DAA exp-3

Name: Deepika Trivedi
UID: 2021700069
Batch: D-4

AIM: Implement divide and conquer technique.

Problem Statement: Implement Strassen's Matrix Multiplication algorithm and compare it with standard matrix multiplication.

Theory:

Divide and Conquer :

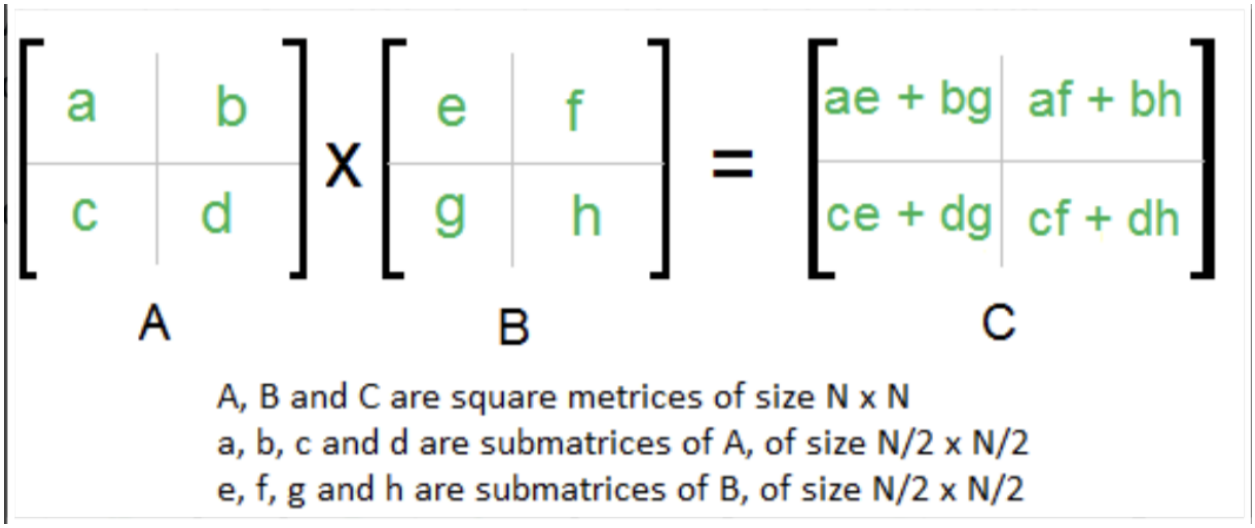
Following is simple Divide and Conquer method to multiply two square matrices.

1. Divide matrices A and B in 4 sub-matrices of size N/2 x N/2 as shown in the below diagram.
2. Calculate following values recursively. ae + bg, af + bh, ce + dg and cf + dh.

Strassen's Algorithm is an efficient algorithm to multiply two matrices. A simple method to multiply two matrices needs 3 nested loops and is O(n^3). Strassen's algorithm multiplies two matrices in O(n^2.8974) time.

T(N) = 7T(N/2) + O(N2)

From Master's Theorem, time complexity of above method is O(NLog7) which is approximately O(N2.8074)


a b
c d
x
e f
g h
=
ae + bg af + bh
ce + dg cf + dh
A
B
C
A, B and C are square metrices of size N x N
a, b, c and d are submatrices of A, of size N/2 x N/2
e, f, g and h are submatrices of B, of size N/2 x N/2

Code:

```c
/*
C code of two 2 by 2 matrix multiplication using Strassen's algorithm
*/
#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: ");
  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: ");
  for(i = 0; i < 2; i++)
    for(j = 0;j < 2; j++)
      scanf("%d", &b[i][j]);

  printf("\nThe first matrix is\n");
  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\n");
  for(i = 0;i < 2; i++){
```

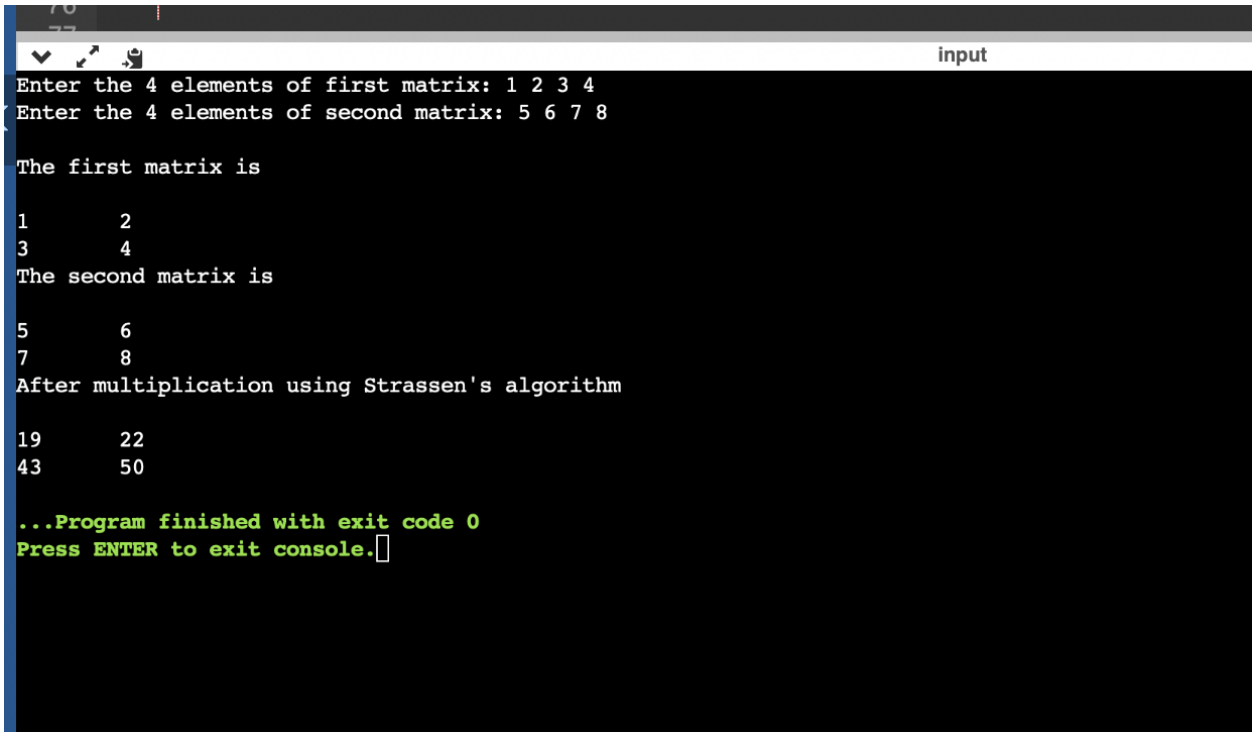
```
for(i = 0;i < 2; i++){
    printf("\n");
    for(j = 0;j < 2; j++)
        printf("%d\t", b[i][j]);
}

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

OUTPUT:



Manual Example:

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

$$C = A \times B$$

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

$$= \begin{bmatrix} 1\times5+2\times7 & 2\times8+1\times6 \\ 3\times5+4\times7 & 3\times6+4\times8 \end{bmatrix}$$

$$= \begin{bmatrix} 5+14 & 16+6 \\ 15+28 & 18+32 \end{bmatrix}$$

$$= \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

CONCLUSION: We used Strassen's Matrix Multiplication but without recursive calls and then we have compared the logic with the Standard Matrix Multiplication logic. On comparison we found out that Strassen's is better than standard method for multiplication of square matrices.