

<b>Name</b>	<b>Advait Ravi Sapkal</b>
<b>UID No.</b>	<b>2021700055</b>
<b>Experiment No.</b>	<b>3</b>

<b>AIM:</b>	To simulate Strassen's Matrix Multiplication algorithm for square matrices of size $2 \times 2$
<b>Program 1</b>	
<b>PROBLEM STATEMENT:</b>	Given 2 matrices of order $2 \times 2$ each, find their product with an algorithm that has a complexity less than $\Theta(N^2)$ . The algorithm to be used is Strassen's Matrix Multiplication algorithm.
<b>ALGORITHM:</b>	<p>Insertion and selection sort are the 2 most fundamental sorting algorithms. Selection sort is a non-scalable, non-practical algorithm whereas insertion sort is practical for small inputs. Insertion sort is a stable sorting algorithm whereas Selection sort is unstable.</p> <p style="text-align: center;"><b>Strassen's Matrix Multiplication Algorithm</b></p> <p><b>Invariant:</b></p> <ol style="list-style-type: none"> <li>1. Divide the input matrices <math>A</math> and <math>B</math> and output matrix <math>C</math> into <math>n/2 \times n/2</math> submatrices, as in equation (4.9). This step takes <math>\Theta(1)</math> time by index calculation, just as in SQUARE-MATRIX-MULTIPLY-RECURSIVE.</li> <li>2. Create 10 matrices <math>S_1, S_2, \dots, S_{10}</math>, each of which is <math>n/2 \times n/2</math> and is the sum or difference of two matrices created in step 1. We can create all 10 matrices in <math>\Theta(n^2)</math> time.</li> <li>3. Using the submatrices created in step 1 and the 10 matrices created in step 2, recursively compute seven matrix products <math>P_1, P_2, \dots, P_7</math>. Each matrix <math>P_i</math> is <math>n/2 \times n/2</math>.</li> <li>4. Compute the desired submatrices <math>C_{11}, C_{12}, C_{21}, C_{22}</math> of the result matrix <math>C</math> by adding and subtracting various combinations of the <math>P_i</math> matrices. We can compute all four submatrices in <math>\Theta(n^2)</math> time.</li> </ol>

**Algorithm:**

```
SQUARE-MATRIX-MULTIPLY-STRASSEN ( $A, B$ )
1   $n = A.rows$ 
2  let  $C$  be a new  $n \times n$  matrix
3  if  $n == 1$ 
4       $c_{11} = a_{11} \cdot b_{11}$ 
5  else partition  $A$ ,  $B$ , and  $C$ 
6      let  $S_1, S_2, \dots$ , and  $S_{10}$  be 10 new  $n/2 \times n/2$  matrices
7      let  $P_1, P_2, \dots$ , and  $P_7$  be 7 new  $n/2 \times n/2$  matrices
8
9      // calculate the sum matrices
10      $S_1 = B_{12} - B_{22}$ 
11      $S_2 = A_{11} + A_{12}$ 
12      $S_3 = A_{21} + A_{22}$ 
13      $S_4 = B_{21} - B_{11}$ 
14      $S_5 = A_{11} + A_{22}$ 
15      $S_6 = B_{11} + B_{22}$ 
16      $S_7 = A_{12} - A_{22}$ 
17      $S_8 = B_{21} + B_{22}$ 
18      $S_9 = A_{11} - A_{21}$ 
19      $S_{10} = B_{11} + B_{12}$ 
20
21     // calculate the product matrices
22      $P_1 = \text{SQUARE-MATRIX-MULTIPLY-STRASSEN}(A_{11}, S_1)$ 
23      $P_2 = \text{SQUARE-MATRIX-MULTIPLY-STRASSEN}(S_2, B_{22})$ 
24      $P_3 = \text{SQUARE-MATRIX-MULTIPLY-STRASSEN}(S_3, B_{11})$ 
25      $P_4 = \text{SQUARE-MATRIX-MULTIPLY-STRASSEN}(A_{22}, S_4)$ 
26      $P_5 = \text{SQUARE-MATRIX-MULTIPLY-STRASSEN}(S_5, S_6)$ 
27      $P_6 = \text{SQUARE-MATRIX-MULTIPLY-STRASSEN}(S_7, S_8)$ 
28      $P_7 = \text{SQUARE-MATRIX-MULTIPLY-STRASSEN}(S_9, S_{10})$ 
29
30     // calculate the final product sub matrices
31      $C_{11} = P_4 + P_5 + P_6 - P_2$ 
32      $C_{12} = P_1 + P_2$ 
33      $C_{21} = P_3 + P_4$ 
34      $C_{22} = P_1 + P_5 - P_3 - P_7$ 
35  return  $C$ 
```

