NAME:	Shubham Vishwakarma
UID No.	2021700071
BRANCH:	S.Y CSE-DS
BATCH:	D
SUBJECT	Design and Analysis of Algorithms
EXPERIMENT No.	3
Date of Performance	28/02/2023
Date of Submission	02/03/2023

AIM:	**Experiment based on divide and conquer approach.**
Program 1	
PROBLEM STATEMENT :	Implement Strassen's Matrix Multiplication algorithm and compare it with standard matrix multiplication.

ALGORITHM/ THEORY:

Let us consider two matrices ***X*** and ***Y***. We want to calculate the resultant matrix ***Z*** by multiplying ***X*** and ***Y***.

Naïve Method

First, we will discuss naïve method and its complexity. Here, we are calculating ***Z* = *X* × *Y***. Using Naïve method, two matrices (***X*** and ***Y***) can be multiplied if the order of these matrices are ***p* × *q*** and ***q* × *r***. Following is the algorithm.

Algorithm: Matrix-Multiplication (X, Y, Z)

```
for i = 1 to p do
   for j = 1 to r do
      Z[i,j] := 0
      for k = 1 to q do
         Z[i,j] := Z[i,j] + X[i,k] × Y[k,j]
```

Complexity

Here, we assume that integer operations take ***O(1)*** time. There are three **for** loops in this algorithm and one is nested in other. Hence, the algorithm takes $O(n^3)$ time to execute.

Strassen's Matrix Multiplication Algorithm

In this context, using Strassen's Matrix multiplication algorithm, the time consumption can be improved a little bit.

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**.

Divide **X**, **Y** and **Z** into four (n/2)×(n/2) matrices as represented below −

$Z=[IKJL]$

$X=[ACBD]$ and $Y=[EGFH]$

Using Strassen's Algorithm compute the following −

$$\begin{gathered} M1:=(A+C)\times(E+F) \\ M2:=(B+D)\times(G+H) \\ M3:=(A-D)\times(E+H) \\ M4:=A\times(F-H) \\ M5:=(C+D)\times(E) \\ M6:=(A+B)\times(H) \\ M7:=D\times(G-E) \end{gathered}$$

Then,

$$\begin{gathered} I:=M2+M3-M6-M7 \\ J:=M4+M6 \\ K:=M5+M7 \\ L:=M1-M3-M4-M5 \end{gathered}$$

Analysis

$T(n)=\{c7xT(n2)+dxn2ifn=1otherwise$

where c and d are constants

Using this recurrence relation, we get $T(n)=O(nlog7)$

Hence, the complexity of Strassen's matrix multiplication algorithm is $O(nlog7)$

PROGRAM:

```
#include<stdio.h>

int main(){

     int a[2][2], b[2][2], c[2][2], i, j;
     int m1, m2, m3, m4 , m5, m6, m7;

printf("\nEnter the elements of first matrix 2x2\n\n");

     for(i = 0;i < 2; i++)
     {
          for(j = 0;j < 2; j++)
          {
               printf("Enter the element %d%d: ",i,j);
               scanf("%d", &a[i][j]);
          }
     }


printf("\nEnter the elements of second matrix 2x2\n\n");

     for(i = 0;i < 2; i++)
     {
          for(j = 0;j < 2; j++)
          {
               printf("Enter the element %d%d: ",i,j);
               scanf("%d", &b[i][j]);
          }
     }


printf("\nThe first matrix is\n\n");

     for(i = 0; i < 2; i++)
     {
          printf("|");
          printf("\t");
          for(j = 0; j < 2; j++)
          {
               printf("%d\t", a[i][j]);
          }
          printf("| ");
          printf("\n");
     }
```

