INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

A Note on the Application of Identity Matrix

Arunodhaya K Nambiar[1], **Amrutha K Adiyodi**[2], **Sanil Shanker**[1]*

1 Department of Information Technology, Kannur University, Kerala, India
2 Department of Physics, Pazhassi Raja NSS College, Mattanur, Kerala, India

OPEN ACCESS

Received: 04-04-2024

Accepted: 03-05-2024

Published: 18-06-2024

Citation: Nambiar AK, Adiyodi AK, Shanker S (2024) A Note on the Application of Identity Matrix. Indian Journal of Science and Technology 17(25): 2599-2609. https://doi.org/10.17485/IJST/v17i25.930

* **Corresponding author**.

sanil@kannuruniv.ac.in

Funding: None

Competing Interests: None

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objectives: The Identity Matrix is an important notion in linear algebra with various applications across computational and mathematical domains. The objective of the paper is to introduce a novel method for transposing matrices by utilizing the properties of Identity Matrix as agent. **Method:** In the proposed method, the input matrix performs Sum of Product (SOP) operations with the Identity Matrix gives the transpose. This involves describing specific mathematical operations that exploit the properties of the Identity Matrix to achieve matrix transpose. **Findings:** The results show that the Identity Matrix can be used to transpose binary, non- binary and complex matrices. The paper explores results obtained through empirical studies. The result shows that the algorithm runs in polynomial time. **Novelty:** The case study explores the application of the Identity Matrix for transposing the binary, non- binary and complex matrices with the support of examples and implementation. The uniqueness of the method is its competence to transpose matrix by operating the Sum of Product (SOP) structure. The method of matrix transpose by briefing Identity Matrix as the agent can be applied to develop a way of research to address artificial intelligence problems.

Keywords: Agent Based Simulation; Binary Matrix; Complex Matrix; Identity Matrix; Matrix Transpose; Sum of Product

1 Introduction

The notion of an agent is normally used in agent-based simulation or modeling contexts and it does not apply to matrix operations like transpose. Agent-based model supports to simulate complex systems by modeling individual agents with their own rules, which can generate targeted results [1,2]. Even though agent-based simulation is a capable tool for studying complex systems, researchers in the field continue to work on developing methods to improve the strength and reliability of agent-based simulations. Linear algebra plays a significant role in the development of agent-based models to represent, simulate, and explore complex systems and the relations of agents within them [3]. The precise mathematical tools and techniques used in agent-based models depend on the nature of the problem and the goals of the modeling study. The paper investigates some possibilities for finding matrix transpose by exploiting the characteristics of the agent-based simulation.

In linear algebra, the Identity Matrix is a fundamental concept and finds applications in mathematical and scientific disciplines. The Identity Matrix functions as the unit element for describing matrix inverse, eigenvalue and eigenvector computations. The acquiescence of the Identity Matrix underlines its comprehensive role for simplifying operations, transformations, and computations across various applications[4]. For instance, it is endeavouring in the initialization of matrices and plays a significant role in numerical methods, such as the Gaussian elimination method for solving linear systems. Whether serving as a preliminary element in mathematical computations, the Identity Matrix stands as an impermeable to the efficacy of mathematical concepts in solving complex problems across miscellaneous technical scenarios. This paper explores the application of the Identity Matrix as agent, proving its significance for matrix transpose.

Matrix transpose is accomplished by changing the rows and columns of the given matrix. Review of the article Shanker KPS et al., 2010 explores the notion of binary matrix transpose by fitting together the features of logical AND with logical OR operations on pair of bits[5,6]. In the precinct of electronic circuits and binary systems, Sum of Product (SOP) expressions are structured by combining logical AND operations whose outputs are then connected through logical OR operations[7–9]. The notion behind the model of non-binary matrix transpose using Identity Matrix is the sum of product of real numbers; generally, refers to the mathematical operation of adding together the products of individual real numbers[10]. Transfer functions and system responses of control systems are conveyed using complex numbers and the Sum of Products of complex numbers can be involved in the design of control systems. The Sum of Product operation of complex numbers, coupled with the identity matrix, plays a pivotal role for transposing complex matrix.

