Bhartiya Vidya Bhavan's

Sardar Patel Institute of Technology

Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India

(Autonomous College Affiliated to University of Mumbai)

NAME:	Vaishnavi Bhagawan Borkar
UID:	2021300016
SUBJECT	Data Analysis and Algorithm
EXPERIMENT NO:	Experiment 3
DATE OF PERFORMANCE	2/3/23
AIM:	To implement Strassen's Matrix Multiplication algorithm. (Experiment on recurrence relation)
THEORY:	**Strassen's Matrix multiplication:** Usually, to multiply 2x2 matrix we need 8 multiplications. In Strassen's Matrix Multiplication this can be done in 7 multiplications. When recursively applied, Strassen's Matrix multiplication performs better than normal matrix multiplication. The procedure of Strassen's matrix multiplication Here is the procedure: 1 Divide a matrix of the order of 2*2 recursively until we get the matrix of order 2*2. To carry out the multiplication of the 2*2 matrix, use the previous set of formulas. 2 Subtraction is also performed within these eight multiplications and four additions. 3 To find the final product or final matrix combine the result of two matrixes. Strassen has used some formulas for multiplying the two 2*2-

dimension matrices where the number of multiplications is seven.

S1 = B12 – B21

S2 = A11 – A12

S3 = A21 – A22

S4 = B21 – B11

S5 = A11 + A22

S6 = B11 + B22

S7 = A12 – A22

S8 = B21 + B22

S9 = A11 – A21

S10 = B11 + B12

P1 = A11. S1

P2 = S2. B22

P3 = S3. B11

P4 = A22. S4

P5 = S5. S6

P6 = S7. S8

P7 = S9. S10

C11 = P5 + P4 – P2 + P6

C12 = P1 + P2

C21 = P3 + P4

C22 = P5 + P1 – P3 – P7

Time Complexity of Strassen's Method:

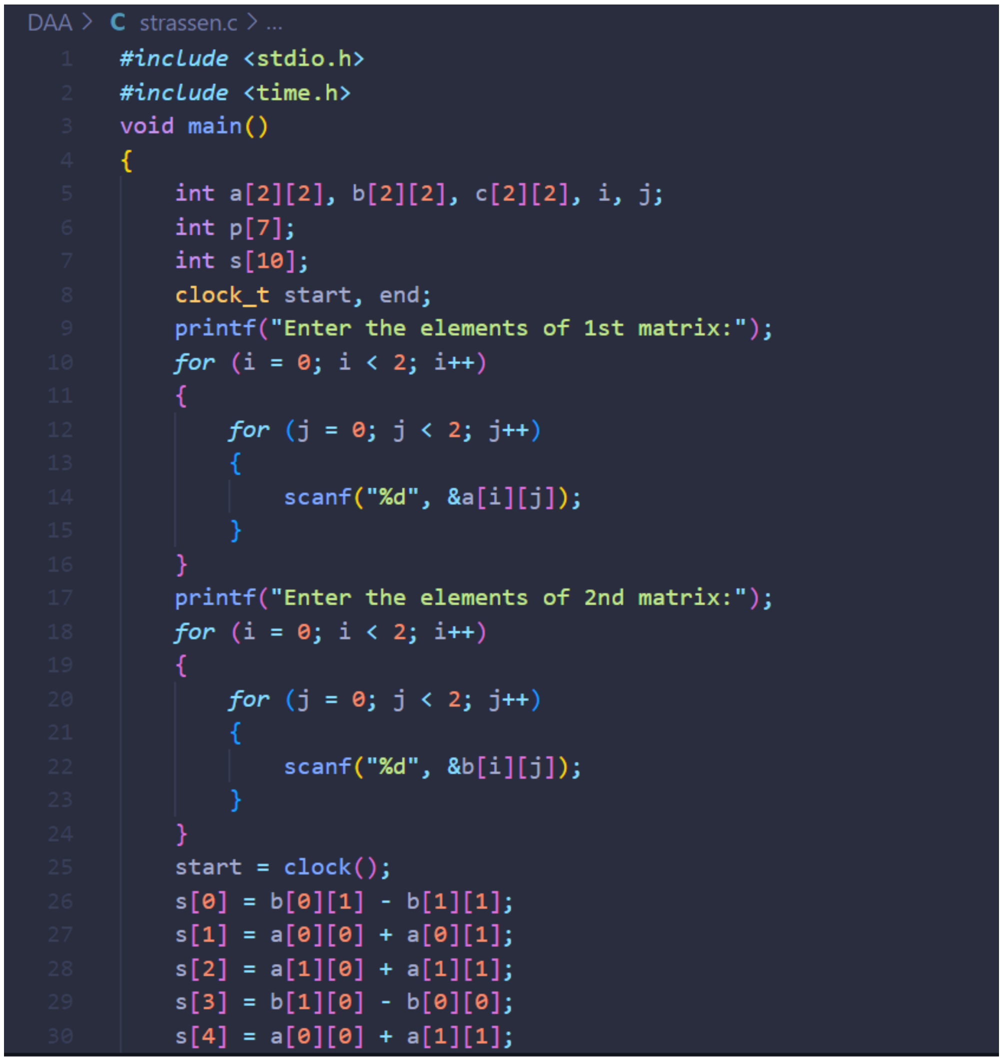
Addition and Subtraction of two matrices takes O(N2) time. So time complexity can be written as

