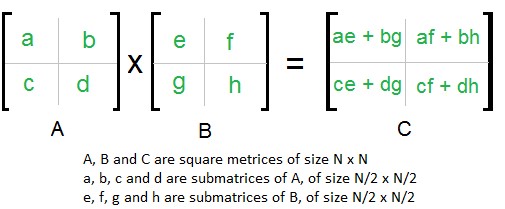
## Practical 3

### **Aim:** Write a program on Strassen's algorithm for matrix multiplication and analyze its complexity

**Theory:**

1. In linear algebra, the Strassen algorithm, named after Volker Strassen, is an algorithm for matrix multiplication.
2. It is faster than the standard matrix multiplication algorithm for large matrices, with a better asymptotic complexity, although the naive algorithm is often better for smaller matrices
3. Strassen’s Matrix multiplication can be performed only on square matrices where n is a power of 2. Order of both of the matrices are n × n.

**Example:**



**Algorithm:**

SQUARE-MATRIX-MULTIPLY

1. n = A.rows
2. let C be a new n X n matrix
3. for i = 1 to n
4. for j = 1 to n
5. cij = 0
6. for k = 1 to n
7. cij = cij + aik . bkj
8. return C

**Code:**

package daa.pracs;

import java.util.Scanner; /\*\* Class Strassen \*\*/

public class strassen

{

/\*\* Function to multiply matrices \*\*/ public int[][] multiply(int[][] A, int[][] B)

{ int n = A.length; int[][] R = new int[n][n]; /\*\* base case \*\*/ if (n == 1)

R[0][0] = A[0][0] \* B[0][0]; else

{ 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];

/\*\* Dividing matrix A into 4 halves \*\*/ split(A, A11, 0 , 0); split(A, A12, 0 , n/2); split(A, A21, n/2, 0); split(A, A22, n/2, n/2);

/\*\* Dividing matrix B into 4 halves \*\*/ split(B, B11, 0 , 0); split(B, B12, 0 , n/2); split(B, B21, n/2, 0); split(B, B22, n/2, n/2);

int [][] P1 = multiply(A11, sub(B12, B22)); int [][] P2 = multiply(add(A11, A12), B22);

int [][] P3 = multiply(add(A21, A22), B11); int [][] P4 = multiply(A22, sub(B21, B11)); int [][] P5 = multiply(add(A11, A22), add(B11, B22));

int [][] P6 = multiply(sub(A12, A22), add(B21, B22)); int [][] P7 = multiply(sub(A11, A21), add(B11, B12));

int [][] C11 = add(sub(add(P5,P4), P2), P6); int [][] C12 = add(P1, P2); int [][] C21 = add(P3, P4); int [][] C22 = add(P5,sub(P1,add(P3,P7)));

/\*\* join 4 halves into one result matrix \*\*/ join(C11, R, 0 , 0); join(C12, R, 0 , n/2); join(C21, R, n/2, 0); join(C22, R, n/2, n/2);

}

/\*\* return result \*\*/ return R;

}

/\*\* Funtion to sub two matrices \*\*/ public int[][] sub(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;

}

/\*\* Funtion to add two matrices \*\*/ public int[][] add(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;

}

/\*\* Funtion to split parent matrix into child matrices \*\*/ public void split(int[][] P, int[][] 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];

}

/\*\* Funtion to join child matrices intp parent matrix \*\*/ public void join(int[][] C, int[][] 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];

}

/\*\* Main function \*\*/ public static void main (String[] args)

{

Scanner scan = new Scanner(System.in); System.out.println("Strassen Multiplication Algorithm Test\n");

/\*\* Make an object of Strassen class \*\*/

strassen s = new strassen();

System.out.println("Enter order n :"); int N = scan.nextInt();

/\*\* Accept two 2d matrices \*\*/

System.out.println("Enter N order matrix 1\n"); int[][] A = new int[N][N]; for (int i = 0; i < N; i++) for (int j = 0; j < N; j++)

A[i][j] = scan.nextInt();

System.out.println("Enter N order matrix 2\n"); int[][] B = new int[N][N]; for (int i = 0; i < N; i++) for (int j = 0; j < N; j++)

B[i][j] = scan.nextInt();

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

System.out.println("\nProduct of matrices A and B : "); for (int i = 0; i < N; i++)

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

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

System.out.println();

}

System.out.println("Neeraj Appari T073");

}

}

**Output**:

Strassen Multiplication Algorithm Test

Enter order n :