2 Methodology

Let M be matrix with order p x q. The input vector performs Sum of Product (SOP) operations with the Identity Matrix of order p x p gives the transpose of M.

2.1 Procedure:

- Step 1. Initialize Identity Matrix, I $_{p\ x\ p}$
- Step 2. Input matrix, M $_{p\ x\ q}$
- Step 3. Compute the transpose by performing column wise sum of product operations.

Algorithm

Input: Matrix M of order p x q
Output: Cell values of transpose of M with order q x p
Data structure: Array

```
1. begin
2. Initialize Identity Matrix, I of order p x p
3. for j from 0 to q-1   do
4.      for i from 0 to p-1 do
5.       Set CellValue := 0
6.              for k from 0 to p-1 do
7.                      Compute CellValue := CellValue + M[i][j] * I[k][i]
8.              end for
9.       Print CellValue
10.     end for
11. end for
12. end
```

Fig 1. Algorithm for computing cell values

The Figure 1 explores the generation of cell values of the matrix transpose in polynomial time. For the transformation of one cell value, there exists p multiplications and p-1 additions result the computational time O(p^2). Here, for transformation of q number columns, the time complexity is O(qp^2).

3 Results and Discussion

The study conducted with binary matrix, non-binary matrix and complex matrices.

3.1 Case- 1 Binary matrix as input

Let M be a binary matrix of order p x q. The transpose of the binary matrix M $_{p\ x\ q}$ can be done by performing column wise sum of product operations of the matrix M $_{p\ x\ q}$ with the Identity Matrix I$_{p\ x\ p}$.

Example: Consider a binary matrix M with the order (4 x 5).

$$\begin{matrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Fig 2.

M operates logical AND with the Identity Matrix I of order (4 x 4) as follows:

$$\begin{array}{ccc} \mathbf{M} & & \mathbf{I_4} \\ \begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline \end{array} & \begin{array}{c} . \\ . \\ . \\ . \end{array} & \begin{array}{|c|c|c|c|} \hline 1 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \\ \hline \end{array} \end{array}$$

Fig 3.

Let x_1, x_2, x_3 and x_4 be the values of first row of the matrix transpose. Then x_1, x_2, x_3 and x_4 can be calculated as $\sum_{i=1}^{p} W_{i,j},\ where\ j = 1, 2, \ldots..p$. The binary vector performs the operation logical AND, with the Identity Matrix as in Table 1.

Table 1. Multiplication of binary vector with Identity Matrix

First Column of Input Matrix (M$_{i1}$)	$\wedge$	I$_4$ as agent			
1 (i=1)	.	1 (i=1,j=1)	0 (i=1,j=2)	0 (i=1,j=3)	0 (i=1,j=4)
0 (i=2)	.	0 (i=2,j=1)	1 (i=2,j=2)	0 (i=2,j=3)	0 (i=2,j=4)
1 (i=3)	.	0 (i=3,j=1)	0 (i=3,j=2)	1 (i=3,j=3)	0 (i=3,j=4)
0 (i=4)	.	0 (i=4,j=1)	0 (i=4,j=2)	0 (i=4,j=3)	1 (i=4,j=4)

(i) The value of x_1 can be computed as:

$$\sum_{i=1}^{4} W_{i,j=1} = W_{1,1} + W_{2,1} + W_{3,1} + W_{4,1} = 1 + 0 + 0 + 0 = 1$$

where,

$$W_{1,1} = M_{(i=1)} \cdot I_{i=1,\ \ j=1} = 1 \cdot 1 = 1$$

$$W_{2,1} = M_{(i=2)} \cdot I_{i=2,\ \ j=1} = 0 \,.\, 0 = 0$$

$$W_{3,1} = M_{(i=3)} \cdot I_{i=3,\ \ j=1} = 1 \,.\, 0 = 0$$

$$W_{4,1} = M_{(i=4)} \cdot I_{i=4,\ \ j=1} = 0 \,.\, 0 = 0$$

(ii) The value of x_2:

$$\sum_{i=1}^{4} W_{i,j=2} = W_{1,2} + W_{2,2} + W_{3,2} + W_{4,2} = 0+0+0+0 = 0$$