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

Theoretically speaking, Naïve method has a time complexity of O(n 3) while Strassen's method has a time complexity of O(n

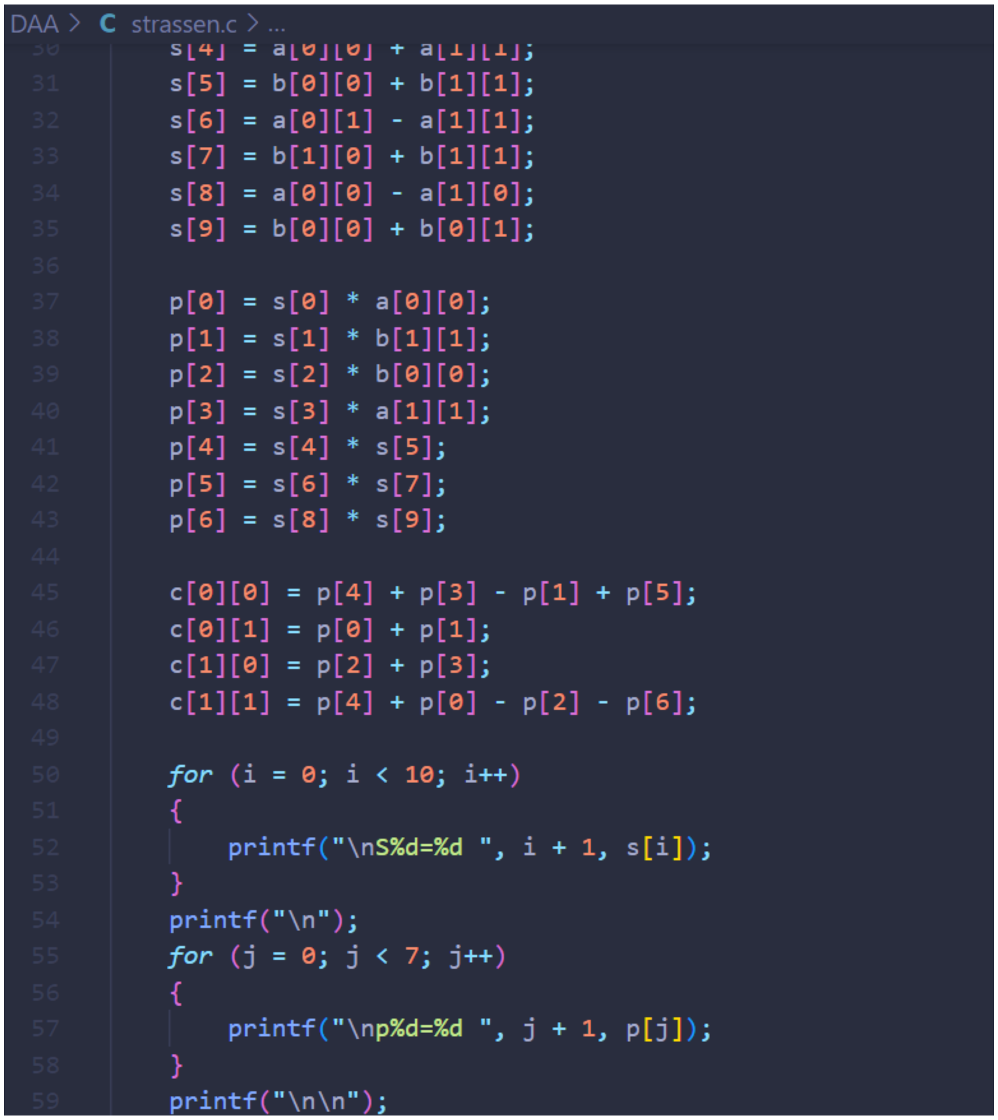
	log2 7) which approximates to O(n 2.808). Generally, Strassen's Method is not preferred for practical applications for the following reasons. The constants used in Strassen's method are high and for a typical application Naive method works better. For Sparse matrices, there are better methods especially 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:	For Strassen's Matrix Multiplication: 1. Start. 2. Calculate the order of input matrices in the form of 2 n that will be able to accommodate the provided elements. 3. Partition the input matrices of order 2 n into four matrices of order 2 n−1. 4. Calculate the required seven matrices M1 to M7 by recursively performing Strassen's matrix multiplication. 5. Obtain the result partitions through addition and subtraction of M matrices. 6. Combine the result partitions to get the final product matrix. 7. End.

