Experiment No.	3
Aim	Implement Strassen's Matrix Multiplication
Name	Aarush Kinhikar
UID No.	2021300063
Class & Division	SE Comps A, Batch D

Theory:

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.

$$Z = \begin{bmatrix} I & J \\ K & L \end{bmatrix} \qquad X = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \text{ and } \qquad Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$$

Using Strassen's Algorithm compute the following −

M1=(A+C)×(E+F)
M2=(B+D)×(G+H)
M3=(A−D)×(E+H)
M4=A×(F−H)
M5=(C+D)×(E)
M6=(A+B)×(H)
M7=D×(G−E)

Then,
I=M2+M3−M6−M7
J=M4+M6
K=M5+M7
L=M1−M3−M4−M5

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 $O(N^{2.8074})$

Generally, Strassen's Method is not preferred for practical applications for following reasons: -

- The constants used in Strassen's method are high and for a typical application Naive method works better.
- For Sparse matrices, there are better methods especially designed for them.
- The submatrices in recursion take extra space.
- Because of the limited precision of computer arithmetic on non-integer values, larger errors accumulate in Strassen's algorithm than in Naive Method

Algorithm:

1. Divide matrix X and matrix Y in 4 sub-matrices of size N/2 x N/2
2. Calculate the 7 matrix multiplications recursively.
3. Compute the submatrices of Z.
4. Combine these submatrices into our new matrix Z.

Program:

```c
#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("Enter the 4 elements of first matrix:\n");
  for(i = 0;i < 2; i++)
      for(j = 0;j < 2; j++)
           scanf("%d", &a[i][j]);

  printf("Enter the 4 elements of second matrix:\n");
  for(i = 0; i < 2; i++)
      for(j = 0;j < 2; j++)
           scanf("%d", &b[i][j]);

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

  printf("\nThe second matrix is");
  for(i = 0;i < 2; i++){
      printf("\n");
      for(j = 0;j < 2; j++)
           printf("%d\t", b[i][j]);
  }

  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");
   for(i = 0; i < 2 ; i++){
```

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

  return 0;
}
```

Results:

```
Enter the 4 elements of first matrix:
8271 1323
7132 1323
Enter the 4 elements of second matrix:
5461 1324
3521 6112

The first matrix is
8271    1323
7132    1323
The second matrix is
5461    1324
3521    6112
After multiplication using Strassen's algorithm
49826214  19036980
43606135  17528944
PS D:\DAA Experiments\Experiment 3>
```

Conclusion:

I have understood the theory and working of Strassen's Matrix Multiplication. It is evident from the steps above that it takes less time than traditional multiplication method as it divide the problem into seven subproblems instead of eight.