Design & Analysis of Algorithm

Name - Hemal Chudasama
Roll No. - 21BCP152
Div - 3 , G - 5

Matrix Multiplication Using Divide and Conquer

- **Program :-**

```
import java.util.*;

class MM_DnC {

    static int ROW_1 = 4,COL_1 = 4, ROW_2 = 4, COL_2
= 4;

    public static void printMat(int[][] a, int r, int
c){
        for(int i=0;i<r;i++){
        for(int j=0;j<c;j++){
            System.out.print(a[i][j]+" ");
        }
        System.out.println("");
        }
        System.out.println("");
    }

    public static void print(String display, int[][]
matrix,int start_row, int start_column, int
end_row,int end_column)
    {
        System.out.println(display + " =>\n");
        for (int i = start_row; i <= end_row; i++) {
        for (int j = start_column; j <= end_column;
j++) {
            System.out.print(matrix[i][j]+" ");
        }
        System.out.println("");
        }
```

```
        System.out.println("");
    }

    public static void add_matrix(int[][]
matrix_A,int[][] matrix_B,int[][] matrix_C, int
split_index)
    {
        for (int i = 0; i < split_index; i++){
        for (int j = 0; j < split_index; j++){
            matrix_C[i][j] = matrix_A[i][j] +
matrix_B[i][j];
        }
        }
    }

    public static void initWithZeros(int a[][], int
r, int c){
        for(int i=0;i<r;i++){
        for(int j=0;j<c;j++){
            a[i][j]=0;
        }
        }
    }

    public static int[][] multiply_matrix(int[][]
matrix_A,int[][] matrix_B)
    {
        int col_1 = matrix_A[0].length;
        int row_1 = matrix_A.length;
        int col_2 = matrix_B[0].length;
        int row_2 = matrix_B.length;

        if (col_1 != row_2) {
        System.out.println("\nError: The number of
columns in Matrix A must be equal to the number of
rows in Matrix B\n");
        int temp[][] = new int[1][1];
        temp[0][0]=0;
        return temp;
        }
```

```
        int[] result_matrix_row = new int[col_2];
        Arrays.fill(result_matrix_row,0);
        int[][] result_matrix = new int[row_1]
[col_2];
        initWithZeros(result_matrix,row_1,col_2);

        if (col_1 == 1){
        result_matrix[0][0] = matrix_A[0][0] *
matrix_B[0][0];
        }else {
        int split_index = col_1 / 2;

        int[] row_vector = new int[split_index];
        Arrays.fill(row_vector,0);

        int[][] result_matrix_00 = new
int[split_index][split_index];
        int[][] result_matrix_01 = new
int[split_index][split_index];
        int[][] result_matrix_10 = new
int[split_index][split_index];
        int[][] result_matrix_11 = new
int[split_index][split_index];

initWithZeros(result_matrix_00,split_index,split_inde
x);

initWithZeros(result_matrix_01,split_index,split_inde
x);

initWithZeros(result_matrix_10,split_index,split_inde
x);

initWithZeros(result_matrix_11,split_index,split_inde
x);

        int[][] a00 = new int[split_index]
[split_index];
```

```
        int[][] a01 = new int[split_index]
[split_index];
        int[][] a10 = new int[split_index]
[split_index];
        int[][] a11 = new int[split_index]
[split_index];
        int[][] b00 = new int[split_index]
[split_index];
        int[][] b01 = new int[split_index]
[split_index];
        int[][] b10 = new int[split_index]
[split_index];
        int[][] b11 = new int[split_index]
[split_index];
        initWithZeros(a00,split_index,split_index);
        initWithZeros(a01,split_index,split_index);
        initWithZeros(a10,split_index,split_index);
        initWithZeros(a11,split_index,split_index);
        initWithZeros(b00,split_index,split_index);
        initWithZeros(b01,split_index,split_index);
        initWithZeros(b10,split_index,split_index);
        initWithZeros(b11,split_index,split_index);


        for (int i = 0; i < split_index; i++){
            for (int j = 0; j < split_index; j++) {
            a00[i][j] = matrix_A[i][j];
            a01[i][j] = matrix_A[i][j + split_index];
            a10[i][j] = matrix_A[split_index + i][j];
            a11[i][j] = matrix_A[i + split_index][j +
split_index];
            b00[i][j] = matrix_B[i][j];
            b01[i][j] = matrix_B[i][j + split_index];
            b10[i][j] = matrix_B[split_index + i][j];
            b11[i][j] = matrix_B[i + split_index][j +
split_index];
            }
        }
```

```
        add_matrix(multiply_matrix(a00,
b00),multiply_matrix(a01, b10),result_matrix_00,
split_index);
        add_matrix(multiply_matrix(a00,
b01),multiply_matrix(a01, b11),result_matrix_01,
split_index);
        add_matrix(multiply_matrix(a10,
b00),multiply_matrix(a11, b10),result_matrix_10,
split_index);
        add_matrix(multiply_matrix(a10,
b01),multiply_matrix(a11, b11),result_matrix_11,
split_index);

        for (int i = 0; i < split_index; i++){
            for (int j = 0; j < split_index; j++) {
            result_matrix[i][j] = result_matrix_00[i]
[j];
            result_matrix[i][j + split_index] =
result_matrix_01[i][j];
            result_matrix[split_index + i][j] =
result_matrix_10[i][j];
            result_matrix[i + split_index] [j +
split_index] = result_matrix_11[i][j];
            }
        }
        }
        return result_matrix;
    }

    public static void main (String[] args) {
        int[][] matrix_A = { { 1, 1, 1, 1 },
                             { 2, 2, 2, 2 },
                             { 3, 3, 3, 3 },
                             { 2, 2, 2, 2 } };

        System.out.println("Array A :-");
        printMat(matrix_A,4,4);

        int[][] matrix_B = { { 1, 1, 1, 1 },
                             { 2, 2, 2, 2 },
```

```
                                { 3, 3, 3, 3 },
                                { 2, 2, 2, 2 } };

        System.out.println("Array B :-");
        printMat(matrix_B,4,4);

        int[][] result_matrix =
multiply_matrix(matrix_A, matrix_B);

        System.out.println("Result Array :-");
        printMat(result_matrix,4,4);
    }
}
```

- **Output :-**

```
Array A :-
1 1 1 1
2 2 2 2
3 3 3 3
2 2 2 2

Array B :-
1 1 1 1
2 2 2 2
3 3 3 3
2 2 2 2

Result Array :-
8 8 8 8
16 16 16 16
24 24 24 24
16 16 16 16
```