PROGRAM:



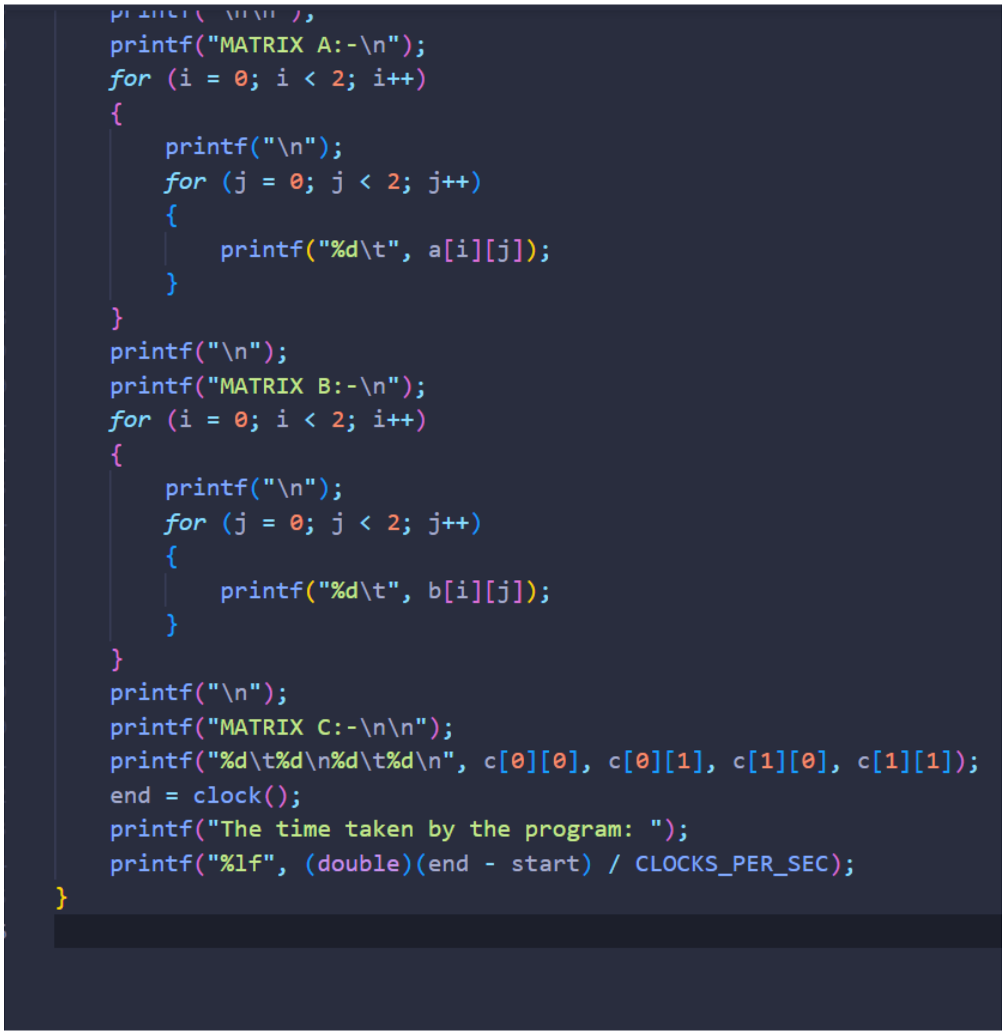
DAA > C strassen.c > ...

```c
#include <stdio.h>
#include <time.h>
void main()
{
    int a[2][2], b[2][2], c[2][2], i, j;
    int p[7];
    int s[10];
    clock_t start, end;
    printf("Enter the elements of 1st matrix:");
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 2; j++)
        {
            scanf("%d", &a[i][j]);
        }
    }
    printf("Enter the elements of 2nd matrix:");
    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];
```



