Sardar Patel Institute of Technology

an's Campus, Munshi Nagar, Andheri (West), Mumbai-400058, India
(Autonomous College Affiliated to University of Mumbai)

NAME	CHINMAY JADHAV
UID	2021300046
BATCH	C
SUBJECT	DAA
EXPERIMENT NO	3
DATE OF PERFORMANCE	27-02-2023
DATE OF SUBMISSION	06-03-2023
AIM	To understand and implement Strassen's Matrix Multiplication.
THEORY	Given two square matrices A and B of size n x n each, find their multiplication matrix. Naive Method takes the Time Complexity of O(N3). Divide and Conquer : Following is a 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. Simple Divide and Conquer also leads to O(N3), can there be

	a better way? In the above divide and conquer method, the main component for high time complexity is 8 recursive calls. 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. Time Complexity of Strassen’s Method Addition and Subtraction of two matrices takes O(N2) time. So time complexity can be written as T(N) = 7T(N/2) + O(N2) Generally Strassen’s Method is not preferred for practical applications for the 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. Start
2. Declare two matrices A and B and take the values from the user.
3. Find S1 to S10 using provided formulae.
4. Find P1 to P7 using provided formulae.
5. Find the elements of matrix C which is the multiplication of A and B.
6. Print the result.

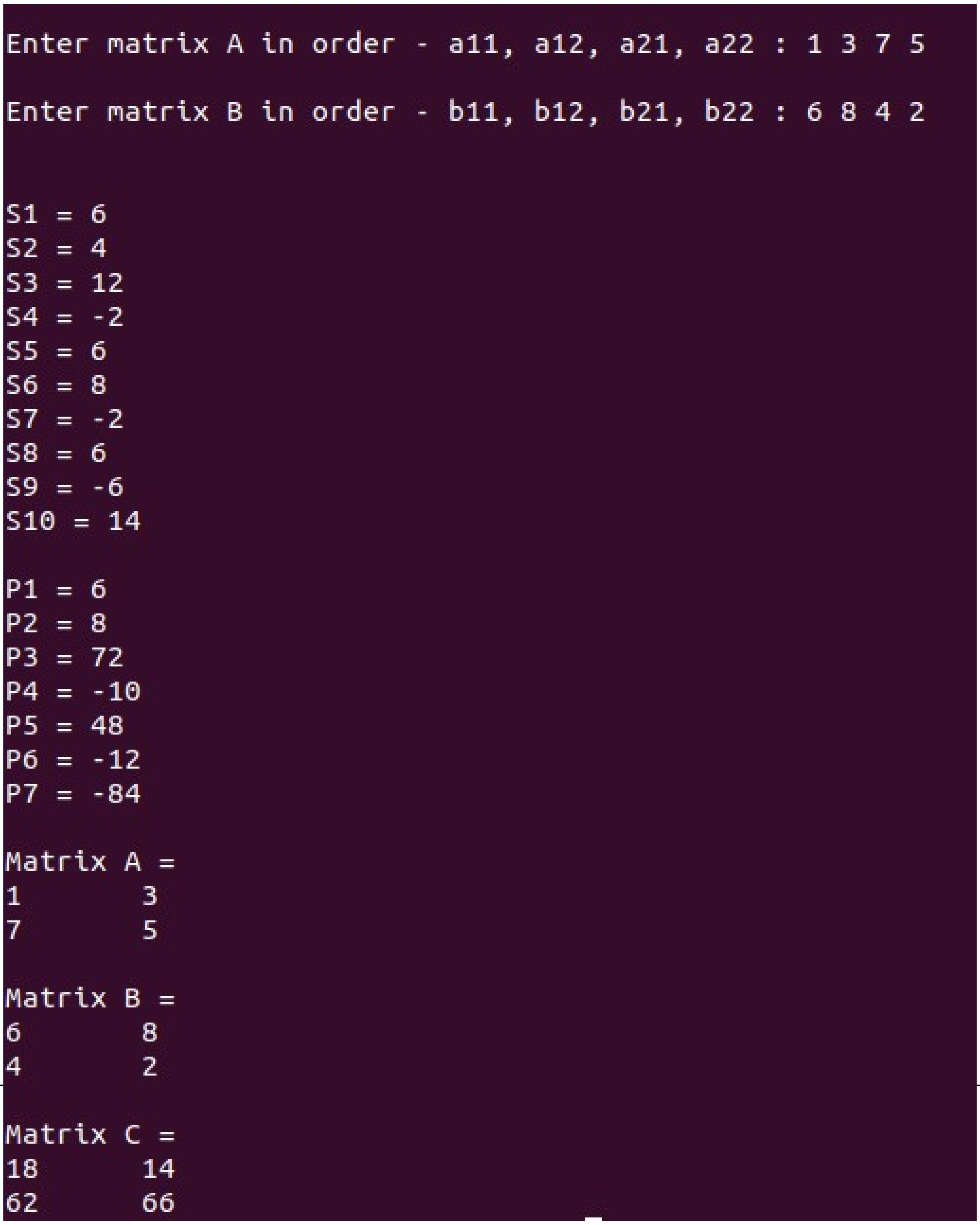
PROGRAM

```
#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]);
                }
        }
```

