LAB 5

Name : Rahul Thapar
ID : 1410110321

RESULT
TIME FOR MATRIX MULTIPLICATION (128 X 128)

RECURSION : 0.015271 sec
STRASSENS : 0.004197 sec

CONCLUSION :
Matrix Multiplication using Strassens Algorithm takes less time than using Recursion.

CODE :

Matrix Multiplication using Recursion :

```
/*
  @author : Rahul Thapar
  ID : 1410110321
  Date : 7th Feb, 2017

  Matrix Multiplication using Recursion [128 X 128]

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

#define MAX 128

void matrix_multiply(int[MAX][MAX],int[MAX][MAX],int[MAX][MAX]);
void display(int[MAX][MAX]);

void matrix_multiply(int A[MAX][MAX], int B[MAX][MAX], int C[MAX][MAX]){

  static int i = 0, j = 0, k = 0;

  if (i >= MAX)
    return;
  e
    if (j < MAX){
      if (k < MAX){
```

```
        C[i][j] += A[i][k] * B[k][j];
        k++;
        matrix_multiply(A, B, C);
      }
      k = 0;
      j++;
      matrix_multiply(A, B, C);
    }
    j = 0;
    i++;
    matrix_multiply(A, B, C);
  }
}

void display(int C[MAX][MAX]){

  int i, j;

  f
    for (j = 0 ; j <= MAX ; j++)
      printf("%d ", C[i][j]);
    printf("\n");
  }
  printf("\n");
}

int main(){

  int A[MAX][MAX], B[MAX][MAX], C[MAX][MAX] = {0};

  clock_t end_t, start_t;
  double total_t;
  srand(time(NULL));

  int i, j;
  f
    for(j = 0; j < MAX; j++)
      A[i][j] = rand() % 10;
  }

  for(i = 0; i < MAX; i++){
    for(j = 0; j < MAX; j++)
      B[i][j] = rand() % 10;
```

```
  }

  printf("\t MATRIX A\n ");
  display(A);

  printf("\t MATRIX B\n ");
  display(B);

  start_t = clock();

  printf("\t MATRIX C\n");
  matrix_multiply(A, B, C);
  display(C);

  end_t = clock();
                                      start_t) / CLOCKS_PER_SEC;
  printf("TOTAL TIME TAKEN  : %lf\n", total_t);

  return 0;


}
```

