Strassen's Matrix Algorithm

DAA ASSIGNMENT-01 ,DAA432C_C3 , GROUP 6

Rahul Roy IIT2019194

Chetan Patidar IIT2019193

Deeptarshi Biswas IIT2019195

I.INTRODUCTION

this report, we are going to discuss strassen matrix multiplication,formula of matrix multiplication and algorithms for strassen matrix multiplication.

YStrassen in 1969 which gives an overview that how we can find the multiplication of two 2*2 dimension matrix by the brute-force algorithm. But by using divide and conquer technique the overall complexity for multiplication two matrices is reduced. This happens by decreasing the total number if multiplication performed at the expenses of a slight increase in the number of addition.

For multiplying the two 2*2 dimension matrices Strassen's used some formulas in which there are seven multiplication and eighteen addition, subtraction, and in brute force algorithm, there is eight multiplication and four addition. The utility of Strassen's formula is shown by its asymptotic superiority when order n of matrix reaches infinity. Let us consider two matrices A and B, n*n dimension, where n is a power of two. It can be observed that we can contain four n/2*n/2 submatrices from A, B and their product C. C is the resultant matrix of A and B.

Let us consider two matrices X and Y. We want to calculate the resultant matrix Z by multiplying X and Y. Matrix multiplication is a binary operation thIn at produces a matrix from two matrices. For matrix multiplication, the number of columns in the first matrix must be equal to the number of rows in the second matrix. The resulting matrix, known as the matrix product, has the number of rows of the first and the number of columns of the second matrix. Thus the product XY is defined if and only if the number of columns in X equals the number of rows in.The utility of Strassen's formula is shown by its asymptotic superiority when order n of matrix reaches infinity.

$Z = x_{i1}y_{1j} + x_{i2}y_{2j} + \quad + x_{in}y_{nj}$

$Z = \Sigma x_{ik}y$

II.ALGORITHM CONCEPT

(A) NAIVE METHOD

Simply run three loops. Loop for each row in matrix A with variable i.Inside the above loop, Loop for each column in matrix B with variable j. Inside the above two loops, Loop for each row element in matrix A with variable k and each column element in matrix B with variable k ie, A[i][k] and B[k][j]. we will find the product of each row element in A with each column element in B. ie, A[i][k] * B[k][j] and add all the products and store in the new matrix C[i][j].

(B) Recursive method :

(1) Divide matrices A and B in 4 sub-matrices of size n/2 X n/2.

(2) Calculate following values Recursively. ae+bg , af+bh , ce + dg and cf+dh

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} X \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix}$$

A B C

A,B,C are square matrices of size nxn.
a,b,c,d are submatrices of A of size n/2 x n/2.

e,f,g,h are submatrices of size n/2 x n/2.

In the above method , we do 8 multiplications for matrices of n/2 x n/2 and 4 additions.

(C) Strassen's Algorithm

The Strassen's method of matrix multiplication is a typical divide and conquer algorithm 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 , but in Strassen's method, the four sub-matrices of result are calculated using following formulae.

P1 = a(f-h) P2 = (a+b)h P3 = (c+d)e
P4 = d(g-e) P5 = (a+d)(e+h) P6 = (b-d)(g+h) P7 = (a-c)(e+f)

C11 = p5 + p4 - p2 + p6 C12 = p1 + p2 C21 = p3 + p4
C22 = p1 + p5 - p3 - p7

Here we describe 2 possible approaches :

Approach 1:

This is a brute force approach for multiplication of two matrices

.**Approach 2:**

Divide a matrix of order of n*n recursively till we get the matrix of 2*2.Use the previous set of formulas to carry out 2*2 matrix multiplication.In this eight multiplication and four additions, subtraction is performed.Combine the result of two matrices to find the final product or final matrix.

Algorithm 1:

Algorithm 1: The Naive Matrix Multiplication Algorithm

```
Data: S[A][B], P[G][H]
Result: Q[][]
if B == G then
    for m = 0; m < A; m++ do
        for r = 0; r < H; r++ do
            Q[m][r] = 0;
            for k = 0; k < G; k++ do
                Q[m][r] += S[m][k] * P[k][r];
            end
        end
    end
end
```

Algorithm 2:

begin
If n = threshold then compute
C = a * b is a conventional matrix.
Else
begin
If n = threshold then compute
C = a * b is a conventional matrix.
Else
Partition a into four sub matrices a11, a12, a21, a22.
Partition b into four sub matrices b11, b12, b21, b22.
Strassen (n/2, a11 + a22, b11 + b22, d1)
Strassen (n/2, a21 + a22, b11, d2)
Strassen (n/2, a11, b12 – b22, d3)
Strassen (n/2, a22, b21 – b11, d4)
Strassen (n/2, a11 + a12, b22, d5)
Strassen (n/2, a21 – a11, b11 + b22, d6)
Strassen (n/2, a12 – a22, b21 + b22, d7)
C = d1+d4-d5+d7 d3+d5
d2+d4 d1+d3-d2-d4
end if
return (C)
end.

