Name:Kunal Virendra Bhatia

UID:2021300010

Experiment No:3

Aim – Experiment to implement strassens matrix multiplication on 2x2 matrix.

Objective: To understand the running time of Strassens matrix multiplication

Theory:

In linear algebra, the **Strassen algorithm**, named after Volkar Strassen, is an algorithm for matrix multiplication. It is faster than the standard matrix multiplication algorithm for large matrices, with a better asymptotic complexity, although the naive algorithm is often better for smaller matrices. The Strassen algorithm is slower than the fastest known alogrithm for extremely large matrices, but such galactic matrices are not useful in practice, as they are much slower for matrices of practical size. For small matrices even faster algorithms exist.

Strassen's algorithm works for any ring, such as plus/multiply, but not all semirings, such as min-plus or boolean algebra, where the naive algorithm still works.

Algorithm:

step 1: Start

step 2: Take 2 matrics as input from user say A and B Step2:

Divide A and B into 10 matrices of n/2 size

```
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];
```

Step 3: Compute p1 to p7

```
P[0] = A[0][0] * S[0];

P[1] = B[1][1] * S[1];

P[2] = B[0][0] * S[2];
```

P[3] = A[1][1] * S[3];

P[4] = S[5] * S[4];

P[5] = S[6] * S[7];

P[6] = S[8] * S[9];

Step 4: computer the resultant matrix c: 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];

Step5: display c

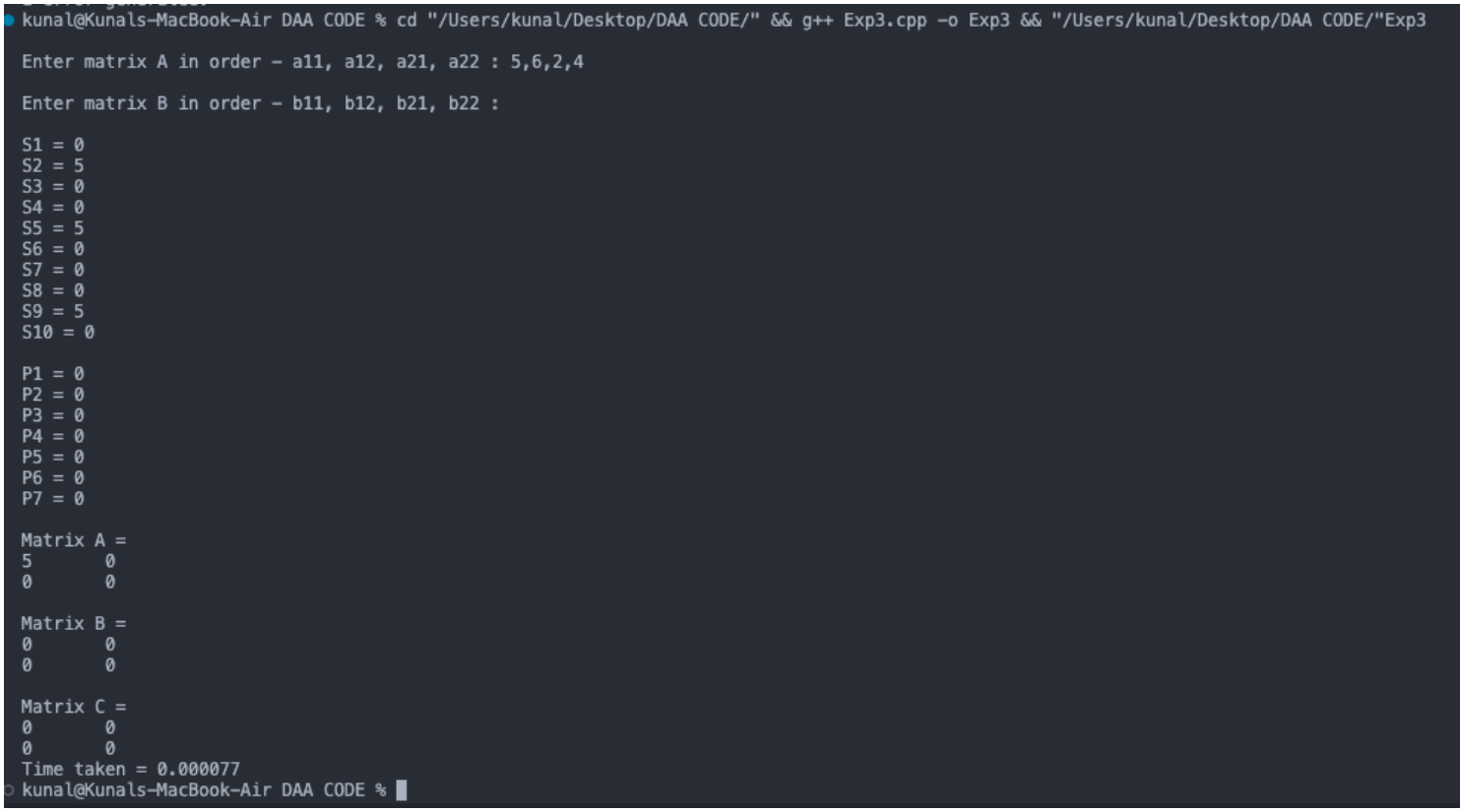
Step6: End

Program:

```
//Strassen multiplication
#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[7];
 clock_t start,end;
 printf("\nEnter matrix A in order - a11, a12, a21, a22 : ");
 for(i=0;i<2;i++)
 {
   for(j=0;j<2;j++)
   {
     scanf("%d",&a[i][j]);
   }
 }
 printf("\nEnter matrix B in order - b11, b12, b21, b22 : ");
 for(i=0;i<2;i++)
 {
   for(j=0;j<2;j++)
   {
     scanf("%d",&b[i][j]);
   }
 }
 start=clock();
 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];
 printf("\n");
 for(i=0;i<10;i++)
 {
   printf("\nS%d = %d",(i+1),s[i]);
 }
 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("\nP%d = %d",(i+1),p[i]);
 }
 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("\n\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();
 printf("Time taken = %lf\n",(double)(end-start)/CLOCKS_PER_SEC);
}
```

Result:



I

Conclusion:

In this experiment I learnt that Stassens matrix multiplication improves the run time a lot when multiplying matrices than traditional matrix multiplicationAlso it is very simple to implement but storage complexity is larger.