Suraj Iyer
2021300045
SE Comps A
Batch C

DAA Experiment 3

Aim - Strassen's Matrix Multiplication (2x2)

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 noninteger values, larger errors accumulate in Strassen’s algorithm than in Naive Method.

Code -

```
#include <stdio.h>

int main () {

        // Strassen’s Matrix Multiplication (2x2)

        // initializing the matrices
        printf("Consider two 2x2 matrices named X and Y\n");
        printf("Enter the 4 elements of X:\n");
        int A, B, C, D;
        scanf("%d %d %d %d", &A, &B, &C, &D);
        printf("Enter the 4 elements of Y:\n");
        int E, F, G, H;
        scanf("%d %d %d %d", &E, &F, &G, &H);

        // starting the computation
        int M [7];
        M[0] = (A+C)*(E+F);
        M[1] = (B+D)*(G+H);
        M[2] = (A-D)*(E+H);
        M[3] = (A)*(F-H);
        M[4] = (C+D)*(E);
        M[5] = (A+B)*(H);
        M[6] = (D)*(G-E);

        // final matrix
        int I = M[1]+M[2]-M[5]-M[6];
        int J = M[3]+M[5];
        int K = M[4]+M[6];
        int L = M[0]-M[2]-M[3]-M[4];

        // result
        printf("Our matrix Z after computing X*Y=Z is:\n");
        printf("%d %d\n%d %d", I, J, K, L);

        return 0;
}
```

Conclusion - I have hereby successfully executed Strassen's Matrix Multiplication for (2x2) matrices, I've come to understand from this that certain specific algorithms are much better in terms of time and space complexities than other generalized ones.