	``c printf("\nThe second matrix is\n\n"); for(i = 0; i < 2; i++) { printf("|"); printf("\t"); for(j = 0; j < 2; j++) { printf("%d\t", b[i][j]); } printf("| "); printf("\n"); } m1= (a[0][0] + a[1][1]) * (b[0][0] + b[1][1]); m2= (a[1][0] + a[1][1]) * b[0][0]; m3= a[0][0] * (b[0][1] - b[1][1]); m4= a[1][1] * (b[1][0] - b[0][0]); m5= (a[0][0] + a[0][1]) * b[1][1]; m6= (a[1][0] - a[0][0]) * (b[0][0]+b[0][1]); m7= (a[0][1] - a[1][1]) * (b[1][0]+b[1][1]); c[0][0] = m1 + m4- m5 + m7; c[0][1] = m3 + m5; c[1][0] = m2 + m4; c[1][1] = m1 - m2 + m3 + m6; printf("\nAfter multiplication using Strassen's algorithm \n\n"); for(i = 0; i < 2; i++) { printf("|"); ```

```
            printf("\t");
            for(j = 0; j < 2; j++)
            {
                printf("%d\t", c[i][j]);
            }
            printf("| ");
            printf("\n");
        }



        return 0;

}
```

RESULT:

```
PS C:\Users\smsha\Desktop\SEM 4\DAA\Practicals\Exp3\output> & .\'strassen.exe'

Enter the elements of first matrix 2x2

Enter the element 00: 1
Enter the element 01: 3
Enter the element 10: 7
Enter the element 11: 5

Enter the elements of second matrix 2x2

Enter the element 00: 6
Enter the element 01: 8
Enter the element 10: 4
Enter the element 11: 2

The first matrix is

|       1       3       |
|       7       5       |

The second matrix is

|       6       8       |
|       4       2       |

After multiplication using Strassen's algorithm

|       18      14      |
|       62      66      |
PS C:\Users\smsha\Desktop\SEM 4\DAA\Practicals\Exp3\output>
```

Self-analysis:

$$A = \begin{bmatrix} 1 & 3 \\ 7 & 5 \end{bmatrix} \qquad B = \begin{bmatrix} 6 & 8 \\ 4 & 2 \end{bmatrix}$$

$a_{11} = 1$ $\quad b_{11} = 6$
$a_{12} = 3$ $\quad b_{12} = 8$
$a_{21} = 7$ $\quad b_{21} = 4$
$a_{22} = 5$ $\quad b_{22} = 2$

$S_1 = b_{12} - b_{22} = 6$
$S_2 = a_{11} + a_{12} = 4$
$S_3 = a_{21} + a_{22} = 12$
$S_4 = b_{21} - b_{11} = -2$
$S_5 = a_{11} + a_{22} = 6$
$S_6 = b_{11} + b_{22} = 8$
$S_7 = a_{12} - a_{22} = -2$
$S_8 = b_{21} + b_{22} = 6$
$S_9 = a_{11} - a_{21} = -6$
$S_{10} = b_{11} + b_{12} = 14$

सोमवार 12

$P_1 = a_{11} * S_1 = 6$
$P_2 = S_2 * b_{22} = 8$
$P_3 = S_3 * b_{11} = 72$
$P_4 = a_{22} * S_4 = -10$
$P_5 = S_5 * S_6 = 48$
$P_6 = S_7 * S_8 = -12$
$P_7 = S_9 * S_{10} = \cancel{48}\ -84$

$C_{11} = p_5 + p_4 - p_2 + p_6 = 18$
$C_{12} = p_1 + p_2 = 14$
$C_{21} = p_3 + p_4 = 62$
$C_{22} = p_5 + p_1 - p_3 - p_7 = 66$

Result matrix C is

$$C = \begin{bmatrix} 18 & 14 \\ 62 & 66 \end{bmatrix}$$

CONCLUSION :	Strassen's Matrix Multiplication, SMM, is used to multiply two matrices, and it is better than Native matrix multiplication. due to the fact that SMM's has a complexity of around $n^{2.81}$whereas usual multiplication's complexity is n^3. The reason for this is because the number of operations required in SMM is less than in usual multiplication. While usual multiplication requires 8 multiplications and 4 additions SMM requires 7 multiplications and 18 additions.