### Trace:

Trace:

$$\begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix} \begin{pmatrix} 6 & 8 \\ 4 & 2 \end{pmatrix} = \begin{pmatrix} 18 & 14 \\ 62 & 66 \end{pmatrix} \leftarrow \text{(mathematically)}$$

A B

sub

$$A_{11}=1 \quad A_{21}=7 \quad B_{11}=6 \quad B_{21}=4$$
$$A_{12}=3 \quad A_{22}=5 \quad B_{12}=8 \quad B_{22}=2$$

sum

$$S_1 = 8 - 2 = 6 \quad S_6 = 6 + 2 = 8$$
$$S_2 = 1 + 3 = 4 \quad S_7 = 3 - 5 = -2$$
$$S_3 = 7 + 5 = 12 \quad S_8 = 4 + 2 = 6$$
$$S_4 = 4 - 6 = -2 \quad S_9 = 1 - 7 = -6$$
$$S_5 = 1 + 5 = 6 \quad S_{10} = 8 + 6 = 14$$

pro

$$P_1 = 1 \times 6 = 6 \quad P_5 = 6 \times 8 = 48$$
$$P_2 = 2 \times 4 = 8 \quad P_6 = -2 \times 6 = -12$$
$$P_3 = 12 \times 6 = 72 \quad P_7 = -6 \times 14 = -84$$
$$P_4 = 5 \times 2 = 10$$