(iii) The value of x_3:

$$\sum_{i=1}^{4} W_{i,j=3} = W_{1,3} + W_{2,3} + W_{3,3} + W_{4,3} = 0+0+1+0 = 1$$

(iv) The value of x_4:

$$\sum_{i=1}^{4} W_{i,j=4} = W_{1,4} + W_{2,4} + W_{3,4} + W_{4,4} = 0+0+0+0 = 0$$

The first row of the transpose matrix will be as follows.

x_1	x_2	x_3	x_4
1	0	1	0

Fig 4.

Apply this procedure for computing the values of the remaining rows. The transpose of the binary matrix M^T of order (5 x 4) is:

$$\begin{matrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{matrix}$$

Fig 5.

3.2 Case- 2 Non- binary matrix as input

Let M $_{p\ x\ q}$ be a non- binary matrix. The transpose of the non- binary matrix M $_{p\ x\ q}$ can be done by performing column wise sum of product operations of the matrix M $_{p\ x\ q}$ with the Identity Matrix.

Example: Let M be a non- binary matrix with order (3 x 4) (Figure 6).

M Performs column wise multiplication with the Identity Matrix I (Figure 7).

$$\begin{matrix} 2 & 3 & 5 & 6 \\ 0 & 1 & 9 & 8 \\ 7 & 4 & 3 & 4 \end{matrix}$$

Fig 6.

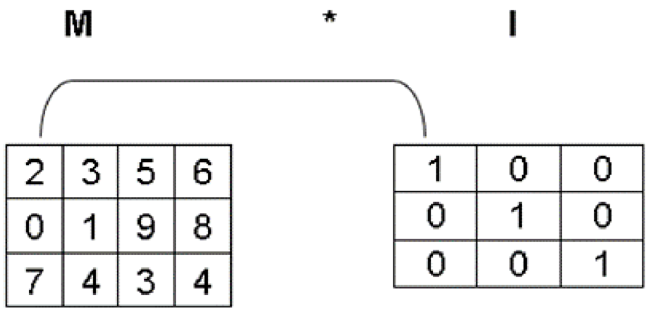


Fig 7.

Table 2. Multiplication of non- binary vector with Identity Matrix

First Column of Input Matrix (M_{i1})			I_3 as agent	
2 (i=1)	*	1 (i=1,j=1)	0 (i=1,j=2)	0 (i=1,j=3)
0 (i=2)	*	0 (i=2,j=1)	1 (i=2,j=2)	0 (i=2,j=3)
7 (i=3)	*	0 (i=3,j=1)	0 (i=3,j=2)	1 (i=3,j=3)

Let x_1, x_2 and x_3 be the values of first row of the matrix transpose. Then x_1, x_2 and x_3 can be calculated as $\sum_{i=1}^{p} W_{i,j},\ where\ j = 1,2,\ldots.p$. The input vector operates multiplication with the Identity Matrix worked out in Table 2.

(i) The value of x_1 can be computed as:

$$\sum_{i=1}^{3} W_{i,j=1} = W_{1,1} + W_{2,1} + W_{3,1} = 2 + 0 + 0 = 2$$

where,

$$W_{1,1} = M_{(i=1)} * I_{i=1,\ \ j=1} = 2 * 1 = 2$$

$$W_{2,1} = M_{(i=2)} * I_{i=2,\ \ j=1} = 0 * 0 = 0$$

$$W_{3,1} = M_{(i=3)} * I_{i=3,\ \ j=1} = 7 * 0 = 0$$

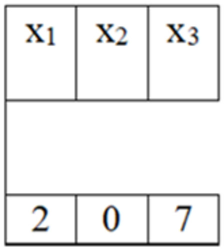
(ii) The value of x_2:

$$\sum_{i=1}^{3} W_{i,j=2} = W_{1,2} + W_{2,2} + W_{3,2} = 0 + 0 + 0 = 0$$

(iii) The value of x_3:

$$\sum_{i=1}^{3} W_{i,j=3} = W_{1,3} + W_{2,3} + W_{3,3} = 0+0+7 = 7$$

The first row of the transpose matrix will be as follows.

