NAME:	Pratham Jain
UID:	2021300051- COMPS A (C-batch)
SUBJECT	DAA
EXPERIMENT NO :	3
DATE OF PERFORMANCE	27-02-23
DATE OF SUBMISSION	5-03-23
AIM:	To multiply two matrices using Strassen's multiplication.
PROBLEM STATEMENT 1:	**Using Strassen's multiplication method to find the product of two matrices.**
THEORY	# Strassen's Matrix Multiplication **Strassen** in 1969 gave an overview on how we can find the multiplication of two **2*2 dimension matrices by the brute-force algorithm**. But by using the divide and conquer technique the overall complexity for the multiplication of two matrices has been reduced. This happens by decreasing the total number of multiplications performed at the expense of a slight increase in the number of additions.

	Strassen has used some formulas for multiplying the two 2*2 dimension matrices where the number of multiplications is seven, additions and subtractions are is eighteen, and in brute force algorithm, there is eight number of multiplications and four addition. When the order **n** of matrix reaches infinity, the utility of Strassen's formula is shown by its asymptotic superiority. For example, let us consider two matrices **A** and **B** of **n*n** dimension, where **n** is a power of two. It can be observed that we can have four submatrices of order **n/2 * n/2** from **A**, **B,** and their product **C** where **C** is the resultant matrix of **A** and **B**. The procedure of Strassen's matrix multiplication Here is the procedure : 1. Divide a matrix of the order of 2*2 recursively until we get the matrix of order 2*2. 2. To carry out the multiplication of the 2*2 matrix, use the previous set of formulas. 3. Subtraction is also performed within these eight multiplications and four additions. 4. To find the final product or final matrix combine the result of two matrixes.

Formulas for Strassen's matrix multiplication. Following are the formulae that are to be used for matrix multiplication.

1. D1 = (a11 + a22) * (b11 + b22)
2. D2 = (a21 + a22)*b11
3. D3 = (b12 – b22)*a11
4. D4 = (b21 – b11)*a22
5. D5 = (a11 + a12)*b22
6. D6 = (a21 – a11) * (b11 + b12)
7. D7 = (a12 – a22) * (b21 + b22)

C00= d1 + d4 – d5 + d7
C01 = d3 + d5
C10 = d2 + d4
C11 = d1 + d3 – d2 – d6

Here, C00, C01, C10, and C11 are the elements of the 2*2 matrix.

Time Complexity of Strassen's Method

Addition and Subtraction of two matrices takes $O(N^2)$ time. So time complexity can be written as

$T(N) = 7T(N/2) + O(N^2)$

From Master's Theorem, time complexity of above method is

$O(N^{Log7})$ which is approximately $O(N^{2.8074})$

ALGORITHM	Algorithm Strass(n, x, y, z)
	
	begin
	If n = threshold then compute
	C = x * y is a conventional matrix.
	Else
	Partition a into four sub matrices a00, a01, a10, a11.
	Partition b into four sub matrices b00, b01, b10, b11.
	Strass (n/2, a00 + a11, b00 + b11, d1)
	Strass (n/2, a10 + a11, b00, d2)
	Strass (n/2, a00, b01 – b11, d3)
	Strass (n/2, a11, b10 – b00, d4)
	Strass (n/2, a00 + a01, b11, d5)
	Strass (n/2, a10 – a00, b00 + b11, d6)
	Strass (n/2, a01 – a11, b10 + b11, d7)
	
	C = d1+d4-d5+d7 d3+d5
	d2+d4 d1+d3-d2-d6
	
	end if
	return (C)
	end.
	
	For general divide & conquer:
	
	Algorithm STRESSEN_MAT_MUL (int *A, int *B, int *C, int n)
	// A and B are input matrices
	// C is the output matrix
	// All matrices are of size n x n
	
	if n == 1 then
	*C = *C + (*A) * (*B)

	else STRESSEN_MAT_MUL (A, B, C, n/4) STRESSEN_MAT_MUL (A, B + (n/4), C + (n/4), n/4) STRESSEN_MAT_MUL (A + 2 * (n/4), B, C + 2 * (n/4), n/4) STRESSEN_MAT_MUL (A + 2 * (n/4), B + (n/4), C + 3 * (n/4), n/4) STRESSEN_MAT_MUL (A + (n/4), B + 2 * (n/4), C, n/4) STRESSEN_MAT_MUL (A + (n/4), B + 3 * (n/4), C + (n/4), n/4) STRESSEN_MAT_MUL (A + 3 * (n/4), B + 2 * (n/4), C + 2 * (n/4), n/4) STRESSEN_MAT_MUL (A + 3 * (n/4), B + 3 * (n/4), C + 3 * (n/4), n/4) end

