directed) can be obtained by performing the following arithmetic subtraction:

$$A(G_2) - A_e^- = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} = A((G_2)').$$

The software implementation of this operation is much simpler than its algebraic justification. To perform it, it is enough to reduce the corresponding elements of the adjacency matrix of the initial graph by the values l_{ij}^-.

The operation of inserting an edge is the inverse of the operation of deleting an edge. When inserting an directed edge, one element of the adjacency matrix is expected to increase. When inserting an undirected edge, a pair of symmetric elements of the adjacency matrix undergo the same changes. Let's say that between vertices v_i and v_j it is necessary to add a certain number of l_{ij}^+ edges. In the software implementation, the corresponding elements of the adjacency matrix of the initial graph are increased by the specified value. The algebraic implementation of this operation involves the arithmetic sum of two matrices. One of the terms is the adjacency matrix of the initial graph. In the second term A_e^+, the elements corresponding to the new edges are equal to l_{ij}^+. The remaining elements of the second term are zero.

Suppose, for example, that it is necessary to introduce into the graph G_1 two strictly parallel directed edges from the vertex v_2 to the vertex v_4 and one undirected edge between the vertices v_3 and v_7. This means that in this case l_{24}^{+}=2 and l_{37}^{+}=l_{73}^{+}=1. Then the adjacency matrix of the graph

$$G_1'' = G_1 + 2 \cdot \{\vec{e}(v_2, v_4)\} + \{e(v_3, v_7)\}$$

can be obtained by performing the following arithmetic sum:

$$A(G_1) + A_e^{+} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} = A(G_1'').$$

Conclusions.

Some operations are more convenient to perform with adjacency matrices, and some with incidence matrices. However, the most common algorithms are still based on processing adjacency matrices. The same algorithms differ depending on whether directed or free graphs are involved in the operations considered. Depending on the types of graphs, there are also restrictions on the representation of meaningful information by matrices of these graphs. But in practical applications, these limitations are usually insignificant. Therefore, for each considered unary operation on graphs and each type of graph, it is possible to propose a series of some arithmetic operations that allow obtaining the matrix of a new graph or a clear, easily programmable algorithm for transforming the matrices of the initial graphs. None of the considered unary operations on graphs is impossible in matrix implementation, and the proposed algorithms can significantly simplify computer processing of graphs.

References:

1. Zykov A.A. (2007). *Lectures on algebra*. Odessa: Astroprint, 400p.

2. Gantmaker F. (2010) *Matrix theory*. Kyiv: Nadrukovano v Ukraini, 560p.

3. Yakimova N.A., Klishin N.E. (2022). Matrix representation of operations on graphs. *Researches in mathematics and mechanics*. Vol.27. Issue 1 – 2(38 – 39) , pp. 121–141.

4. Yakimova N.A. (2022) *Discrete Mathematics. Part 1. Set theory. Graph theory.* Odessa: ONU im. I.I. Mechnikova, 102p.

Анотація. *Сьогодення ставить перед наукою все більшу кількість задач, пов'язаних із комп'ютеризацією. Саме потреби комп'ютерної обробки інформації потребують дослідження можливостей матричного подання графів і, як наслідок, матричного подання усіх можливих перетворень графів. Матрична реалізація операцій над графами має свої особливості в залежності від виду кожного конкретного графу. Через це, вочевидь, процедура їх виконання не є універсальною. Матриці суміжності та інцидентності повністю характеризують граф. Тому для матричного виконання розглянутих основних унарних операцій над графами треба залучати апарат звичайних арифметичних матричних операцій, описаних в лінійній алгебрі.*

Ключові слова: *орієнтований та неорієнтований граф, матриця суміжності, операції над графами, арифметичні операції над матрицями.*

The article has been sent: 13.03.2025.