**Name:** Aditya Jethani

**ROLL No:** 21BCP166

**DIV-**3 **G-**5

**DAA Assignment 5**

**AIM:** Matrix Multiplication Using Divide and Conquer.

**Theory:**

The divide and conquer algorithm is a problem-solving strategy that involves breaking down a large problem into smaller sub-problems that can be solved independently, and then combining the solutions to solve the larger problem. The basic steps of the divide and conquer algorithm are as follows:

1. Divide the problem into smaller sub-problems that are easier to solve.
2. Solve each sub-problem independently.
3. Combine the solutions of the sub-problems to solve the original problem.

This algorithm is commonly used in many computer science applications, such as sorting algorithms (such as Merge Sort and Quick Sort), searching algorithms (such as Binary Search), and matrix multiplication.

**CODE:**

import java.util.Arrays;

public class MatrixMultiplyDnC {

    public static void main(String[] args) throws IllegalArgumentException{

        // Test Case

        int[][] a = {

                {8,2,5,3, 5},

                {5,7,9,1, 5},

                {2,4,3,1, 5},

                {3,1,9,2, 5},

                {3,1,9,2, 5}

        };

        int[][] b = {

                {2,2,6,4, 8},

                {4,7,5,1, 8},

                {3,7,3,2, 8},

                {7,4,5,6, 8},

                {7,4,5,6, 8}

        };

        System.out.println("Matrix A =>");

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

            System.out.println(Arrays.toString(a[i]));

        }

        System.out.println();

        System.out.println("Matrix B =>");

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

            System.out.println(Arrays.toString(b[i]));

        }

        System.out.println();

        System.out.println("Matrix after Multiplication resultMatrix =>");

        int[][] resultMatrix = matrixMultiply(a,b);

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

            System.out.println(Arrays.toString(resultMatrix[i]));

        }

    }

    public static int[][] matrixMultiply(int[][] A, int[][] B) {

        int l1 = A.length, h1 = A[0].length, l2 = B.length, h2 = B[0].length;

        if (h1 != l2) {

            throw new IllegalArgumentException("Matrices cannot be multiplied");

        }

        int max = Math.max(Math.max(l1, h1), Math.max(l2, h2));

        float next2 = (float) (Math.log(max) / Math.log(2));

        int next = (int) Math.ceil(next2);

        int n = (int) Math.pow(2, next);

        int[][] a = new int[n][n];

        int[][] b = new int[n][n];

        for (int i = 0; i < l1; i++) {

            for (int j = 0; j < h1; j++) {

                a[i][j] = A[i][j];

            }

        }

        for (int i = 0; i < n; i++) {

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

                if(i < l2 && j < h2)

                    b[i][j] = B[i][j];

                else

                    b[i][j] = 0;

            }

        }

        return matrixMultiplyDivideConquer(a,b);

    }

    public static int[][] matrixMultiplyDivideConquer(int[][] A, int[][] B) {

        int n = A.length;

        // If the matrices are 1x1, just do a simple multiplication

        if (n == 1) {

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

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

            return C;

        }

        // Split matrices into quarters

        int size = n / 2;

        int[][] a11 = new int[size][size];

        int[][] a12 = new int[size][size];

        int[][] a21 = new int[size][size];

        int[][] a22 = new int[size][size];

        int[][] b11 = new int[size][size];

        int[][] b12 = new int[size][size];

        int[][] b21 = new int[size][size];

        int[][] b22 = new int[size][size];

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

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

                a11[i][j] = A[i][j];

                a12[i][j] = A[i][j + size];

                a21[i][j] = A[i + size][j];

                a22[i][j] = A[i + size][j + size];

                b11[i][j] = B[i][j];

                b12[i][j] = B[i][j + size];

                b21[i][j] = B[i + size][j];

                b22[i][j] = B[i + size][j + size];

            }

        }

        // Recursively compute products

        int[][] p1 = matrixMultiplyDivideConquer(a11, b11);

        int[][] p2 = matrixMultiplyDivideConquer(a12, b21);

        int[][] p3 = matrixMultiplyDivideConquer(a11, b12);

        int[][] p4 = matrixMultiplyDivideConquer(a12, b22);

        int[][] p5 = matrixMultiplyDivideConquer(a21, b11);

        int[][] p6 = matrixMultiplyDivideConquer(a22, b21);

        int[][] p7 = matrixMultiplyDivideConquer(a21, b12);

        int[][] p8 = matrixMultiplyDivideConquer(a22, b22);

    // Compute result matrix

        int[][] C = new int[n][n];

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

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

                C[i][j] = p1[i][j] + p2[i][j];

                C[i][j + size] = p3[i][j] + p4[i][j];

                C[i + size][j] = p5[i][j] + p6[i][j];

                C[i + size][j + size] = p7[i][j] + p8[i][j];

            }

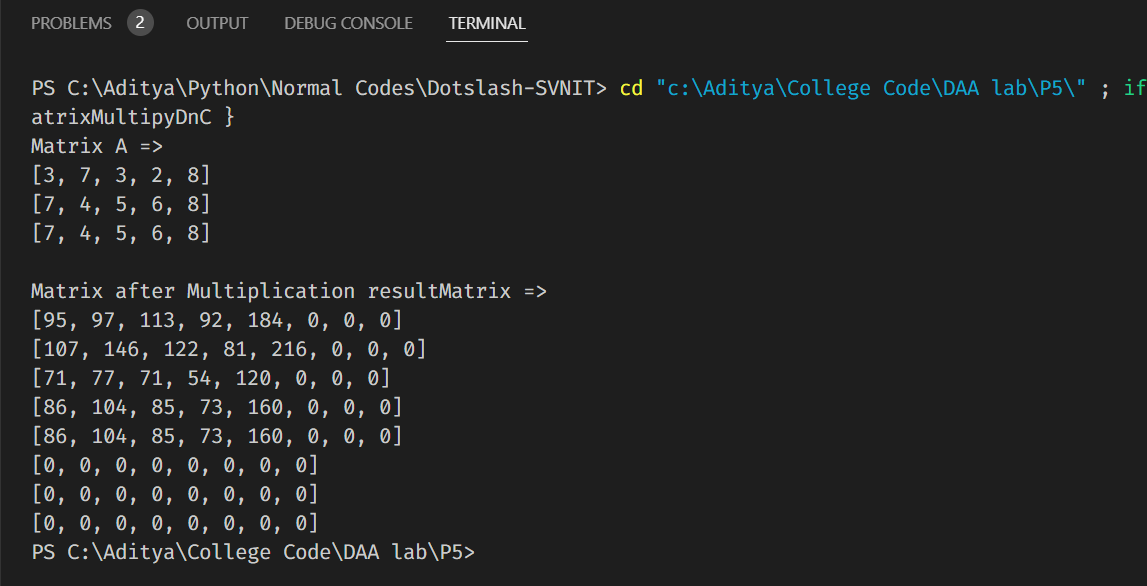
        }

        return C;

    }

}

**OUTPUT:**

****