A paper with text and numbers

Description automatically generated

A page of a document

Description automatically generated with medium confidence

Square Matrix Multiply code:

public class SquareMatrixMultiply{

    public int[][] multiply(int[][] A, int[][] B){

        int n = A.length;

        int[][] C = new int[n][n];

        for(int i=0; i<n; i++){

            for(int j=0; j<n; j++){

                C[i][j] = 0;

                for(int k=0; k<n; k++){

                    C[i][j] += A[i][k] \* B[k][j];

                }

            }

        }

        return C;

    }

    public static void main(String[] args) {

        SquareMatrixMultiply smm = new SquareMatrixMultiply();

        int[][] A = {{1, 2}, {3, 4}};

        int[][] B = {{5, 6}, {7, 8}};

        int[][] C = smm.multiply(A, B);

        for(int i=0; i<C.length; i++){

            for(int j=0; j<C.length; j++){

                System.out.print(C[i][j] + " ");

            }

            System.out.println();

        }

    }

}

Square Matrix Multiply output:

A screen shot of a computer program

Description automatically generated

Strassen Square Matrix Multiply code:

public class StrassenSquareMatrixMultiply {

    public int[][] multiply(int[][] A, int[][] B) {

        int n = A.length;

        // Check if matrices are not square or not power of 2

        if (!isPowerOfTwo(n) || A[0].length != n || B.length != n || B[0].length != n) {

            throw new IllegalArgumentException("Matrices are not square or not power of 2.");

        }

        // If input matrices are of size 1x1, perform simple multiplication

        if (n == 1) {

            int[][] C = new int[1][1];

            C[0][0] = A[0][0] \* B[0][0];

            return C;

        }

        // Divide input matrices into submatrices

        int[][] A11 = new int[n / 2][n / 2];

        int[][] A12 = new int[n / 2][n / 2];

        int[][] A21 = new int[n / 2][n / 2];

        int[][] A22 = new int[n / 2][n / 2];

        int[][] B11 = new int[n / 2][n / 2];

        int[][] B12 = new int[n / 2][n / 2];

        int[][] B21 = new int[n / 2][n / 2];

        int[][] B22 = new int[n / 2][n / 2];

        divideMatrix(A, A11, A12, A21, A22);

        divideMatrix(B, B11, B12, B21, B22);

        // Calculate the 7 products of submatrices

        int[][] M1 = multiply(addMatrix(A11, A22), addMatrix(B11, B22));

        int[][] M2 = multiply(addMatrix(A21, A22), B11);

        int[][] M3 = multiply(A11, subtractMatrix(B12, B22));

        int[][] M4 = multiply(A22, subtractMatrix(B21, B11));

        int[][] M5 = multiply(addMatrix(A11, A12), B22);

        int[][] M6 = multiply(subtractMatrix(A21, A11), addMatrix(B11, B12));

        int[][] M7 = multiply(subtractMatrix(A12, A22), addMatrix(B21, B22));

        // Calculate result submatrices

        int[][] C11 = addMatrix(subtractMatrix(addMatrix(M1, M4), M5), M7);

        int[][] C12 = addMatrix(M3, M5);

        int[][] C21 = addMatrix(M2, M4);

        int[][] C22 = addMatrix(subtractMatrix(subtractMatrix(M1, M2), M3), M6);

        // Combine result submatrices into final result matrix

        int[][] C = new int[n][n];

        combineMatrix(C, C11, C12, C21, C22);

        return C;

    }

    // Check if a number is a power of two

    private boolean isPowerOfTwo(int n) {

        return (n & (n - 1)) == 0;

    }

    // Divide matrix M into submatrices

    private void divideMatrix(int[][] M, int[][] C11, int[][] C12, int[][] C21, int[][] C22) {

        int n = M.length / 2;

        for (int i = 0; i < n; i++) {

            for (int j = 0; j < n; j++) {

                C11[i][j] = M[i][j];

                C12[i][j] = M[i][j + n];

                C21[i][j] = M[i + n][j];

                C22[i][j] = M[i + n][j + n];

            }

        }

    }

    // Add two matrices

    private int[][] addMatrix(int[][] A, int[][] B) {

        int n = A.length;

        int[][] C = new int[n][n];

        for (int i = 0; i < n; i++) {

            for (int j = 0; j < n; j++) {

                C[i][j] = A[i][j] + B[i][j];

            }

        }

        return C;

    }

    // Subtract two matrices

    private int[][] subtractMatrix(int[][] A, int[][] B) {

        int n = A.length;

        int[][] C = new int[n][n];

        for (int i = 0; i < n; i++) {

            for (int j = 0; j < n; j++) {

                C[i][j] = A[i][j] - B[i][j];

            }

        }

        return C;

    }

    // Combine result submatrices into final result matrix

    private void combineMatrix(int[][] C, int[][] C11, int[][] C12, int[][] C21, int[][] C22) {

        int n = C.length / 2;

        for (int i = 0; i < n; i++) {

            for (int j = 0; j < n; j++) {

                C[i][j] = C11[i][j];

                C[i][j + n] = C12[i][j];

                C[i + n][j] = C21[i][j];

                C[i + n][j + n] = C22[i][j];

            }

        }

    }

    public static void main(String[] args) {

        StrassenSquareMatrixMultiply matrixMultiplier = new StrassenSquareMatrixMultiply();

        // Example usage

        int[][] A = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}, {13, 14, 15, 16}};

        int[][] B = {{17, 18, 19, 20}, {21, 22, 23, 24}, {25, 26, 27, 28}, {29, 30, 31, 32}};

        int[][] C = matrixMultiplier.multiply(A, B);

        // Print result matrix C

        for (int[] row : C) {

            for (int cell : row) {

                System.out.print(cell + " ");

            }

            System.out.println();

        }

    }

}

Strassen Square Matrix Multiply optimization output:

A screenshot of a computer program

Description automatically generated

Everything works as intended!