Day 13: Matrix Transpose

Gati Goyal

"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

```
#include <stdio.h>

void inputMatrix(int rows, int cols, int matrix[rows][cols]) {
    printf("Enter elements of the %dx%d matrix:\n", rows, cols);
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }
}

void printMatrix(int rows, int cols, int matrix[rows][cols]) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d␣", matrix[i][j]);
        }
        printf("\n");
    }
}

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

int main() {
    int rows, cols;

    printf("Enter␣rows␣and␣columns␣of␣the␣matrix:␣");
    scanf("%d␣%d", &rows, &cols);

    int matrix[rows][cols], transposed[cols][rows];

    inputMatrix(rows, cols, matrix);

    transposeMatrix(rows, cols, matrix, transposed);

    printf("Original␣Matrix:\n");
    printMatrix(rows, cols, matrix);

    printf("Transposed␣Matrix:\n");
    printMatrix(cols, rows, transposed);

    return 0;
}
```

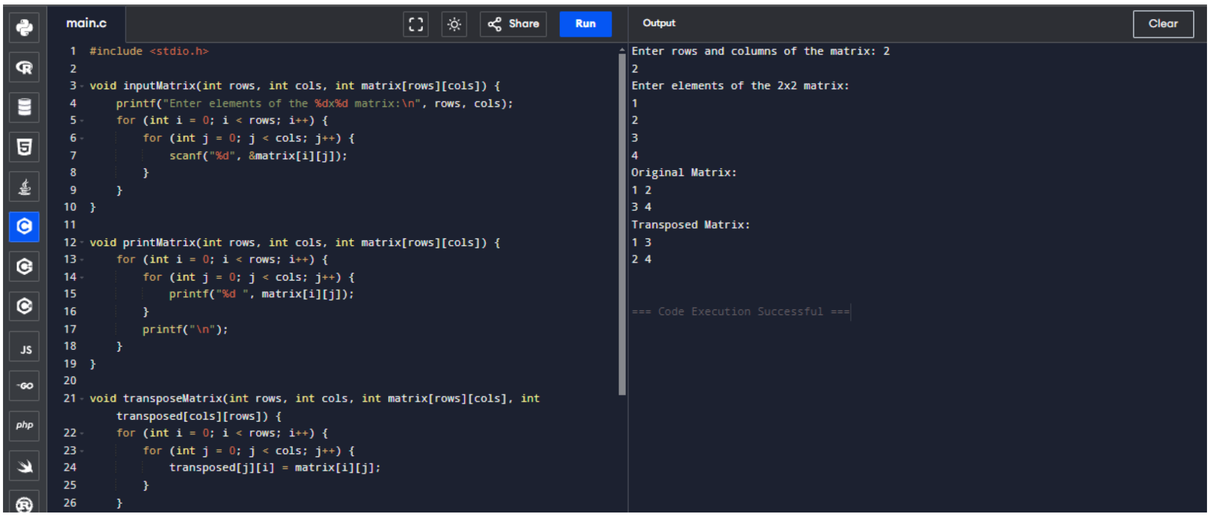
5 Complexity Analysis

- **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



```c
#include <stdio.h>

void inputMatrix(int rows, int cols, int matrix[rows][cols]) {
    printf("Enter elements of the %dx%d matrix:\n", rows, cols);
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }
}

void printMatrix(int rows, int cols, int matrix[rows][cols]) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}

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

```
Enter rows and columns of the matrix: 2
2
Enter elements of the 2x2 matrix:
1
2
3
4
Original Matrix:
1 2
3 4
Transposed Matrix:
1 3
2 4


=== Code Execution Successful ===
```

Figure 1: Program Output Screenshot

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.