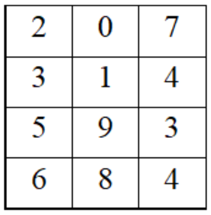


x_1	x_2	x_3
2	0	7

Fig 8.

The same procedure can be applied to calculate the values of the remaining rows. Then the transpose of the matrix will be as follows.

Then M^T is:



2	0	7
3	1	4
5	9	3
6	8	4

Fig 9.

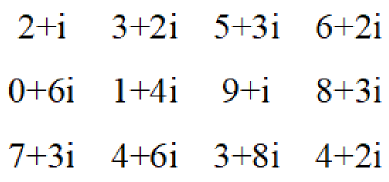
Here, M^T of order (4 x 3) is the transpose of M.

3.3 Case- 3 Complex matrix as input

Let C be the complex matrix of order p x q. This input complex matrix C performs column wise sum of product operations with the Identity Matrix I of order p x p which gives transpose of C.

Example: Let C be a complex matrix with order (3 x 4).

Here, C is:



2+i	3+2i	5+3i	6+2i
0+6i	1+4i	9+i	8+3i
7+3i	4+6i	3+8i	4+2i

Fig 10.

C Performs column wise multiplication with the Identity Matrix I (Figure 11).

Let x_1, x_2 and x_3 be the values of first row of the matrix transpose. Then $x_{1,}$ x_2 and x_3 can be calculated as$\sum_{i=1}^{p} W_{i,j},\ where\ j = 1, 2, \ldots ..p$. The complex vector operates multiplication with the Identity Matrix carried out in Table 3.

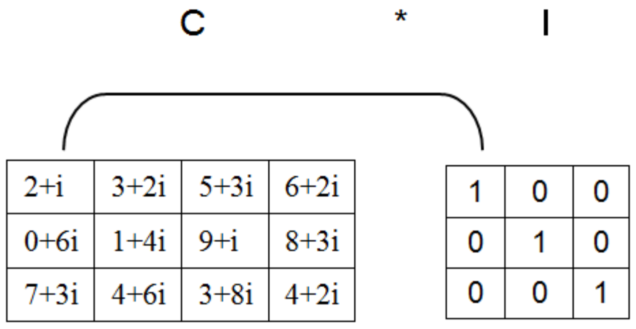


Fig 11.

Table 3. Multiplication of complex vector with the Identity Matrix

First Column of Input Matrix (C_{i1})	*	**I_3 as agent**		
2+i (i=1)	.	1 (i=1,j=1)	0 (i=1,j=2)	0 (i=1,j=3)
0+6i (i=2)	.	0 (i=2,j=1)	1 (i=2,j=2)	0 (i=2,j=3)
7+3i (i=3)	.	0 (i=3,j=1)	0 (i=3,j=2)	1 (i=3,j=3)

(i) The value of x_1 can be computed as:

$$\sum_{i=1}^{3} W_{i,j=1} = \ W_{1,1} + W_{2,1} + W_{3,1} = (2+i) + 0 + 0 = 2+i$$

(ii) The value of x_2:

$$\sum_{i=1}^{3} W_{i,j=2} = \ W_{1,2} + W_{2,2} + W_{3,2} = 0 + (0+6i) + 0 = 0+6i$$

(iii) The value of x_3:

$$\sum_{i=1}^{3} W_{i,j=3} = \ W_{1,3} + W_{2,3} + W_{3,3} = 0 + 0 + (7+3i) = 7+3i$$

The first row of the transpose matrix will be as follows.