III. ALGORITHM AND ANALYSIS

Time Complexity

Approach 1;

:Here, we assume that integer operations take O(1) time. Here
we simply run three loops first loop run r1 times, the second
loop runs c2 times, and the final loop runs r2 times.
tavg = O(n ∗ n ∗ n)
tbest = O(0)

Approach 2:

Addition and Subtraction of two matrices takes O(N2) time.
So time complexity can be written as:
T(N) = 7T(N/2) + O(N2)
From Master's Theorem, time complexity of above method is O(NLog7)
tavg = O(NLog7)
tbest = O(0)

. The ratio of this operation count to that required by the standard algorithm alone is seen to be
7n^3 + 11n^2
8n^3 -4n^2 ;
which approaches 7/8 as m gets large, implying that for sufficiently large matrices one level of Strassen's construction produces a 12.5% improvement over regular matrix multiplication.

(B) Space Complexity:
The space complexity of this algorithm for both the approaches are as follows:

Approach 1:

Here we create extra space for storing the result of the multiplication of matrices. Here we also declared r1*c1 size for taking input the first matrix and r2*c2 size for taking input the second Matrix. Space complexity : O(n 2)

Approach 2:

Computing S1, . . . , S7 and the matrix that we return in the last step requires O(n 2) space.We start with 3n 2 space for the two input matrices and the output matrix. We then allocate 3(n/2)2 space for the recursive call to compute S1. Once S1 is returned, we add it to the upper left quadrant of the output matrix. In this same 3(n/2)2 space, we compute S2, and then we add it to the upper left and lower right quadrants of the output matrix. We continue in the same manner for the rest of the Si Space complexity : O(n 2)

