14.Strassen matrix

#include <stdio.h>

#define N 2 // Matrix size (change to the desired size)

void add\_matrices(int matrix1[N][N], int matrix2[N][N], int result[N][N]) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

result[i][j] = matrix1[i][j] + matrix2[i][j];

}

}

}

void subtract\_matrices(int matrix1[N][N], int matrix2[N][N], int result[N][N]) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

result[i][j] = matrix1[i][j] - matrix2[i][j];

}

}

}

void copy\_matrix(int source[N][N], int destination[N][N]) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

destination[i][j] = source[i][j];

}

}

}

int strassen\_multiply(int matrix1[N][N], int matrix2[N][N], int result[N][N]) {

// Base case: If matrix size is 1, perform regular multiplication

if (N == 1) {

result[0][0] = matrix1[0][0] \* matrix2[0][0];

return 0;

}

int n\_div\_2 = N / 2;

// Sub-matrices

int a11[n\_div\_2][n\_div\_2], a12[n\_div\_2][n\_div\_2], a21[n\_div\_2][n\_div\_2], a22[n\_div\_2][n\_div\_2];

int b11[n\_div\_2][n\_div\_2], b12[n\_div\_2][n\_div\_2], b21[n\_div\_2][n\_div\_2], b22[n\_div\_2][n\_div\_2];

int p1[n\_div\_2][n\_div\_2], p2[n\_div\_2][n\_div\_2], p3[n\_div\_2][n\_div\_2], p4[n\_div\_2][n\_div\_2];

int p5[n\_div\_2][n\_div\_2], p6[n\_div\_2][n\_div\_2], p7[n\_div\_2][n\_div\_2];

// Copy sub-matrices

for (int i = 0; i < n\_div\_2; i++) {

for (int j = 0; j < n\_div\_2; j++) {

a11[i][j] = matrix1[i][j];

a12[i][j] = matrix1[i][j + n\_div\_2];

a21[i][j] = matrix1[i + n\_div\_2][j];

a22[i][j] = matrix1[i + n\_div\_2][j + n\_div\_2];

b11[i][j] = matrix2[i][j];

b12[i][j] = matrix2[i][j + n\_div\_2];

b21[i][j] = matrix2[i + n\_div\_2][j];

b22[i][j] = matrix2[i + n\_div\_2][j + n\_div\_2];

}

}

// Recursive calls for sub-problems

add\_matrices(a11, a22, p1);

add\_matrices(b11, b22, p2);

subtract\_matrices(a21, a11, p3);

subtract\_matrices(b22, b11, p4);

add\_matrices(a11, a12, p5);

add\_matrices(b21, b22, p6 