PROGRAM:	```#include <stdio.h>
	#include <time.h>
	void main()
	{
	int i, j;
	int a[2][2], b[2][2], c[2][2];
	int s[10], p[7];
	clock_t start, end;
	printf("\nEnter matrix A in order - a11, a12, a21, a22 :
	");
	for (i = 0; i < 2; i++)
	{
	for (j = 0; j < 2; j++)
	{
	scanf("%d", &a[i][j]);
	}
	}
	printf("\nEnter matrix B in order - b11, b12, b21, b22 :
	");
	for (i = 0; i < 2; i++)
	{
	for (j = 0; j < 2; j++)
	{
	scanf("%d", &b[i][j]);
	}
	}
	start = clock();
	s[0] = b[0][1] - b[1][1];
	s[1] = a[0][0] + a[0][1];
	s[2] = a[1][0] + a[1][1];
	s[3] = b[1][0] - b[0][0];
	s[4] = a[0][0] + a[1][1];```

```
    s[5] = b[0][0] + b[1][1];
    s[6] = a[0][1] - a[1][1];
    s[7] = b[1][0] + b[1][1];
    s[8] = a[0][0] - a[1][0];
    s[9] = b[0][0] + b[0][1];
    printf("\n");
    for (i = 0; i < 10; i++)
    {
        printf("\nS%d = %d", (i + 1), s[i]);
    }
    p[0] = s[0] * a[0][0];
    p[1] = s[1] * b[1][1];
    p[2] = s[2] * b[0][0];
    p[3] = s[3] * a[1][1];
    p[4] = s[4] * s[5];
    p[5] = s[6] * s[7];
    p[6] = s[8] * s[9];
    printf("\n");
    for (i = 0; i < 7; i++)
    {
        printf("\nP%d = %d", (i + 1), p[i]);
    }
    c[0][0] = p[4] + p[3] - p[1] + p[5];
    c[0][1] = p[0] + p[1];
    c[1][0] = p[2] + p[3];
    c[1][1] = p[4] + p[0] - p[2] - p[6];
    printf("\n\nMatrix A =");
    for (i = 0; i < 2; i++)
    {
        printf("\n");
        for (j = 0; j < 2; j++)
        {
            printf("%d\t", a[i][j]);
        }
    }
    printf("\n\nMatrix B =");
    for (i = 0; i < 2; i++)
    {
        printf("\n");
        for (j = 0; j < 2; j++)
        {
            printf("%d\t", b[i][j]);
        }
    }
```

```
    printf("\n\nMatrix C =");
    for (i = 0; i < 2; i++)
    {
        printf("\n");
        for (j = 0; j < 2; j++)
        {
            printf("%d\t", c[i][j]);
        }
    }
    printf("\n");
    end = clock();
    printf("Time taken = %lf\n", (double)(end - start) /
CLOCKS_PER_SEC);
}
```

General code:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int n, i, j;
    printf("Enter the dimensions of the matrix.\n");
    scanf("%d", &n);
    int A[n][n], B[n][n];
    int C[n][n];
    printf("Enter the contents of matrix 'A' \n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            scanf("%d", &A[i][j]);
        }
    }
    printf("Enter the contents of matrix 'B' \n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
```

```
            scanf("%d", &B[i][j]);
        }
    }
    int s1 = B[0][1] - B[1][1];
    int s2 = A[0][0] + A[0][1];
    int s3 = A[1][0] + A[1][1];
    int s4 = B[1][0] - B[0][0];
    int s5 = A[0][0] + A[1][1];
    int s6 = B[0][0] + B[1][1];
    int s7 = A[0][1] - A[1][1];
    int s8 = B[1][0] + B[1][1];
    int s9 = A[0][0] - A[1][0];
    int s10 = B[0][0] + B[0][1];

    int p1, p2, p3, p4, p5, p6, p7;
    p1 = A[0][0] * s1;
    p2 = s2 * B[1][1];
    p3 = s3 * B[0][0];
    p4 = s4 * A[1][1];
    p5 = s5 * s6;
    p6 = s7 * s8;
    p7 = s9 * s10;

    C[0][0] = p5 + p4 - p2 + p6;
    C[0][1] = p1 + p2;
    C[1][0] = p3 + p4;
    C[1][1] = p5 + p1 - p3 - p7;

    printf("Matrix A*B is \n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            printf("%d\t", C[i][j]);
        }
        printf("\n");
    }
}
```

RESULT (SNAPSHOT):

```
Enter matrix A in order - a11, a12, a21, a22 : 1 0 1 0

Enter matrix B in order - b11, b12, b21, b22 : 2 2 2 2


S1 = 0
S2 = 1
S3 = 1
S4 = 0
S5 = 1
S6 = 4
S7 = 0
S8 = 4
S9 = 0
S10 = 4

P1 = 0
P2 = 2
P3 = 2
P4 = 0
P5 = 4
P6 = 0
P7 = 0

Matrix A =
1       0
1       0

Matrix B =
2       2
2       2

Matrix C =
2       2
2       2
Time taken = 0.014000
```

CONCLUSION:	I have successfully learned how to successfully implement Strassen's multiplication method. And also learned the difference between normal and Strassen's multiplication.