Name	Rishi Tiku
UID no.	2021700067
Experiment No.	3

AIM:	Implement Divide and Conquer technique.
Program	
PROBLEM STATEMENT :	Implement Strassen's Matrix Multiplication algorithm to multiply two 2x2 matrices.

ALGORITHM/ THEORY:	Let us assume two matrices X and Y. We want to calculate the resultant matrix Z by multiplying X and Y.
	
	Naïve Method:
	
	First, we will discuss naïve method and its complexity. Here, we are calculating Z = X × Y. Using Naïve method, two matrices (X and Y) can be multiplied if the order of these matrices are p × q and q × r. Following is the algorithm.
	
	`void multiply(int A[][N], int B[][N], int C[][N])`
	`{`
	`  for (int i = 0; i < N; i++)`
	`  {`
	`  for (int j = 0; j < N; j++)`
	`  {`
	`  C[i][j] = 0;`
	`  for (int k = 0; k < N; k++)`
	`  {`
	`  C[i][j] += A[i][k]*B[k][j];`
	`  }`
	`  }`
	`  }`
	`}`
	
	There are three for loops in this algorithm and one is nested in other. Hence, the algorithm takes $O(n^3)$ time to execute.
	
	Strassen's Matrix Multiplication Algorithm:
	
	Strassen's Matrix multiplication can be performed only on square matrices where n is a power of 2. Order of both of the matrices are n × n.

Divide X, Y into four (n/2)×(n/2) matrices and Using Strassen's Algorithm compute the following –

```
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]);
```

Then,

```
c[0][0] = m1 + m4- m5 + m7;
c[0][1] = m3 + m5;
c[1][0] = m2 + m4;
c[1][1] = m1 - m2 + m3 + m6;
```

Time Complexity of Strassen's Method

Addition and Subtraction of two matrices takes $O(N^2)$ time. So time complexity can be written as

$$T(N) = 7T(N/2) + O(N^2)$$

From Master's Theorem, time complexity of above method is $O(N^{Log7})$ which is approximately $O(N^{2.8074})$

Generally Strassen's Method is not preferred for practical applications for following reasons.

1. The constants used in Strassen's method are high and for a typical application Naive method works better.
2. For Sparse matrices, there are better methods especially designed for them.
3. The submatrices in recursion take extra space.
4. Because of the limited precision of computer arithmetic on non-integer values, larger errors accumulate in Strassen's algorithm than in Naive Method.

In the naïve method we use to do 8 multiplications using Strassen's we have reduced that to 7 multiplications.

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("Enter the 4 elements of first matrix: ");
  for(i = 0;i < 2; i++)
      for(j = 0;j < 2; j++)
           scanf("%d", &a[i][j]);

  printf("Enter the 4 elements of second matrix: ");
  for(i = 0; i < 2; i++)
      for(j = 0;j < 2; j++)
           scanf("%d", &b[i][j]);

  printf("\nThe first matrix is\n");
  for(i = 0; i < 2; i++){
      printf("\n");
      for(j = 0; j < 2; j++)
           printf("%d\t", a[i][j]);
  }

  printf("\nThe second matrix is\n");
  for(i = 0;i < 2; i++){
      printf("\n");
      for(j = 0;j < 2; j++)
           printf("%d\t", b[i][j]);
  }

  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");

    printf("\nAfter multiplication using Naive algorithm \n");
    printf("%dx%d + %dx%d\t", a[0][0], b[0][0], a[0][1],
b[1][0]);
    printf("%dx%d + %dx%d\n", a[0][0], b[0][1], a[0][1],
b[1][1]);
    printf("%dx%d + %dx%d\t", a[1][0], b[0][0], a[1][1],
b[1][0]);
    printf("%dx%d + %dx%d\n", a[1][0], b[0][1], a[1][1],
b[1][1]);

    printf("\n\n");

    printf("%d\t", a[0][0] * b[0][0] + a[0][1] * b[1][0]);
    printf("%d\n", a[0][0] * b[0][1] + a[0][1] * b[1][1]);
    printf("%d\t", a[1][0] * b[0][0] + a[1][1] * b[1][0]);
    printf("%d\n", a[1][0] * b[0][1] + a[1][1] * b[1][1]);

    printf("\n\nThis proves strassens' method provides the same
result as naive method.\n");

   return 0;
}
```

Output:

```
Enter the 4 elements of first matrix: 1 2 3 4 5 6 7 8
Enter the 4 elements of second matrix:
The first matrix is

1       2
3       4
The second matrix is

5       6
7       8
After multiplication using Strassen's algorithm

19      22
43      50

After multiplication using Naive algorithm
1x5 + 2x7       1x6 + 2x8
3x5 + 4x7       3x6 + 4x8


19      22
43      50


This proves strassens' method provides the same result as naive method.
```

CONCLUSION:	Strassens' method of Matrix multiplication is implemented for a 2x2 matrix.