A white paper with black text

Description automatically generated

A white paper with black text

Description automatically generated

MatrixMultiply code:

import java.util.Vector;

public class MatrixMultiply{

    public static Vector<Vector<Double>> matrix\_multiply(Vector<Vector<Double>> inputA, Vector<Vector<Double>> inputB){

        int aRows = inputA.size();

        int aCols = inputA.get(0).size();

        int bRows = inputB.size();

        int bCols = inputB.get(0).size();

        if (aCols != bRows) {

            throw new IllegalArgumentException("incompatible dimensions");

        }

        else{

            Vector<Vector<Double>> c = new Vector<Vector<Double>>();

            for (int i = 0; i < aRows; i++) {

                c.add(new Vector<Double>());

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

                    c.get(i).add(0.0);

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

                        c.get(i).set(j, c.get(i).get(j) + inputA.get(i).get(k) \* inputB.get(k).get(j));

                    }

                }

            }

            return c;

        }

    }

    public static void main(String[] args) {

        Vector<Vector<Double>> A = new Vector<Vector<Double>>();

        Vector<Vector<Double>> B = new Vector<Vector<Double>>();

        A.add(new Vector<Double>());

        A.add(new Vector<Double>());

        A.get(0).add(1.0);

        A.get(0).add(2.0);

        A.get(1).add(3.0);

        A.get(1).add(4.0);

        B.add(new Vector<Double>());

        B.add(new Vector<Double>());

        B.get(0).add(5.0);

        B.get(0).add(6.0);

        B.get(1).add(7.0);

        B.get(1).add(8.0);

        Vector<Vector<Double>> C = matrix\_multiply(A, B);

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

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

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

            }

            System.out.println();

        }

    }

}

Matrix Multiply output:

A screen shot of a computer program

Description automatically generated

Optimization: StrassenMatrixMultiply

public class StrassenMatrixMultiply {

    public static double[][] strassenMatrixMultiply(double[][] A, double[][] B) {

        int n = A.length;

        // Base case

        if (n <= 2) {

            double[][] C = new double[n][n];

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

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

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

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

                    }

                }

            }

            return C;

        } else {

            int newSize = n / 2;

            double[][] A11 = new double[newSize][newSize];

            double[][] A12 = new double[newSize][newSize];

            double[][] A21 = new double[newSize][newSize];

            double[][] A22 = new double[newSize][newSize];

            double[][] B11 = new double[newSize][newSize];

            double[][] B12 = new double[newSize][newSize];

            double[][] B21 = new double[newSize][newSize];

            double[][] B22 = new double[newSize][newSize];

            // Split matrices into quadrants

            splitMatrix(A, A11, 0, 0);

            splitMatrix(A, A12, 0, newSize);

            splitMatrix(A, A21, newSize, 0);

            splitMatrix(A, A22, newSize, newSize);

            splitMatrix(B, B11, 0, 0);

            splitMatrix(B, B12, 0, newSize);

            splitMatrix(B, B21, newSize, 0);

            splitMatrix(B, B22, newSize, newSize);

            // Calculate intermediate products using Strassen's Algorithm recursively

            double[][] M1 = strassenMatrixMultiply(addMatrix(A11, A22), addMatrix(B11, B22));

            double[][] M2 = strassenMatrixMultiply(addMatrix(A21, A22), B11);

            double[][] M3 = strassenMatrixMultiply(A11, subtractMatrix(B12, B22));

            double[][] M4 = strassenMatrixMultiply(A22, subtractMatrix(B21, B11));

            double[][] M5 = strassenMatrixMultiply(addMatrix(A11, A12), B22);

            double[][] M6 = strassenMatrixMultiply(subtractMatrix(A21, A11), addMatrix(B11, B12));

            double[][] M7 = strassenMatrixMultiply(subtractMatrix(A12, A22), addMatrix(B21, B22));

            // Calculate result quadrants

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

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

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

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

            // Combine result quadrants into the resulting matrix

            double[][] C = new double[n][n];

            joinMatrix(C11, C, 0, 0);

            joinMatrix(C12, C, 0, newSize);

            joinMatrix(C21, C, newSize, 0);

            joinMatrix(C22, C, newSize, newSize);

            return C;

        }

    }

    // Function to add two matrices

    public static double[][] addMatrix(double[][] A, double[][] B) {

        int n = A.length;

        double[][] C = new double[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;

    }

    // Function to subtract two matrices

    public static double[][] subtractMatrix(double[][] A, double[][] B) {

        int n = A.length;

        double[][] C = new double[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;

    }

    // Function to split matrix into quadrants

    public static void splitMatrix(double[][] P, double[][] C, int iB, int jB) {

        for (int i1 = 0, i2 = iB; i1 < C.length; i1++, i2++) {

            for (int j1 = 0, j2 = jB; j1 < C.length; j1++, j2++) {

                C[i1][j1] = P[i2][j2];

            }

        }

    }

    // Function to join quadrants into a single matrix

    public static void joinMatrix(double[][] C, double[][] P, int iB, int jB) {

        for (int i1 = 0, i2 = iB; i1 < C.length; i1++, i2++) {

            for (int j1 = 0, j2 = jB; j1 < C.length; j1++, j2++) {

                P[i2][j2] = C[i1][j1];

            }

        }

    }

    public static void main(String[] args) {

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

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

        double[][] C = strassenMatrixMultiply(A, B);

        // Printing the result

        System.out.println("Resultant Matrix C:");

        printMatrix(C);

    }

    // Function to print a matrix

    public static void printMatrix(double[][] matrix) {

        for (double[] row : matrix) {

            for (double element : row) {

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

            }

            System.out.println();

        }

    }

}

Optimization output:

A screenshot of a computer program

Description automatically generated

**ALL CODE RUNS PROPERLY!!**