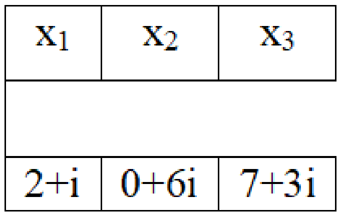
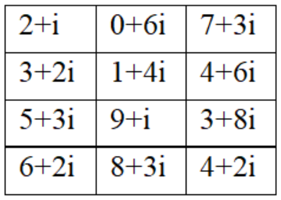


Fig 12.

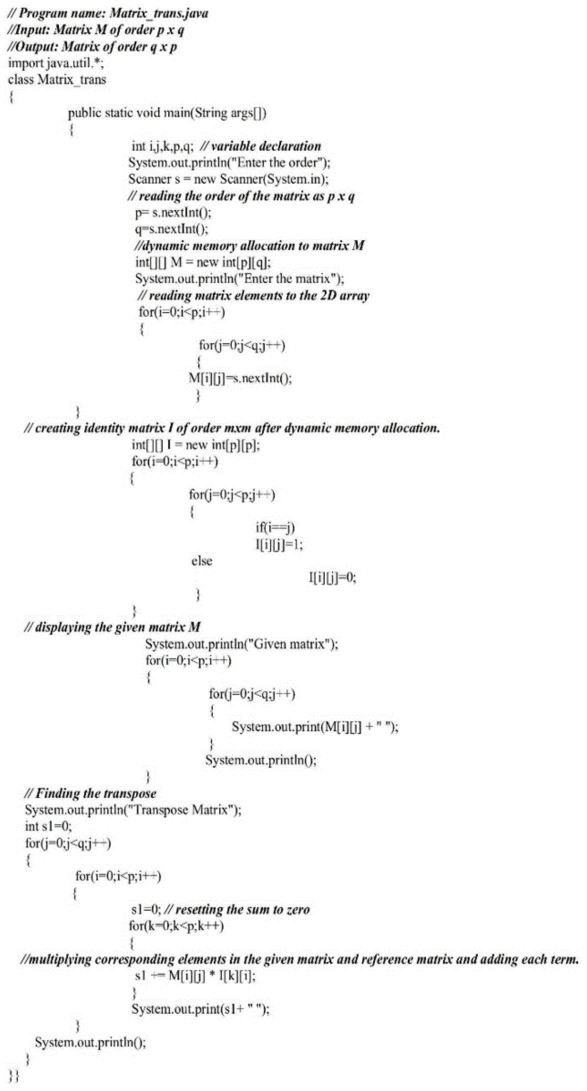
The same procedure can be applied to calculate the values of the remaining rows. Then the transpose of the complex matrix is as follows.



2+i	0+6i	7+3i
3+2i	1+4i	4+6i
5+3i	9+i	3+8i
6+2i	8+3i	4+2i

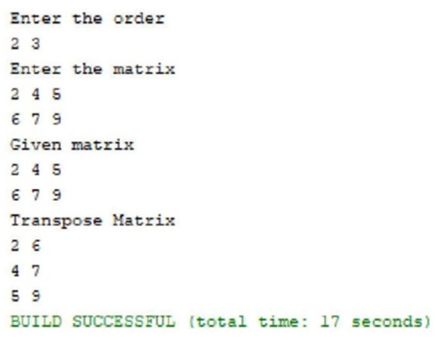
Fig 13.

That is the transpose of complex matrix C is C^T of order (4 x 3). The examples and simulated results show that the Identity Matrix can be used to transpose binary, non- binary and complex matrices. The Figures 14 and 15 explore the simulations of non-binary matrix as input. The Figures 16 and 17 travel around the code and simulation result of matrix transpose when complex matrix as input.



