Day 12: Matrix Multiplication

Abhinav Yadav

"Mathematics is not about numbers, equations, or algorithms: it is about understanding."
— William Paul Thurston

---

1 Introduction

Matrix multiplication is a fundamental operation in linear algebra with wide applications in computer science, engineering, and data science. The task involves computing the dot product of rows from the first matrix and columns from the second matrix.

2 Problem Statement

Problem: Multiply two matrices of compatible sizes. **Hint:** Use nested loops for dot product computation. **Edge Case:** Ensure the number of columns in the first matrix equals the number of rows in the second matrix.

3 Algorithm

1. Input dimensions of the two matrices.
2. Check if the matrices are compatible for multiplication (`c1 == r2`).
3. Initialize a result matrix of size `r1 x c2` with zeros.
4. Compute each element of the result matrix:
   - Iterate through rows of the first matrix and columns of the second matrix.
   - Use a nested loop for element-wise multiplication and summation.

4 Code

```java
import java.util.Scanner;

public class MatrixMultiplication {

    // Method to input a matrix
    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();
            }
        }
    }

    // Method to print a matrix
    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();
        }
    }

    // Method to multiply two matrices
    public static void multiplyMatrices(int r1, int c1, int r2, int c2, int
        for (int i = 0; i < r1; i++) {
            for (int j = 0; j < c2; j++) {
                result[i][j] = 0;
                for (int k = 0; k < c1; k++) {
                    result[i][j] += mat1[i][k] * mat2[k][j];
                }
            }
        }
    }

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

        // Input dimensions for the first matrix
        System.out.print("Enter rows and columns for the first matrix: ");
        int r1 = scanner.nextInt();
        int c1 = scanner.nextInt();

        // Input dimensions for the second matrix
        System.out.print("Enter rows and columns for the second matrix: ");
        int r2 = scanner.nextInt();
        int c2 = scanner.nextInt();
```

```java
        // Check if multiplication is possible
        if (c1 != r2) {
            System.out.println("Matrix multiplication not possible. Columns
            scanner.close();
            return;
        }

        // Initialize matrices
        int[][] mat1 = new int[r1][c1];
        int[][] mat2 = new int[r2][c2];
        int[][] result = new int[r1][c2];

        // Input matrices
        inputMatrix(r1, c1, mat1, scanner);
        inputMatrix(r2, c2, mat2, scanner);

        // Multiply matrices
        multiplyMatrices(r1, c1, r2, c2, mat1, mat2, result);

        // Print the resultant matrix
        System.out.println("Resultant matrix after multiplication:");
        printMatrix(r1, c2, result);

        scanner.close();
    }
}
```

5 Complexity Analysis

- **Time Complexity:** $O(m \times n \times p)$, where m, n, p are the dimensions of the matrices.
- **Space Complexity:** $O(m \times p)$, for the result matrix.

6 Examples and Edge Cases

Matrix 1	Matrix 2	Result	Comments
$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$	$\begin{bmatrix}5 & 6\\7 & 8\end{bmatrix}$	$\begin{bmatrix}19 & 22\\43 & 50\end{bmatrix}$	Valid
$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}$	$\begin{bmatrix}7\\8\\9\end{bmatrix}$	$\begin{bmatrix}50\\122\end{bmatrix}$	Valid
$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$	$\begin{bmatrix}5\\6\end{bmatrix}$	-	Invalid dimensions

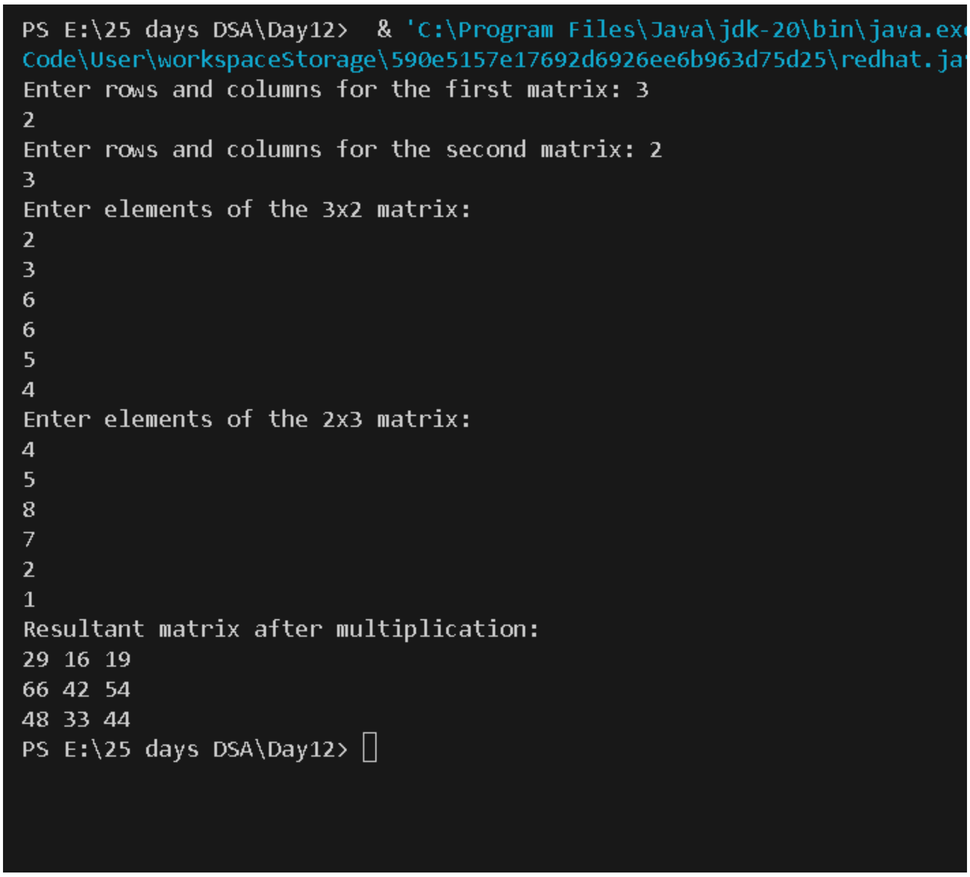

Figure 1: Program Output Screenshot

7 Conclusion

Matrix multiplication forms the foundation of many computational algorithms. This implementation efficiently computes the product of two matrices, leveraging nested loops for element-wise operations. The approach ensures compatibility of dimensions before performing operations, highlighting the importance of input validation in matrix computations.