DESIGN AND ANALYSIS OF ALGORITHMS

EXPERIMENT 3

OM CHANDRA
CSE DS D1 BATCH
UID: 2021700014

AIM:

EXPERIMENT BASED ON DIVIDE AND CONQUER APPROACH TO IMPLEMENT STRASSEN'S MATRIX MULTIPLICATION.

THEORY:

The idea of Strassen's method is to reduce the number of recursive calls to 7. Strassen's method is similar to above simple divide and conquer method in the sense that this method also divide matrices to sub-matrices of size N/2 x N/2 as shown in the above diagram, but in Strassen's method, the four sub-matrices of result are calculated using following formulae.

$$p1 = a(f - h) \qquad p2 = (a + b)h$$
$$p3 = (c + d)e \qquad p4 = d(g - e)$$
$$p5 = (a + d)(e + h) \qquad p6 = (b - d)(g + h)$$
$$p7 = (a - c)(e + f)$$

The A x B can be calculated using above seven multiplications.
Following are values of four sub-matrices of result C

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} p5 + p4 - p2 + p6 & p1 + p2 \\ p3 + p4 & p1 + p5 - p3 - p7 \end{bmatrix}$$

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
p1, p2, p3, p4, p5, p6 and p7 are submatrices of size N/2 x N/2

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.

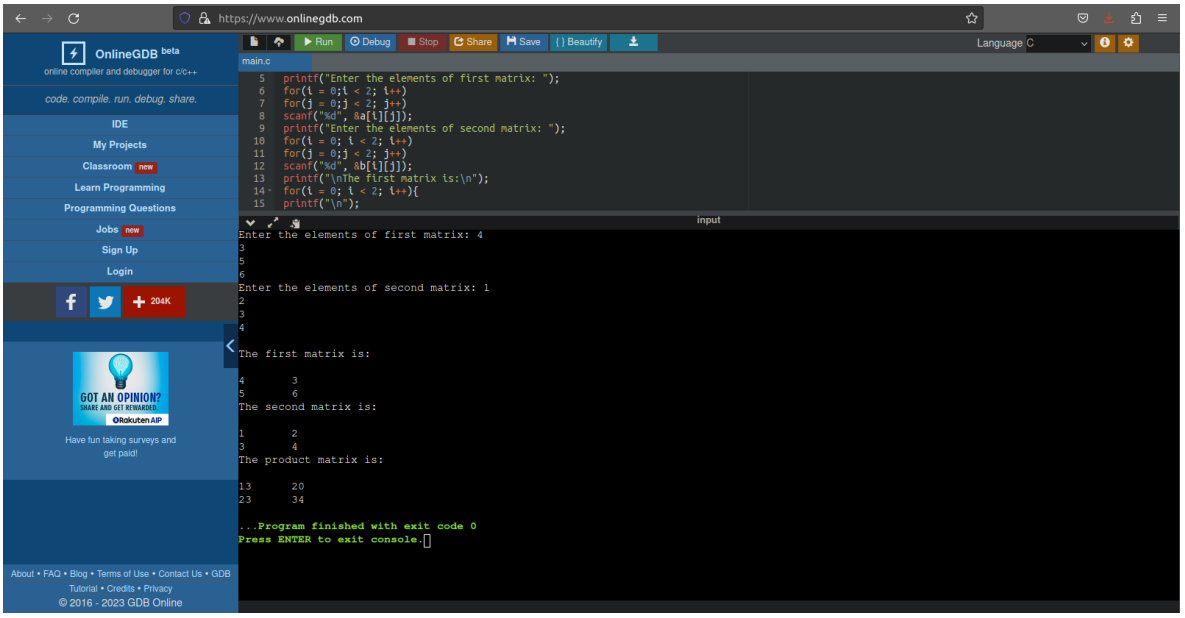
1. The constants used in Strassen's method are high and for a typical application Naive method works better.
2. For Sparse matrices, there are better methods especially designed for them.
3. The submatrices in recursion take extra space.
4. Because of the limited precision of computer arithmetic on noninteger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

CODE:

```
#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 elements of first matrix: ");
 for(i = 0;i < 2; i++)
 for(j = 0;j < 2; j++)
 scanf("%d", &a[i][j]);
 printf("Enter the 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++){
 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("\nThe product matrix is: \n");
 for(i = 0; i < 2 ; i++){
 printf("\n");
 for(j = 0;j < 2; j++)
 printf("%d\t", c[i][j]);
 }
 return 0;
}
```

OUTPUT:



CONCLUSION:

From this experiment I learnt how to apply divide and conquer approach to multiply two matrices using Strassen's matrix multiplication algorithm.