Experiment No. 3

Name: Om Bhutki

UID: 2022301003

Subject: DAA

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

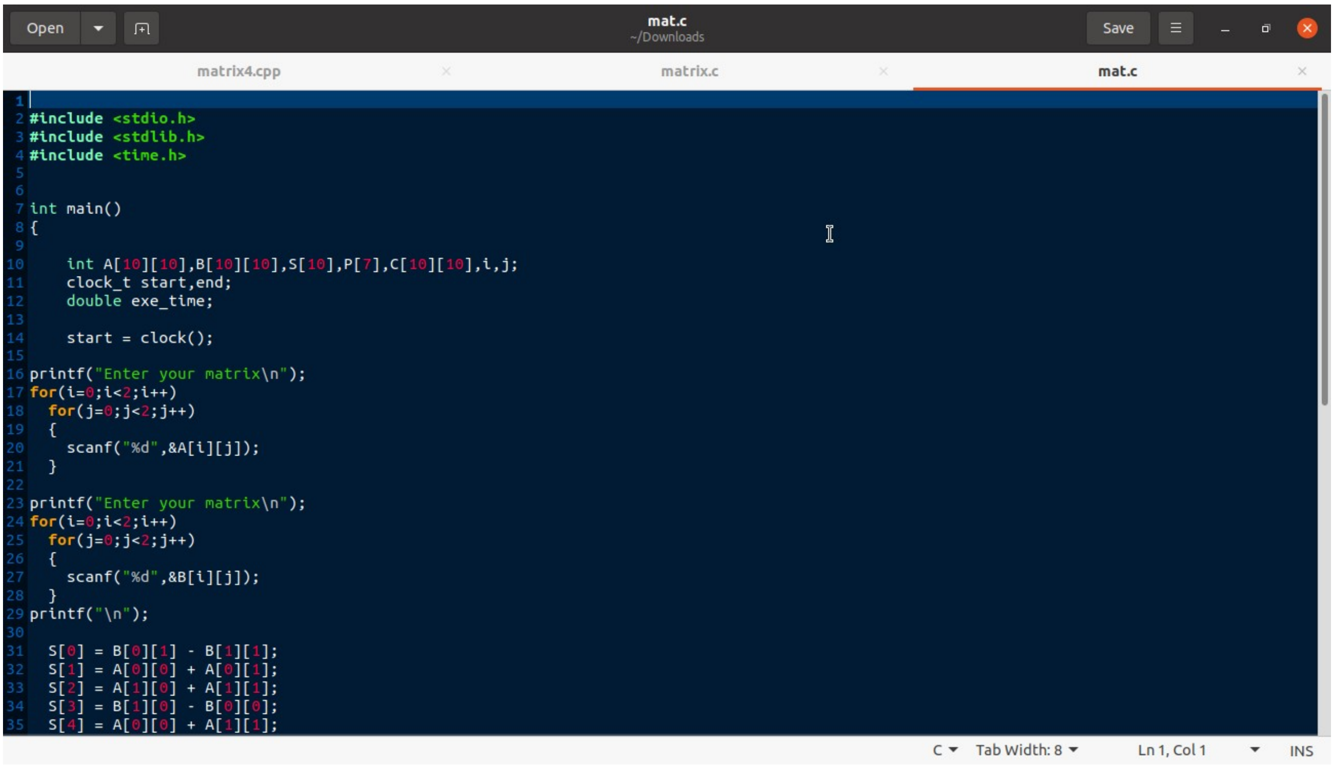
Step2: Divide A and B into 10 matrices of n/2 size

```
  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];
```

