Day 13: Matrix Transpose

Abhinav Yadav

"Everything should be made as simple as possible, but not simpler."
— Albert Einstein

---

1 Introduction

The transpose of a matrix is obtained by swapping its rows with columns. Transposing is useful in many computational tasks such as solving equations, data transformation, and image processing.

2 Problem Statement

Problem: Compute the transpose of a matrix. **Hint:** Swap elements `mat[i][j]` with `mat[j][i]`. **Edge Case:** Handle square and non-square matrices separately.

3 Algorithm

1. Input the matrix dimensions `rows` and `cols`.
2. Store the input elements in a 2D array.
3. Create another 2D array of size `cols x rows`.
4. Swap elements `mat[i][j]` with `transposed[j][i]`.

4 Code

```
import java.util.Scanner;

public class MatrixTranspose {

    public static void inputMatrix(int rows, int cols, int[][] matrix, Scan
        System.out.println("Enter elements of the " + rows + "x" + cols + "
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
```

```
                matrix[i][j] = scanner.nextInt();
            }
        }
    }

    public static void printMatrix(int rows, int cols, int[][] matrix) {
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }

    public static void transposeMatrix(int rows, int cols, int[][] matrix,
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                transposed[j][i] = matrix[i][j];
            }
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter rows and columns of the matrix: ");
        int rows = scanner.nextInt();
        int cols = scanner.nextInt();

        int[][] matrix = new int[rows][cols];
        int[][] transposed = new int[cols][rows];

        inputMatrix(rows, cols, matrix, scanner);
        transposeMatrix(rows, cols, matrix, transposed);

        System.out.println("Original Matrix:");
        printMatrix(rows, cols, matrix);

        System.out.println("Transposed Matrix:");
        printMatrix(cols, rows, transposed);

        scanner.close();
    }
}
```

5 Complexity Analysis

- **Time Complexity:** $O(m \times n)$, where m and n are the dimensions of the matrix.
- **Space Complexity:** $O(m \times n)$, for storing the transposed matrix.

6 Examples and Edge Cases

Matrix	Transposed Matrix	Comments
$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$	Square matrix
$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$	$\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$	Non-square matrix
$\begin{bmatrix} 7 \end{bmatrix}$	$\begin{bmatrix} 7 \end{bmatrix}$	Single-element matrix

7 Output

8 Conclusion

Matrix transposition is a simple but essential operation in computational mathematics. It demonstrates the importance of manipulating rows and columns effectively. This implementation works efficiently for both square and non-square matrices with $O(m \times n)$ complexity.

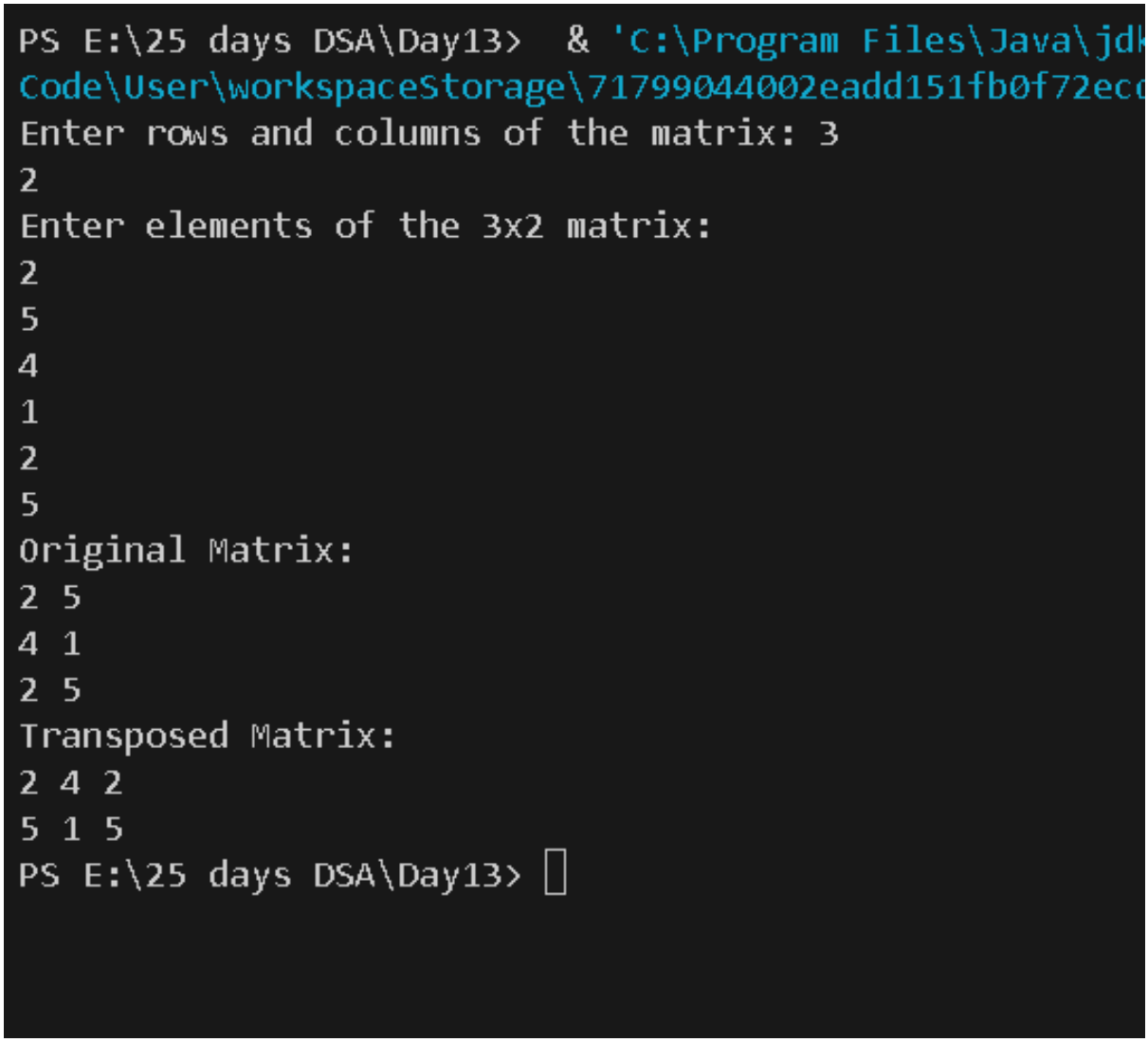

Figure 1: Program Output Screenshot