Matrix Multiplication using Strassens Algorithm :

```
/*
  @author : Rahul Thapar
  ID : 1410110321
  Date : 7th Feb, 2017

  Matrix Multiplication using Strassen Algorithm [128 X 128]

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

const int N = 128;

void display(m matrix){

      int i, j;
      for (i = 0 ; i <= 128 ; i++){
```

```
            for (j = 0 ; j <= 128 ; j++)
                  printf("%d ", matrix.a[i][j]);
            printf("\n");
      }
       printf("\n");
}

m plus(m m1, m m2){

       m result;
       int m1_i, m1_j;
       int m2_i, m2_j;
       int i, j;
       int n = m1.re    m1.rs;

       result.rs = result.cs = 0;
       result.re = result.ce = n;

       for (m1_i=m1.rs, m2_i=m2.rs, i=0 ; m1_i<=m1.re ; m1_i++, m2_i++, i+
+)

       for (m1_j=m1.cs, m2_j=m2.cs, j=0 ; m1_j<=m1.ce ; m1_j++, m2_j++, j+
+)
                  result.a[i][j] = m1.a[m1_i][m1_j] + m2.a[m2_i][m2_j];

       return result;
}
m minus(m m1, m m2){

       m result;
       int m1_i, m1_j;
       int m2_i, m2_j;
       int i, j;
       int n = m1.re    m1.rs;

       result.rs = result.cs = 0;
       result.re = result.ce = n;

       for (m1_i=m1.rs, m2_i=m2.rs, i=0 ; m1_i<=m1.re ; m1_i++, m2_i++, i+
+)

       for (m1_j=m1.cs, m2_j=m2.cs, j=0 ; m1_j<=m1.ce ; m1_j++, m2_j++, j+
+)
                  result.a[i][j] = m1.a[m1_i][m1_j]    m2.a[m2_i][m2_j];
```

```
      return result;

}

m multiply(m m1, m m2){

    m A, B, C, D, E, F, G, H;
    m P1, P2, P3, P4, P5, P6, P7;
    m Q1, Q2, Q3, Q4;
    m result;
    int m1_i, m1_j;
    int i, j;
    int n = m1.re   m1.rs + 1;

    /*If N == 2*/
    if (n <= 2) {

      int a, b, c, d, e, f, g, h;

      /*Applying Strassen's Formulae*/
      m m3 = m1;
      a = m1.a[m1.rs][m1.cs];
      b = m1.a[m1.rs][m1.cs+1];
      c = m1.a[m1.rs+1][m1.cs];
      d = m1.a[m1.rs+1][m1.cs+1];
      e = m2.a[m2.rs][m2.cs];
      f = m2.a[m2.rs][m2.cs+1];
      g = m2.a[m2.rs+1][m2.cs];
      h = m2.a[m2.rs+1][m2.cs+1];
      m3.a[m3.rs][m3.cs] = a*e + b*g;
        m3.a[m3.rs][m3.cs+1] = a*f + b*h;
        m3.a[m3.rs+1][m3.cs] = c*e + d*g;
        m3.a[m3.rs+1][m3.cs+1] = c*f + d*h;

      return m3;

    /*When N > 2*/

    result.ce = result.re = n 1;

    A = B = C = D = m1;
    E = F = G = H = m2;
```

```
/*Dividing the matrices*/
A.rs = m1.rs;
A.re = m1.re/2;
A.cs = m1.cs;
A.ce = m1.ce/2;

B.rs = m1.rs;
B.re = m1.re/2;
B.cs = m1.ce/2 + 1;
B.ce = m1.ce;

C.rs = m1.re/2 + 1;
C.re = m1.re;
C.cs = m1.cs;
C.ce = m1.ce/2;

D.rs = m1.re/2 + 1;
D.re = m1.re;
D.cs = m1.ce/2 + 1;
D.ce = m1.ce;

E.rs = m2.rs;
E.re = m2.re/2;
E.cs = m2.cs;
E.ce = m2.ce/2;

F.rs = m2.rs;
F.re = m2.re/2;
F.cs = m2.ce/2 + 1;
F.ce = m2.ce;

G.rs = m2.re/2 + 1;
G.re = m2.re;
G.cs = m2.cs;
G.ce = m2.ce/2;

H.rs = m2.re/2 + 1;
H.re = m2.re;
H.cs = m2.ce/2 + 1;
H.ce = m2.ce;

/*Strassen's Formulae*/

P1 = multiply(A, minus(F, H));
```

```
    P2 = multiply(plus(A, B), H);
    P3 = multiply(plus(C, D), E);
    P4 = multiply(D, minus(G, E));

    P6 = multiply(minus(B, D), plus(G, H));
    P7 = multiply(minus(A, C), plus(E, F));


    Q2 = plus(P1, P2);

    Q4 = minus(minus(plus(P1, P5), P3), P7);

    for (m1_i=Q1.rs, i=0 ; m1_i<=Q1.re ; m1_i++, i++)
     for (m1_j=Q1.cs, j=0 ; m1_j<=Q1.ce ; m1_j++, j++)
          result.a[i][j] = Q1.a[m1_i][m1_j];

    for (m1_i=Q2.rs, i=0 ; m1_i<=Q2.re ; m1_i++, i++)
     for (m1_j=Q2.cs, j=n/2 ; m1_j<=Q2.ce ; m1_j++, j++)
          result.a[i][j] = Q2.a[m1_i][m1_j];

    for (m1_i=Q3.rs, i=n/2 ; m1_i<=Q3.re ; m1_i++, i++)
     for (m1_j=Q3.cs, j=0 ; m1_j<=Q3.ce ; m1_j++, j++)
          result.a[i][j] = Q3.a[m1_i][m1_j];

    for (m1_i=Q4.rs, i=n/2 ; m1_i<=Q4.re ; m1_i++, i++)
     for (m1_j=Q4.cs, j=n/2 ; m1_j<=Q4.ce ; m1_j++, j++)
          result.a[i][j] = Q4.a[m1_i][m1_j];
    return result;
}
```