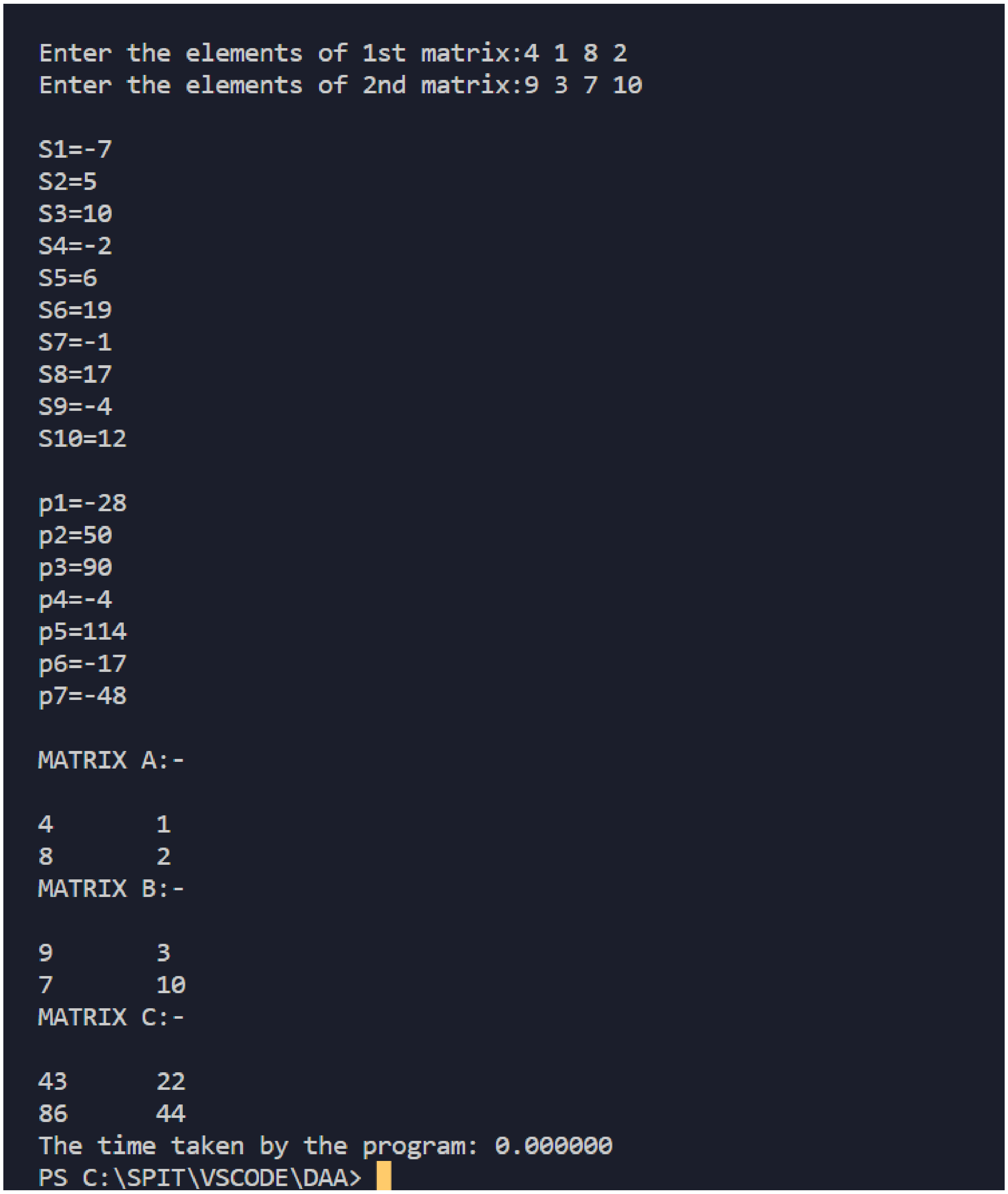
DAA > C strassen.c > ...

```c
    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];

    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];

    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];

    for (i = 0; i < 10; i++)
    {
        printf("\nS%d=%d ", i + 1, s[i]);
    }
    printf("\n");
    for (j = 0; j < 7; j++)
    {
        printf("\np%d=%d ", j + 1, p[j]);
    }
    printf("\n\n");
```



```
    printf("\n\n");
    printf("MATRIX A:-\n");
    for (i = 0; i < 2; i++)
    {
        printf("\n");
        for (j = 0; j < 2; j++)
        {
            printf("%d\t", a[i][j]);
        }
    }
    printf("\n");
    printf("MATRIX B:-\n");
    for (i = 0; i < 2; i++)
    {
        printf("\n");
        for (j = 0; j < 2; j++)
        {
            printf("%d\t", b[i][j]);
        }
    }
    printf("\n");
    printf("MATRIX C:-\n\n");
    printf("%d\t%d\n%d\t%d\n", c[0][0], c[0][1], c[1][0], c[1][1]);
    end = clock();
    printf("The time taken by the program: ");
    printf("%lf", (double)(end - start) / CLOCKS_PER_SEC);
}
```

RESULT:



```
Enter the elements of 1st matrix:4 1 8 2
Enter the elements of 2nd matrix:9 3 7 10

S1=-7
S2=5
S3=10
S4=-2
S5=6
S6=19
S7=-1
S8=17
S9=-4
S10=12

p1=-28
p2=50
p3=90
p4=-4
p5=114
p6=-17
p7=-48

MATRIX A:-

4       1
8       2
MATRIX B:-

9       3
7       10
MATRIX C:-

43      22
86      44
The time taken by the program: 0.000000
PS C:\SPIT\VSCODE\DAA>
```

CONCLUSION:	By performing this experiment, I was able to understand Strassen's Matrix Multiplication algorithm. I was also able to implement the same and compare it with Naïve method of matrix multiplication.