final

$$C_{11} = -10 + 48 - 8 + (-12) = 18$$
$$C_{12} = 6 + 8 = 14$$
$$C_{21} = 72 - 10 = 62$$
$$C_{22} = 54 - 72 + 84 = 66$$

$\therefore \begin{bmatrix} 18 & 14 \\ 62 & 66 \end{bmatrix}$  which is correct

### Code:

```
//strassens matrix multiplication algorithm demo
#include<stdio.h>

//for printing a 2 x 2 matrix
void print_matrix(int mat1[2][2]){
    for(int i=0;i<2;i++){
        for(int j=0;j<2;j++){
            printf(" %d",mat1[i][j]);
        }
        printf(" \n");
    }
}

int main(){

    //initializing
    int A[2][2]={    {3,4},
                     {1,2}    };
}
```

```

int B[2][2]={    {2,0},
                  {0,2}   };

//display
print_matrix(A);   printf("  X\n");   print_matrix(B);

//sum matrices from sub matrices
int S[11];
S[ 1]=B[0][1]-B[1][1];  S[ 6]=B[0][0]+B[1][1];
S[ 2]=A[0][0]-A[0][1];  S[ 7]=A[0][1]-A[1][1];
S[ 3]=A[1][0]-A[1][1];  S[ 8]=B[1][0]+B[1][1];
S[ 4]=B[1][0]-B[0][0];  S[ 9]=A[0][0]-A[1][0];
S[ 5]=A[0][0]-A[1][1];  S[10]=B[0][0]+B[1][1];

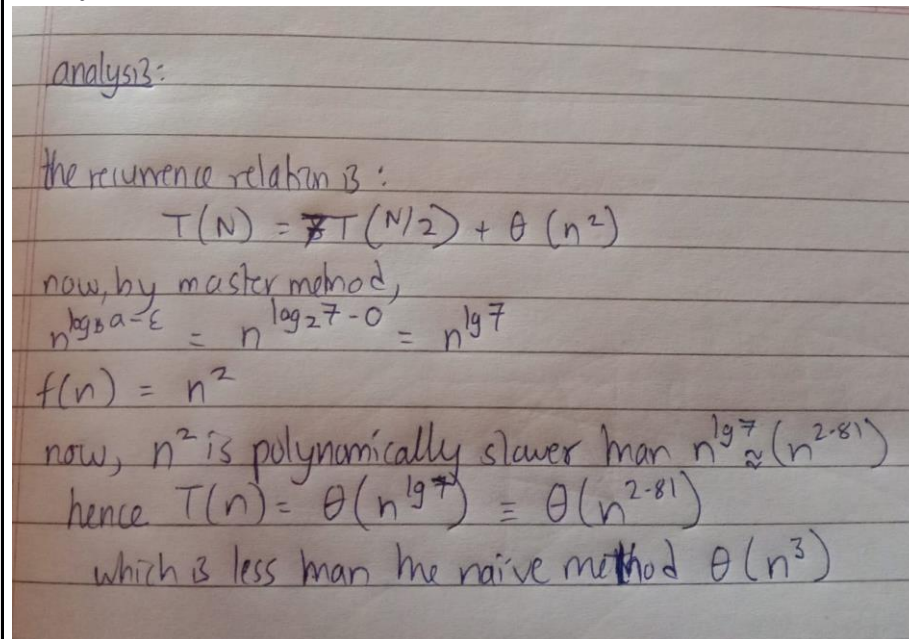
//product matrices from sum matrices
int P[8];
P[1]=A[0][0]*S[1];      P[5]=S[5]*S[6];
P[2]=B[1][1]*S[2];      P[6]=S[7]*S[8];
P[3]=B[0][0]*S[3];      P[7]=S[9]*S[10];
P[4]=A[1][1]*S[4];

//final matrices from product matrices
int ret[2][2];
ret[0][0]=P[5]+P[4]-P[2]+P[6];
ret[0][1]=P[1]+P[2];
ret[1][0]=P[3]+P[4];
ret[1][1]=P[1]+P[5]-P[3]-P[7];

//display
printf("  =\n");
print_matrix(ret);
}

```

### Analysis:



### RESULT:

### Outputs

d:\DAA\expt3>	d:\DAA\expt3>	d:\DAA\expt3>	d:\DAA\expt3>
3 4	1 3	3 4	3 4
1 2	7 5	1 2	1 2
X	X	X	X
2 0	6 8	0 0	1 0
0 2	4 2	0 0	0 1
=	=	=	=
6 8	18 14	0 0	3 4
2 4	62 66	0 0	1 2

### Conclusions:

1. There are 7 effective matrix multiplications compared to the 8 in naïve multiplication. This is because we converted the cost of the 8<sup>th</sup> multiplication from a recurrence cost to an asymptotic cost. Due to this, the coefficient on the recurrence relation was reduced, and the one in the asymptotic cost increased, but it will anyways get subsumed.
2. The algorithm is only slightly better than the naïve one, if we compare  $\lg 7$  with 3, the difference is only 0.20.
3. The algorithm is not much effective for practical use due to various reasons like accumulating memory of the submatrices, higher constant factor, ineffectiveness on sparse matrices, etc.

- |  |  |
|--|--|
|  | <ol style="list-style-type: none"><li>4. The rounding errors in SMM get accumulated and result in not a very precise answer.</li><li>5. Although useful for larger matrices, for sparse or small matrices, the naïve method performs better.</li><li>6. The code is too complex, requiring &gt;50 lines even in languages like Python.</li></ol> <hr/> |
|--|--|