2

Enter N order matrix 1

1 2

3 4

Enter N order matrix 2

4 3

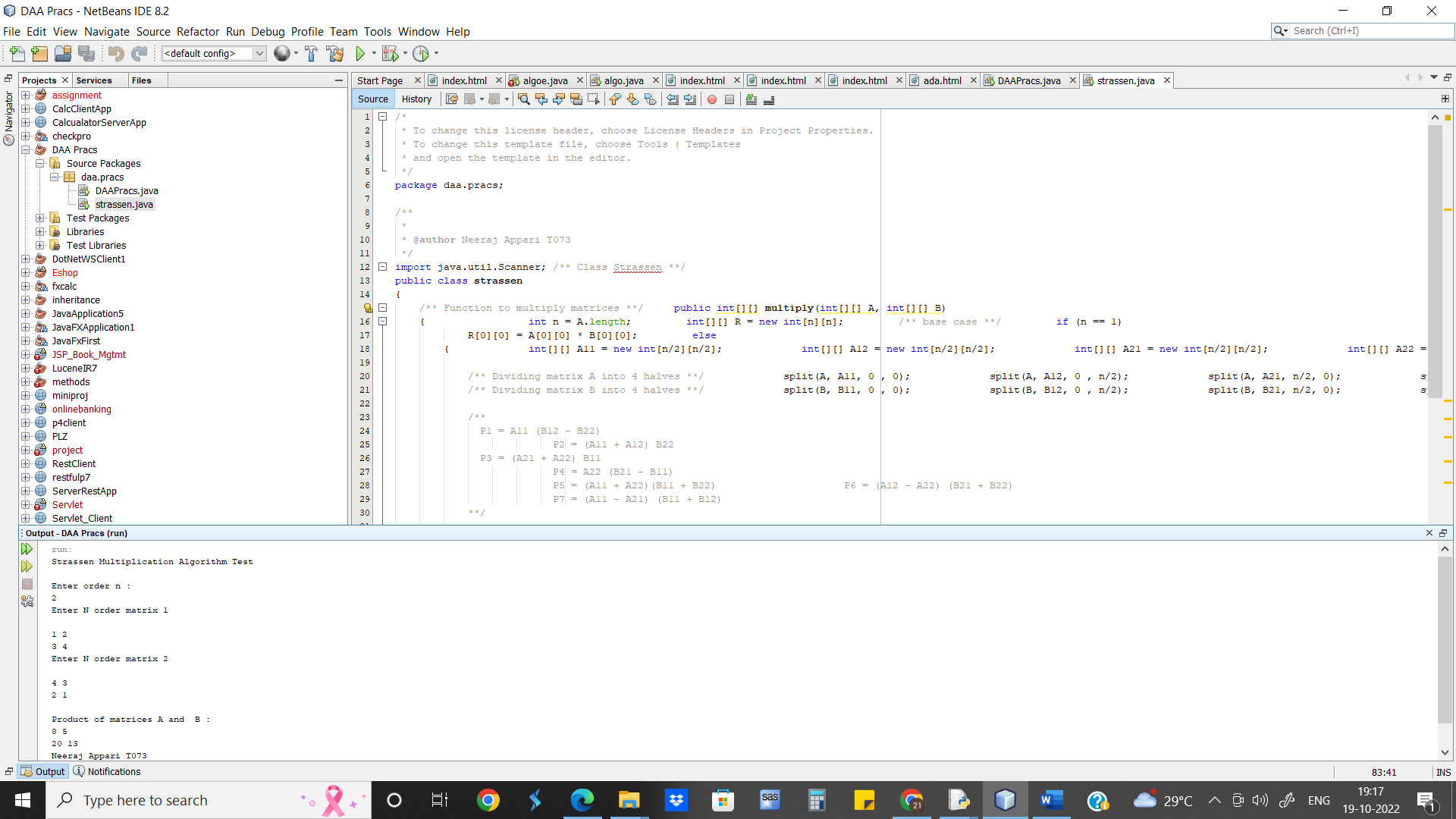
2 1

Product of matrices A and B :

8 5

20 13

Neeraj Appari T073



### **Runtime for Strassen’s algorithm is O(n2.81)**

**Conclusion:** Using Strassen’s Matrix multiplication algorithm, the time consumption can be improved a little bit.

The worst-case runtime complexity of Insertion Sort is O(n2.81).

Thus, for small matrix sizes Naïve method is preferred whereas when the size increases Strassen’s method is more preferred.