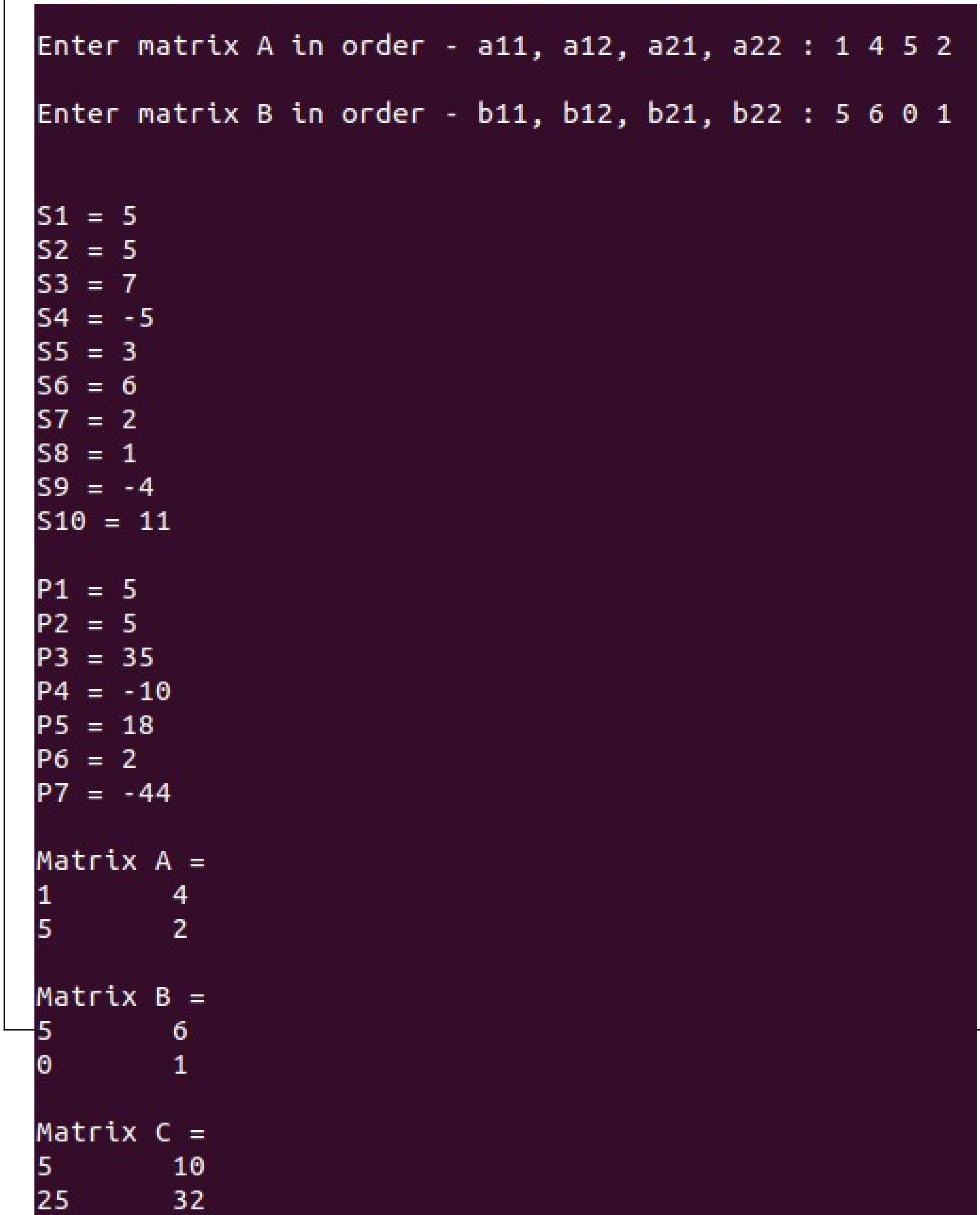
```
    printf("\nEnter matrix B in order - b11, b12, b21, b22 : ");
    for(i=0;i<2;i++)
    {
        for(j=0;j<2;j++)
        {
            scanf("%d",&b[i][j]);
        }
    }
    start=clock();
    s[0]=b[0][1]-b[1][1];
    s[1]=a[0][0]+a[0][1];
    s[2]=a[1][0]+a[1][1];
    s[3]=b[1][0]-b[0][0];
    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);
}
```

RESULT (SNAPSHOT):



```
Enter matrix A in order - a11, a12, a21, a22 : 1 3 7 5

Enter matrix B in order - b11, b12, b21, b22 : 6 8 4 2


S1 = 6
S2 = 4
S3 = 12
S4 = -2
S5 = 6
S6 = 8
S7 = -2
S8 = 6
S9 = -6
S10 = 14

P1 = 6
P2 = 8
P3 = 72
P4 = -10
P5 = 48
P6 = -12
P7 = -84

Matrix A =
1       3
7       5

Matrix B =
6       8
4       2

Matrix C =
18      14
62      66
```



```
Enter matrix A in order - a11, a12, a21, a22 : 1 4 5 2

Enter matrix B in order - b11, b12, b21, b22 : 5 6 0 1


S1 = 5
S2 = 5
S3 = 7
S4 = -5
S5 = 3
S6 = 6
S7 = 2
S8 = 1
S9 = -4
S10 = 11

P1 = 5
P2 = 5
P3 = 35
P4 = -10
P5 = 18
P6 = 2
P7 = -44

Matrix A =
1       4
5       2

Matrix B =
5       6
0       1

Matrix C =
5       10
25      32
```

CONCLUSION :

n conclusion, Strassen's Matrix Multiplication is a highly efficient algorithm for multiplying matrices. It reduces the number of multiplications required to multiply two matrices from 8 to 7, which may not seem like a big improvement, but can significantly reduce the overall computational complexity of the algorithm. Strassen's algorithm is particularly useful for multiplying large matrices, where the standard matrix multiplication algorithm may be too slow.