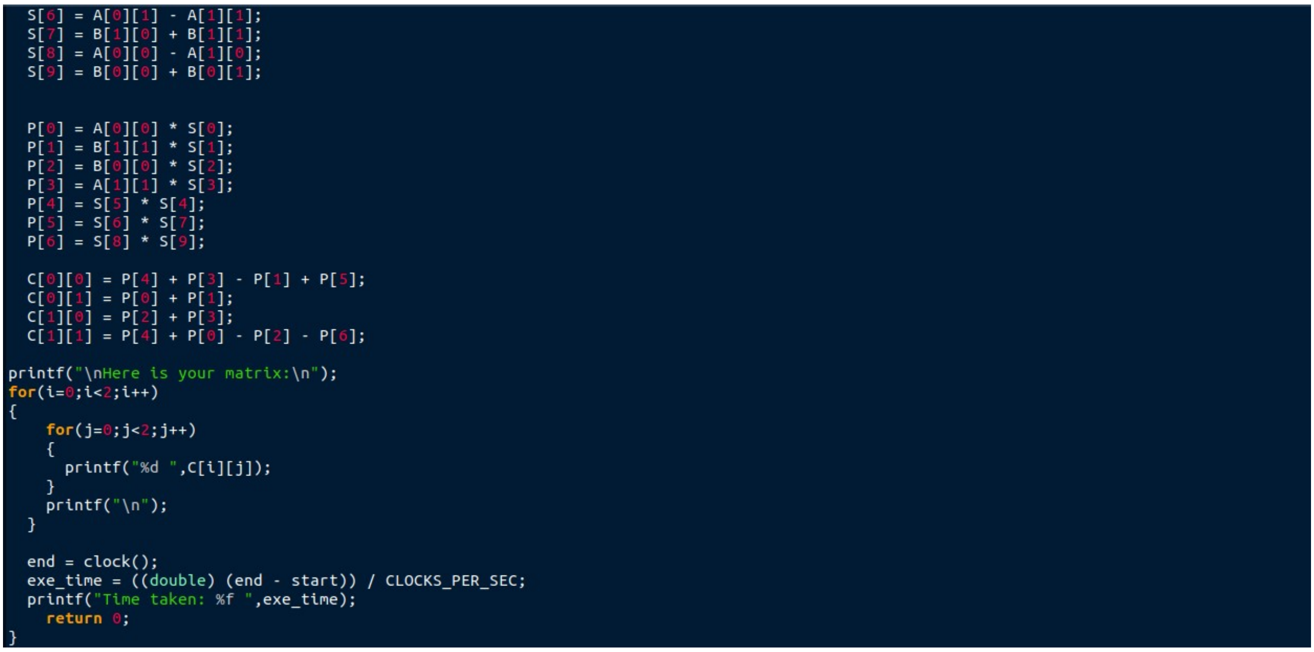
Step 3: Compute p1 to p7

```
  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:

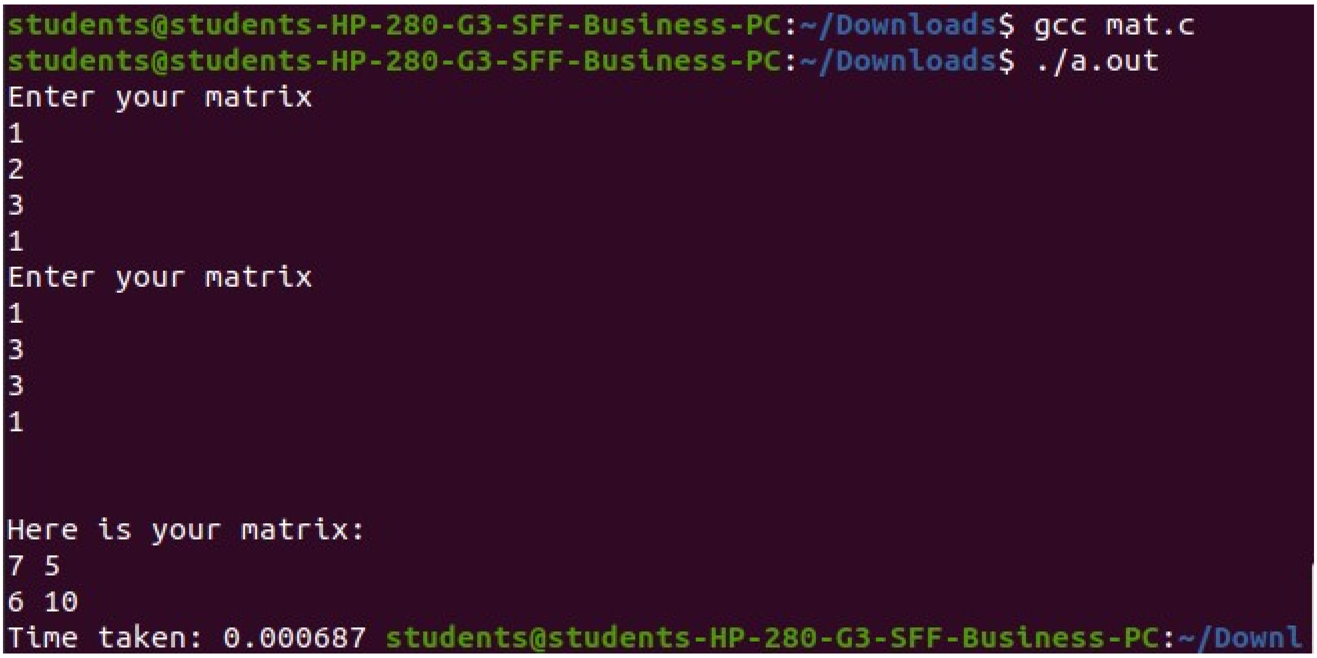


```c

#include <stdio.h>
#include <stdlib.h>
#include <time.h>


int main()
{

    int A[10][10],B[10][10],S[10],P[7],C[10][10],i,j;
    clock_t start,end;
    double exe_time;

    start = clock();

printf("Enter your matrix\n");
for(i=0;i<2;i++)
  for(j=0;j<2;j++)
  {
    scanf("%d",&A[i][j]);
  }

printf("Enter your matrix\n");
for(i=0;i<2;i++)
  for(j=0;j<2;j++)
  {
    scanf("%d",&B[i][j]);
  }
printf("\n");

  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];
```



```c
  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];


  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];

  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("\nHere is your matrix:\n");
for(i=0;i<2;i++)
{
    for(j=0;j<2;j++)
    {
      printf("%d ",C[i][j]);
    }
    printf("\n");
  }

  end = clock();
  exe_time = ((double) (end - start)) / CLOCKS_PER_SEC;
  printf("Time taken: %f ",exe_time);
    return 0;
}
```

Result:



```
students@students-HP-280-G3-SFF-Business-PC:~/Downloads$ gcc mat.c
students@students-HP-280-G3-SFF-Business-PC:~/Downloads$ ./a.out
Enter your matrix
1
2
3
1
Enter your matrix
1
3
3
1


Here is your matrix:
7 5
6 10
Time taken: 0.000687 students@students-HP-280-G3-SFF-Business-PC:~/Downl
```

I

Conclusion:

Thus, after performing this experiment I understood that Stassens matrix multiplication improves the run time a lot when multiplying matrices than traditional matrix multiplication

It is also very easy to implement but takes a lot of space as we need to store multiple arrays.