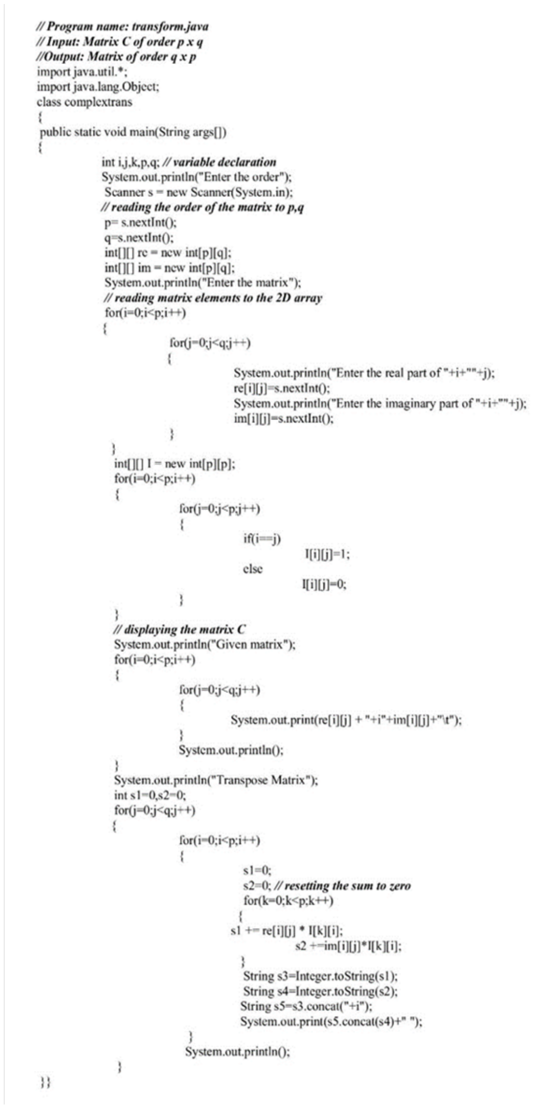
```
// Program name: Matrix_trans.java
//Input: Matrix M of order p x q
//Output: Matrix of order q x p
import java.util.*;
class Matrix_trans
{
            public static void main(String args[])
            {
                        int i,j,k,p,q; // variable declaration
                       System.out.println("Enter the order");
                       Scanner s = new Scanner(System.in);
                       // reading the order of the matrix as p x q
                        p= s.nextInt();
                        q=s.nextInt();
                        //dynamic memory allocation to matrix M
                        int[][] M = new int[p][q];
                        System.out.println("Enter the matrix");
                         // reading matrix elements to the 2D array
                         for(i=0;i<p;i++)
                         {
                                    for(j=0;j<q;j++)
                                    {
                                  M[i][j]=s.nextInt();
                                    }
            }
   // creating identity matrix I of order mxm after dynamic memory allocation.
                        int[][] I = new int[p][p];
                        for(i=0;i<p;i++)
                        {
                                  for(j=0;j<p;j++)
                                  {
                                              if(i==j)
                                              I[i][j]=1;
                                  else
                                                        I[i][j]=0;
                                   }
                        }
   // displaying the given matrix M
                          System.out.println("Given matrix");
                          for(i=0;i<p;i++)
                          {
                                     for(j=0;j<q;j++)
                                     {
                                         System.out.print(M[i][j] + " ");
                                     }
                                    System.out.println();
                          }
   // Finding the transpose
   System.out.println("Transpose Matrix");
   int s1=0;
   for(j=0;j<q;j++)
   {
            for(i=0;i<p;i++)
            {
                        s1=0; // resetting the sum to zero
                       for(k=0;k<p;k++)
                       {
  //multiplying corresponding elements in the given matrix and reference matrix and adding each term.
                        s1 += M[i][j] * I[k][i];
                       }
                       System.out.print(s1+ " ");
            }
     System.out.println();
   }
}}
```

