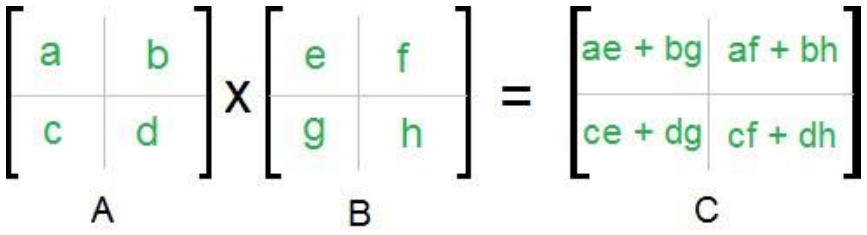
Name	**Gitanjali Gangurde**
UID	**2021300034**
Class	**SE-Comps A Batch-B**
Experiment No.	**3**
AIM:	Use divide and conquer strategy: Strassen's matrix multiplication.
THEORY:	Divide and Conquer method to multiply two square matrices: 1. Divide matrices A and B in 4 sub-matrices of size N/2 x N/2 as shown in the below diagram. 2. Calculate following values recursively. ae + bg, af + bh, ce + dg and cf + dh.  A, B and C are square metrices of size N x N a, b, c and d are submatrices of A, of size N/2 x N/2 e, f, g and h are submatrices of B, of size N/2 x N/2

CODE:	

```c
#include <stdio.h>
#include <time.h>
int main()
{ int i,j; int a[2][2], b[2][2], c[2][2]; int
    s[10], p[8]; clock_t start, end; printf("Enter
    the elements of first matrix:"); for(int
    i=0;i<2;i++)
    { for (int j = 0; j <2; j++)
        { scanf("%d",&a[i][j]);
        }
```

```
    }

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

s[0]=b[0][1]-b[1][1]; s[1]=a[0][0]+a[0][1];
s[2]=a[1][0]+a[1][1]; s[3]=b[1][0]-b[0][0];
s[4]=a[0][0]+a[1][1]; s[5]=b[0][0]+b[1][1];
s[6]=a[0][1]-a[1][1]; s[7]=b[1][0]+b[1][1];
s[8]=a[0][0]-a[1][0]; s[9]=b[0][0]+b[0][1];

for(int i=0;i<10;i++)
{ printf("S%d: %d",(i+1),s[i]);
    printf("\n");
}

p[0]=s[0]*a[0][0];
p[1]=s[1]*b[1][1];
p[2]=s[2]*b[0][0];
p[3]=s[3]*a[1][1];
p[4]=s[4]*s[5]; p[5]=s[6]*s[7];
p[6]=s[8]*s[9];

printf("\n"); for(i=0;i<7;i++)
{ printf("P%d:
%d",(i+1),p[i]);
printf("\n");
}
```

```
c[0][0]=p[4]+p[3]-p[1]+p[5];
c[0][1]=p[0]+p[1];
c[1][0]=p[2]+p[3];
c[1][1]=p[4]+p[0]-p[2]-p[6];

printf("\nMatrix A =");
for(i=0;i<2;i++)
{
printf("\n");
for(j=0;j<2;j++)
{
printf("%d\t",a[i][j]);
}
}
printf("\n\nMatrix B =");
for(i=0;i<2;i++)
{
printf("\n");

for(j=0;j<2;j++)
{
printf("%d\t",b[i][j]);
}
}
printf("\n\nMatrix C =");
for(i=0;i<2;i++)
{
printf("\n");
for(j=0;j<2;j++)
{
printf("%d\t",c[i][j]);
}
}
printf("\n");
end=clock();
double diff=(double)((end-start)/CLOCKS_PER_SEC);
return 0;
}
```

OUTPUT:

```
Enter the elements of first matrix:1
2
3
4
Enter the elements of second matrix:5
6
7
8
S1: -2
S2: 3
S3: 7
S4: 2
S5: 5
S6: 13
S7: -2
S8: 15
S9: -2
S10: 11

P1: -2
P2: 24
P3: 35
P4: 8
P5: 65
P6: -30
P7: -22

Matrix A =
1       2
3       4

Matrix B =
5       6
7       8
```

```
Matrix C =
19      22
43      50
```

CONCLUSION:	1. Time complexity of normal matrix multiplication is given as : $T(N) = 8T(N/2) + O(N2)$ 2. From Master's Theorem, time complexity of above method is $O(N^3)$ 3. Normal matrix multiplication method has 8 recursive calls, the idea of Strassen's method is to reduce the number of recursive calls to 7. 4. Time Complexity of Strassen's Method : $T(N) = 7T(N/2) + O(N2)$ 5. From Master's Theorem, time complexity of above method is $O(NLog7)$ which is approximately $O(N^{2.8074})$.
