Bharatiya Vidya Bhavan's

SARDAR PATEL INSTITUTE OF TECHNOLOGY

(Autonomous Institute Affiliated to University of Mumbai)

Munshi Nagar, Andheri (W), Mumbai – 400 058.

Name – Advika Kharat
UID – 2021300060
Subject – DAA
Class – SE Comps A

Experiment No. 3

Aim – Strassen's Matrix Multiplication

Details –

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.

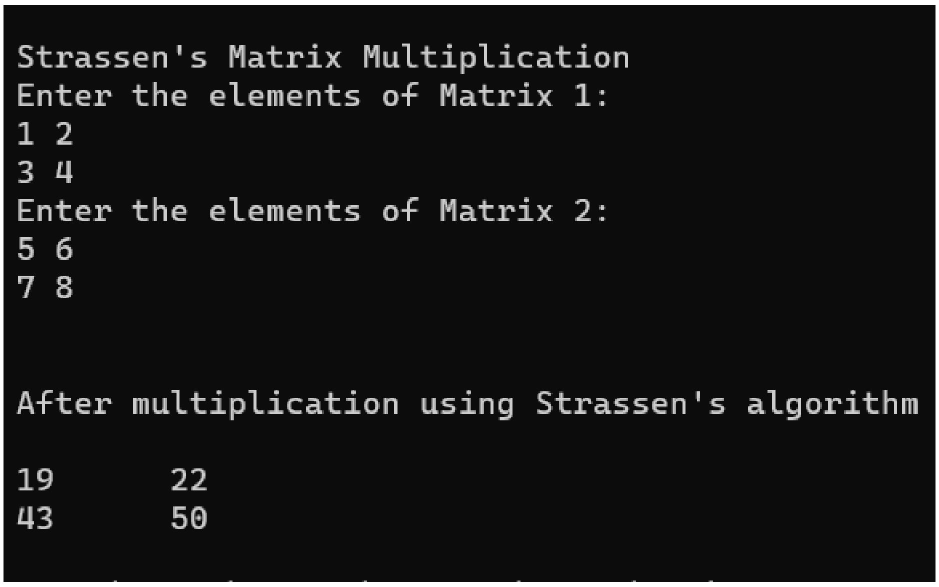
The idea of Strassen's method is to reduce the number of recursive calls to 7. Strassen's method is similar to above simple divide and conquer method in the sense that this method also divide matrices to sub-matrices of size N/2 x N/2 as shown in the above diagram, but in Strassen's method, the four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

Code:

```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("\nStrassen's Matrix Multiplication\n");
    printf("Enter the elements of Matrix 1: \n");
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 2; j++)
        {
            scanf("%d", &a[i][j]);
        }
    }

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

    return 0;
}
```

Output:



```
Strassen's Matrix Multiplication
Enter the elements of Matrix 1:
1 2
3 4
Enter the elements of Matrix 2:
5 6
7 8


After multiplication using Strassen's algorithm

19      22
43      50
```

Conclusion : Hence, I've come to the conclusion that normal matrix multiplication for a 2x2 matrix takes O(n^3) and 8 multiplications whereas Strassen's method takes a little lesser time than normal multiplication method and only 7 multiplications.