Fig 14. Matrix transpose when non-binary matrix as input



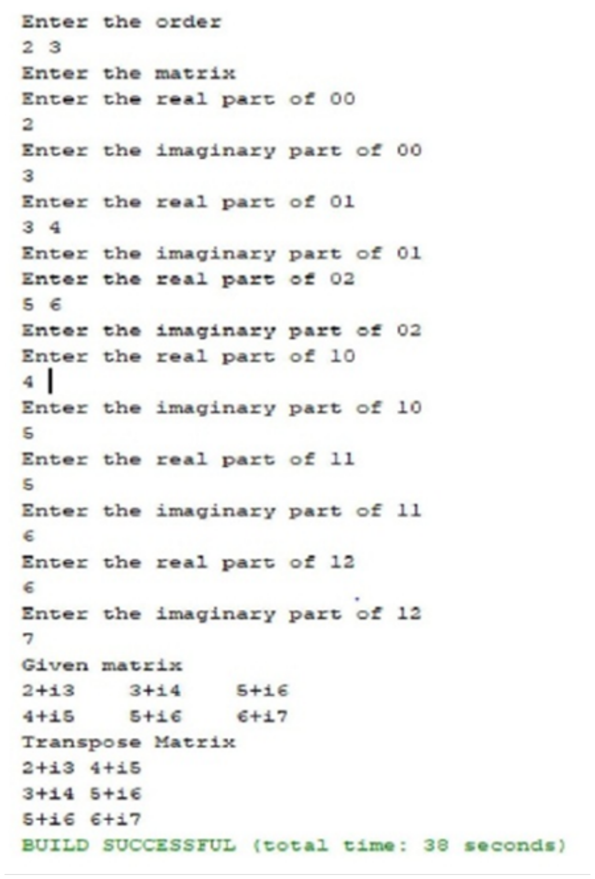
```
Enter the order
2 3
Enter the matrix
2 4 5
6 7 9
Given matrix
2 4 5
6 7 9
Transpose Matrix
2 6
4 7
5 9
BUILD SUCCESSFUL (total time: 17 seconds)
```

Fig 15. The output when non-binary matrix as input



```
// Program name: transform.java
// Input: Matrix C of order p x q
//Output: Matrix of order q x p
import java.util.*;
import java.lang.Object;
class complextrans
{
 public static void main(String args[])
{
            int i,j,k,p,q; // variable declaration
            System.out.println("Enter the order");
             Scanner s = new Scanner(System.in);
            // reading the order of the matrix to p,q
             p= s.nextInt();
             q=s.nextInt();
             int[][] re = new int[p][q];
             int[][] im = new int[p][q];
             System.out.println("Enter the matrix");
             // reading matrix elements to the 2D array
             for(i=0;i<p;i++)
             {
                         for(j=0;j<q;j++)
                         {
                                     System.out.println("Enter the real part of "+i+""+j);
                                     re[i][j]=s.nextInt();
                                     System.out.println("Enter the imaginary part of "+i+""+j);
                                     im[i][j]=s.nextInt();
                         }
              }
              int[][] I = new int[p][p];
              for(i=0;i<p;i++)
              {
                          for(j=0;j<p;j++)
                          {
                                      if(i==j)
                                                  I[i][j]=1;
                                      else
                                                 I[i][j]=0;
                          }
              }
              // displaying the matrix C
              System.out.println("Given matrix");
              for(i=0;i<p;i++)
              {
                          for(j=0;j<q;j++)
                          {
                                     System.out.print(re[i][j] + "+i"+im[i][j]+"\t");
                          }
                          System.out.println();
              }
              System.out.println("Transpose Matrix");
              int s1=0,s2=0;
              for(j=0;j<q;j++)
              {
                          for(i=0;i<p;i++)
                          {
                                      s1=0;
                                      s2=0; // resetting the sum to zero
                                      for(k=0;k<p;k++)
                                     {
                                    s1 += re[i][j] * I[k][i];
                                                s2 +=im[i][j]*I[k][i];
                                     }
                                      String s3=Integer.toString(s1);
                                      String s4=Integer.toString(s2);
                                     String s5=s3.concat("+i");
                                     System.out.print(s5.concat(s4)+" ");
                           }
                          System.out.println();
               }
}}
```