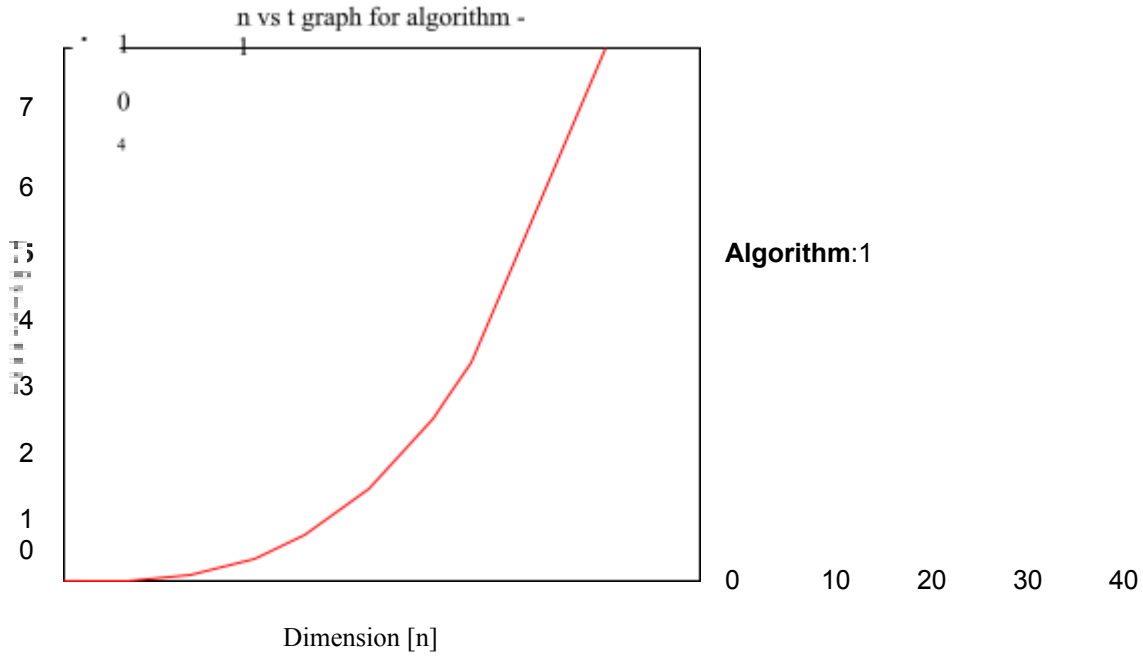
Program for strassen's algorithm

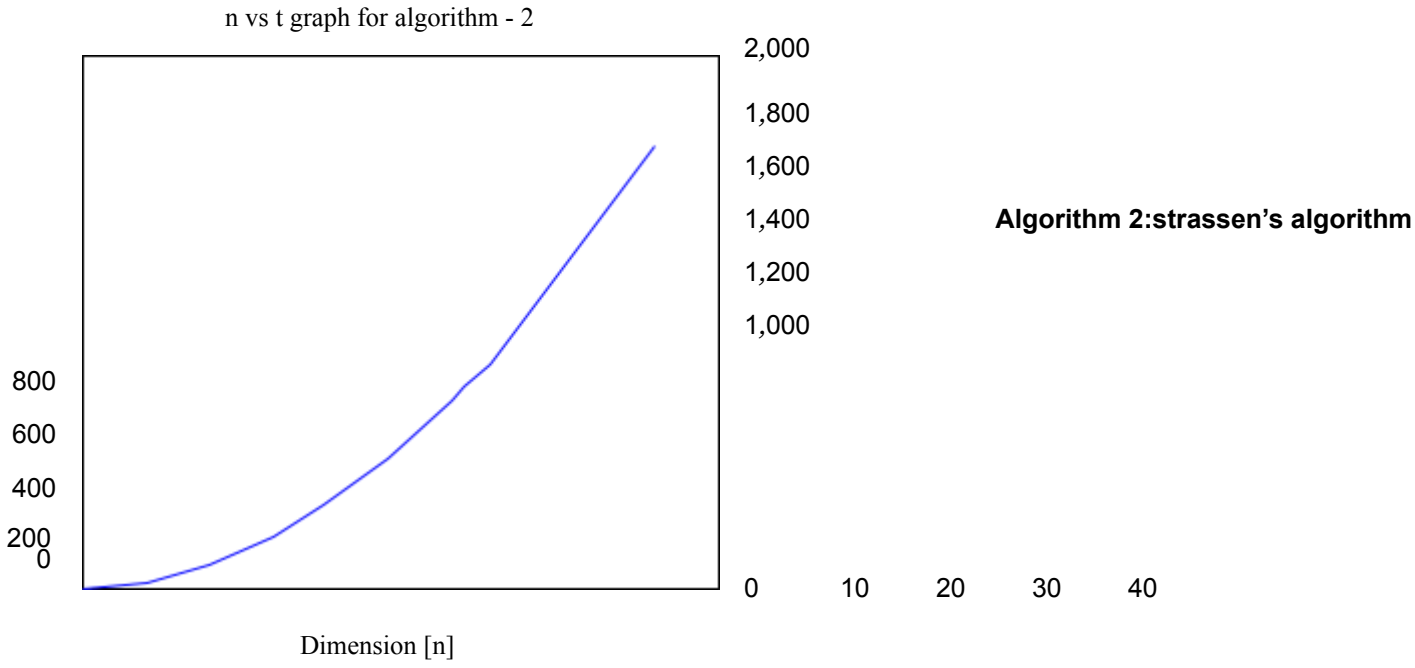
```
def strassen(mA, mB, n):
    n1 = len(mA)
    n2 = len(mB)

    if(n1 and n2 <= n):
        return (mA * mB)

    else:

        print(mA)
        A = partition(mA)
        B = partition(mB)
        c = np.matrix([0 for i in range(len(mA))]for j in
range(len(mB)))
        C = partition(c)

        a11 = np.array(A[0])
        a12 = np.array(A[2])
        a21 = np.array(A[1])
        a22 = np.array(A[3])
b11 = np.array(B[0])
    b12 = np.array(B[2])
    b21 = np.array(B[1])
    b22 = np.array(B[3])

    m1 = np.array(strassen((a11 + a22), (b11 + b22)))
    m2 = np.array(strassen((a21 + a22), b11))
    m3 = np.array(strassen(a11, (b12 - b22)))
    m4 = np.array(strassen(a22, (b21 - b11)))
    m5 = np.array(strassen((a11 + a12), b22))
    m6 = np.array(strassen((a21 - a11), (b11 + b12)))
    m7 = np.array(strassen((a12 - a22), (b21 + b22)))

    C[0] = np.array((m1 + m4 - m5 + m7))
    C[2] = np.array((m3 + m5))
    C[1] = np.array((m2 + m4))
    C[3] = np.array((m1 - m2 + m3 + m6))

    return np.array(C)
```

IV. EXPERIMENTAL ANALYSIS

Graph - 1 : it can be clearly seen that first graph has O(n 3) behaviour.

Graph - 2 : it can be clearly seen that first graph has something like n 2 behaviour.





V. CONCLUSION

In the above divide and conquer method, the main component for high complexity is 8 recursive calls. The idea of **Strassen's method** is to reduce the number of recursive calls to 7.Thus, Implementation of Strassen's algorithm for matrix multiplication is several times faster than the na¨ıve method for matrix multiplication, if applied on large enough matrices.

Parameter	The Naive Algorithm	The Solvay Strassen Algorithm
Time Complexity	$\mathcal{O}(N^3)$	$\mathcal{O}(N^{2.8074})$
Approach	Iterative	Divide-and-conquer
Application	Square as well as non-square matrices	Suitable for square matrices only

VI. REFERENCES

1) https://www.geeksforgeeks.org/ strassens-matrix-multiplication/
2) https://www.cs.cmu.edu/ ~15451-f19/LectureNotes/ lec01-strassen.pdf
3) www.baeldung.com/cs/matrix-multiplication-algorithms
4) cme323_lec3.pdf
5) https://www.includehelp.com/algorithms/strassen-matrix-multiplication.aspx