EXPERIMENT 6

20CP209P – Design and Analysis of Algorithm Lab

Aim:

Implement both a standard $O(n$ matrix multiplication algorithm and Strassen's matrix 3) multiplication algorithm. Using empirical testing, try and estimate the constant factors for the runtime equations of the two algorithms. How big must n be before Strassen's algorithm becomes more efficient than the standard algorithm?

Code:

Strassen's Algorithm:

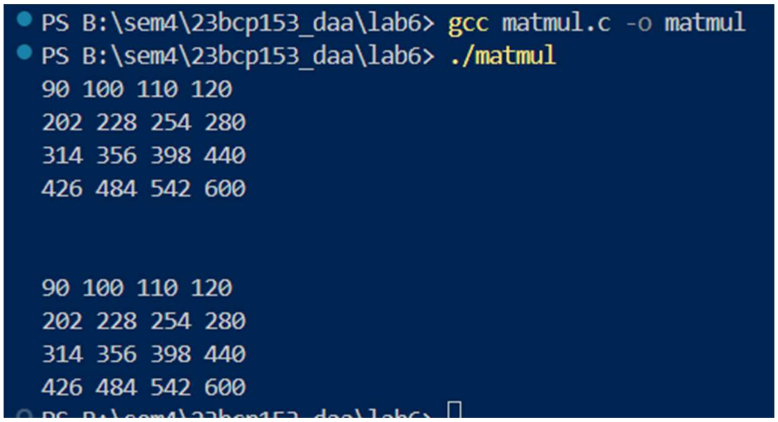
```
#include <stdio.h>
#include <stdlib.h>

#define fr(i, a, b) for (int i = a; i < b; i++)

void matmul(int arra[4][4], int arrb[4][4], int arrc[4][4]);
void add_matrix(int size, int a[size][size], int b[size][size], int c[size][size]);
void sub_matrix(int size, int a[size][size], int b[size][size], int c[size][size]);
void strassen_multiply(int size, int a[size][size], int b[size][size], int c[size][size]);
void strassen_4x4(int A[4][4], int B[4][4], int C[4][4]);

int main(void)
{
   int arra[4][4] = {{1, 2, 3, 4},
                  {5, 6, 7, 8},
                  {9, 10, 11, 12},
                  {13, 14, 15, 16}};
   int arrb[4][4] = {{1, 2, 3, 4},
                  {5, 6, 7, 8},
                  {9, 10, 11, 12},
                  {13, 14, 15, 16}};
   int arrc[4][4];
   matmul(arra, arrb, arrc);
   fr(i, 0, 4)
   {
      fr(j, 0, 4)
      {
         printf("%d ", arrc[i][j]);
      }
      printf("\n");
   }
   printf("\n"); printf("\n");
   strassen_4x4(arra, arrb, arrc);
   fr(i, 0, 4)
```

```
    {
        fr(j, 0, 4)
        {
            printf("%d ", arrc[i][j]);
        }
        printf("\n");
    }
    return 0;
}

void matmul(int arra[4][4], int arrb[4][4], int arrc[4][4])
{
    fr(i, 0, 4)
    {
        fr(j, 0, 4)
        {
            arrc[i][j] = 0;
            fr(k, 0, 4)
            {
                arrc[i][j] += arra[i][k] * arrb[k][j];
            }
        }
    }
    return;
}

void add_matrix(int size, int a[size][size], int b[size][size], int c[size][size])
{
    fr(i, 0, size)
    {
        fr(j, 0, size)
        {
            c[i][j] = a[i][j] + b[i][j];
        }
    }
}

void sub_matrix(int size, int a[size][size], int b[size][size], int c[size][size])
{
    fr(i, 0, size)
    {
        fr(j, 0, size)
        {
            c[i][j] = a[i][j] - b[i][j];
        }
    }
}

void strassen_multiply(int size, int a[size][size], int b[size][size], int c[size][size])
{
```

```
if (size == 2)
{
    // this is base case for final 2x2 mat
    c[0][0] = a[0][0] * b[0][0] + a[0][1] * b[1][0];
    c[0][1] = a[0][0] * b[0][1] + a[0][1] * b[1][1];
    c[1][0] = a[1][0] * b[0][0] + a[1][1] * b[1][0];
    c[1][1] = a[1][0] * b[0][1] + a[1][1] * b[1][1];
}

else
{
    int new_size = size / 2;

    int a11[2][2], a12[2][2], a21[2][2], a22[2][2];
    int b11[2][2], b12[2][2], b21[2][2], b22[2][2];
    int c11[2][2], c12[2][2], c21[2][2], c22[2][2];

    int M1[2][2], M2[2][2], M3[2][2], M4[2][2], M5[2][2], M6[2][2], M7[2][2];
    int temp1[2][2], temp2[2][2];

    fr(i, 0, new_size)
    {
        fr(j, 0, new_size)
        {
            a11[i][j] = a[i][j];
            a12[i][j] = a[i][j + new_size];
            a21[i][j] = a[i + new_size][j];
            a22[i][j] = a[i + new_size][j + new_size];

            b11[i][j] = b[i][j];
            b12[i][j] = b[i][j + new_size];
            b21[i][j] = b[i + new_size][j];
            b22[i][j] = b[i + new_size][j + new_size];
        }
    }

    add_matrix(new_size, a11, a22, temp1);
    add_matrix(new_size, b11, b22, temp2);
    strassen_multiply(new_size, temp1, temp2, M1);

    add_matrix(new_size, a21, a22, temp1);
    strassen_multiply(new_size, temp1, b11, M2);

    sub_matrix(new_size, b12, b22, temp1);
    strassen_multiply(new_size, a11, temp1, M3);

    sub_matrix(new_size, b21, b11, temp1);
    strassen_multiply(new_size, a22, temp1, M4);

    add_matrix(new_size, a11, a12, temp1);
```

```
        strassen_multiply(new_size, temp1, b22, M5);

        sub_matrix(new_size, a21, a11, temp1);
        add_matrix(new_size, b11, b12, temp2);
        strassen_multiply(new_size, temp1, temp2, M6);

        sub_matrix(new_size, a12, a22, temp1);
        add_matrix(new_size, b21, b22, temp2);
        strassen_multiply(new_size, temp1, temp2, M7);

        add_matrix(new_size, M1, M4, temp1);
        sub_matrix(new_size, temp1, M5, temp2);
        add_matrix(new_size, temp2, M7, c11);

        add_matrix(new_size, M3, M5, c12);

        add_matrix(new_size, M2, M4, c21);

        sub_matrix(new_size, M1, M2, temp1);
        add_matrix(new_size, temp1, M3, temp2);
        add_matrix(new_size, temp2, M6, c22);

        fr(i, 0, new_size)
        {
            fr(j, 0, new_size)
            {
                c[i][j] = c11[i][j];
                c[i][j + new_size] = c12[i][j];
                c[i + new_size][j] = c21[i][j];
                c[i + new_size][j + new_size] = c22[i][j];
            }
        }
    }
}

void strassen_4x4(int A[4][4], int B[4][4], int C[4][4])
{
    strassen_multiply(4, A, B, C);
}
```

Output:



```
PS B:\sem4\23bcp153_daa\lab6> gcc matmul.c -o matmul
PS B:\sem4\23bcp153_daa\lab6> ./matmul
90 100 110 120
202 228 254 280
314 356 398 440
426 484 542 600


90 100 110 120
202 228 254 280
314 356 398 440
426 484 542 600
PS B:\sem4\23bcp153_daa\lab6>
```