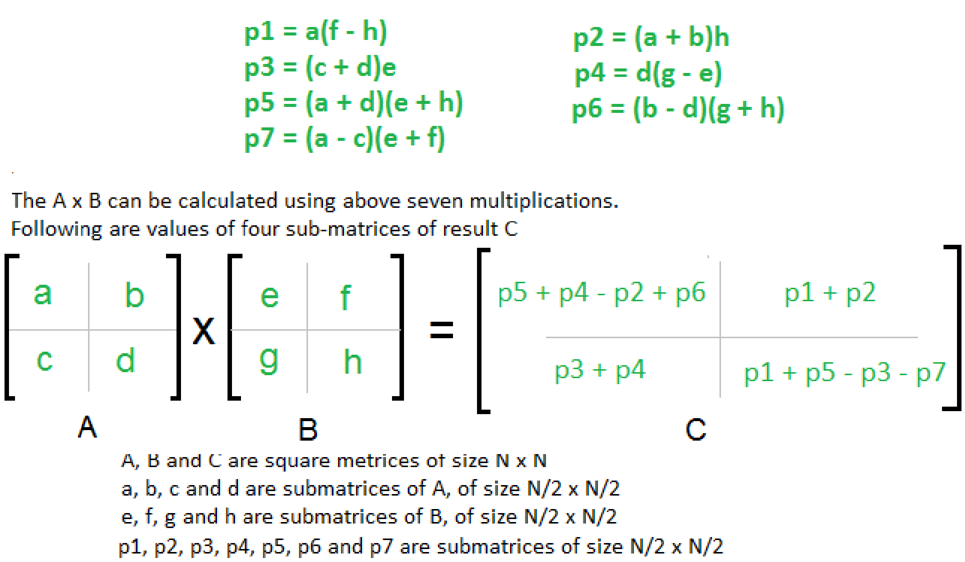
NAME:	Krish Jain
UID:	2021300049
SUBJECT	DAA
EXPERIMENT NO :	03
AIM:	To perform matrix multiplication using strassen’s algorithm.
PROBLEM STATEMENT 1:	
THEORY	The main reason for high time complexity in naive divide and conquer is because of 8 recursive calls.The strassen matrix reduces these recursive calls to 7.The idea is similar to naive divide and conquer i.e divide the matrix in sub-matrices of N/2xN/2 dimension until we get 2x2 matrix.We compute this matrix using formulae provided by strassens algorithm.



	As there are 7 recursive calls we get following recurrence- T(N) = 7T(N/2) + O(N^2) Solving above recurrence we get time complexity of O(N^2.81) which is less than naive approach. Disadvantages of strassens algorithm- 1.)The constants used in strassens multiplication are high and naive approach is used for typical application. 2.)For sparse matrix there are better algorithms. 3.)The sub-matrices in recursion take extra space. 4.)Because of limited percision of computer arithemetic on non integer values,large errors accumulate in strassens algorithm.
ALGORITHM	1.)Divide the input matrices A and B into N/2xN/2 dimension. After dividing we get sub-matrices A11,A12,A21,A22 and B11,B12,B21,B22. 2.)Computing sub-matrices using following formulae- M1=(A11+A22)*(B11+B22) M2=(A21+A22)*B11 M3=A11*(B12-B22) M4=A22*(B21-B11) M5=(A11+A12)*B22 M6=(A21-A11)*(B11+B12) M7=(A12-A22)*(B21+B22) 3.)Compute the output using intermediate matrices C11=M1+M4-M5+M7 C12=M3+M5 C21=M2+M4 C22=M1-M2+M3+M6

4.)Combine the output matrices for final matrix.

PROGRAM:

strassen.cpp > main()

```cpp
#include<iostream>

using namespace std;

int main(){
    int A[2][2],B[2][2];
    cout << "Enter 4 elements for matrix A: ";
    for(int i=0;i<2;i++)
        for(int j=0;j<2;j++)
            cin >> A[i][j];
    cout << "Enter 4 elements for matrix B: ";
    for(int i=0;i<2;i++)
        for(int j=0;j<2;j++)
            cin >> B[i][j];
    int p[7];
    p[0] = A[0][0]*(B[0][1] - B[1][1]);
    p[1] = (A[0][0] + A[0][1])*B[1][1];
    p[2] = (A[1][0] + A[1][1])*B[0][0];
    p[3] = A[1][1]*(B[1][0] - B[0][0]);
    p[4] = (A[0][0] + A[1][1])*(B[0][0] + B[1][1]);
    p[5] = (A[0][1] - A[1][1])*(B[1][0] + B[1][1]);
    p[6] = (A[0][0] - A[1][0])*(B[0][0] + B[0][1]);

    int C[2][2];
    C[0][0] = p[3]+p[4]-p[1]+p[5];
    C[0][1] = p[0] + p[1];
    C[1][0] = p[2] + p[3];
    C[1][1] = p[0] + p[4] - p[2] -p[6];

    cout << "Multiplication of A and B using strassen's algorithm: " << endl;
    for(int i=0;i<2;i++){
        for(int j=0;j<2;j++){
            cout << C[i][j] << "\t";
        }
        cout << endl;
    }
}
```

RESULT (SNAPSHOT)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS E:\Sem4\DAA\Exp3> cd "e:\Sem4\DAA\Exp3\" ; if ($?) { g++ strassen.cpp
Enter 4 elements for matrix A: 1 2 3 4
Enter 4 elements for matrix B: 5 6 7 8
Multiplication of A and B using strassen's algorithm:
19      22
43      50
PS E:\Sem4\DAA\Exp3>
```

CONCLUSION:	Through this experiment I understood how to compute matrices product using strassens algorithm.The complexity of this alogrithm was found out to be $O(N^{2.81})$ which is slightly better than naive approach whose time complexity is $O(N^3)$.This can have huge impact on matrices with high dimensions.