Fig 16. Matrix transpose when complex matrix as input



```
Enter the order
2 3
Enter the matrix
Enter the real part of 00
2
Enter the imaginary part of 00
3
Enter the real part of 01
3 4
Enter the imaginary part of 01
Enter the real part of 02
5 6
Enter the imaginary part of 02
Enter the real part of 10
4 |
Enter the imaginary part of 10
5
Enter the real part of 11
5
Enter the imaginary part of 11
6
Enter the real part of 12
6
Enter the imaginary part of 12
7
Given matrix
2+i3    3+i4    5+i6
4+i5    5+i6    6+i7
Transpose Matrix
2+i3 4+i5
3+i4 5+i6
5+i6 6+i7
BUILD SUCCESSFUL (total time: 38 seconds)
```

Fig 17. The output when complex matrix as input

4 Conclusion

The method for transposing the binary, non- binary and complex matrices by utilizing the characteristics of Identity Matrix as agent is illustrated in this paper. The results explore that the Identity Matrix can be applied to develop the tool for transposing the matrix. The algorithm is applicable for matrix transformation in polynomial time by operating p multiplications and p-1 additions. The study confirms the applicability of the proposed method on different types of matrices. The proposed method can be applied to develop a way of research to address artificial intelligence problems such as reversing data streams, finding the transpose of character matrix and data encoding.

References

1) Bhattacharya P, Ekanayake S, Kuhlman CJ, Lebiere C, Morrison D, Swarup S, et al. The Matrix: An Agent-Based Modeling Framework for Data Intensive Simulations. In: AAMAS '19: Proceedings of the 18th International Conference on Autonomous Agents and Multi Agent Systems. 2019;p. 1635–1643. Available from: https://dl.acm.org/doi/10.5555/3306127.3331884.
2) Adiga A, Barrett C, Eubank S, Kuhlman CJ, Marathe MV, Mortveit HS. Validating agent-based models of large networked systems. In: Proceedings of the 2019 Winter Simulation Conference. 2020;p. 2807–2818. Available from: https://doi.org/10.1109/WSC40007.2019.9004718.
3) Bandini S, Manzoni S, Vizzari G. Agent Based Modeling and Simulation. *Encyclopedia of Complexity and Systems Science*. 2009;p. 184–197. Available from: https://doi.org/10.1007/978-0-387-30440-3_12.
4) Vermeersch C, De Moor B. Two Double Recursive Block Macaulay Matrix Algorithms to Solve Multiparameter Eigenvalue Problems. *IEEE Control Systems Letters*. 2022;7:319–324. Available from: https://doi.org/10.1109/LCSYS.2022.3182632.
5) Shanker KPS, Turner A, Sherly E, Austin J. Sequential data mining using correlation matrix memory. In: 2010 International Conference on Networking and Information Technology. 2010;p. 470–472. Available from: https://doi.org/10.1109/ICNIT.2010.5508469.
6) Shanker KPS. An Algorithm to Transpose Zero One Matrix. *International Journal of Computer Science and Information Technologies*. 2016;7(4):1960–1961. Available from: https://ijcsit.com/docs/Volume%207/vol7issue4/ijcsit2016070464.pdf.
7) Desana M, Schnörr C. Sum-product graphical models. *Machine Learning*. 2019;109:135–173. Available from: https://doi.org/10.1007/s10994-019-05813-2.

8) Rosowski A. Fast commutative matrix algorithms. *Journal of Symbolic Computation*. 2023;114:302–321. Available from: https://doi.org/10.1016/j.jsc.2022.05.002.

9) Sanchez-Cauce R, Paris I, Diez FJ. Sum-product networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2022;44:3821–3839. Available from: https://www.computer.org/csdl/journal/tp/2022/07/09363463/1rvycXMzsRO.

10) Liao J, Liu H, Shao M, Xu X. A matrix identity and its applications. *Linear Algebra and its Applications*. 2015;471:346–352. Available from: https://doi.org/10.1016/j.laa.2014.12.032.