CIENCIAS DE LA COMPUTACIÓN

MULTIPLICACIÓN DE MATRICES

ANALISIS Y DISEÑO DE ALGORITMOS

Villanueva Borda, Harold

Alejandro

5º SEMESTRE

2022

“Los alumnos declaran haber realizado el presente trabajo de acuerdo a las normas de la Universidad Católica San Pablo”

```cpp
#include <iostream>
#include <fstream>

using namespace std;

int** create_matrix(int rows, int columns) {
    int** matrix = new int*[rows];

    for(int i = 0; i < rows; ++i) {
        *(matrix + i) = new int[columns];

        for(int j = 0; j < columns; ++j) {
            *(*(matrix + i) + j ) = 0;
        }
    }

    return matrix;
}

void standard_algorithm(int** A, int** B, int** C, int m, int n, int o) {

    int product = 0;
    for(int i = 0; i < m; ++i) {
        for(int j = 0; j < o; ++j) {
            for(int k = 0; k < n; ++k) {
                product = (*(*(A + i) + k)) * (*(*(B + k) + j));
                *(*(C + i) + j) += product;
            }
        }
    }
}

void adition(int** A, int** B, int** C, int N){
    for (int i = 0; i < N; ++i)
        for (int j = 0; j < N; ++j){
            *(*(C + i) + j ) = *(*(A + i) + j ) + *(*(B + i) + j );
        }
}

void substraction(int** A, int** B, int** C, int N){
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++){
            *(*(C + i) + j ) = *(*(A + i) + j ) - *(*(B + i) + j );
        }
}

void strassen_algorithm(int** A, int** B, int** C, int N){
    if (N == 1)
        *(*(C + 0) + 0 ) =  *(*(A + 0) + 0 ) *  *(*(B + 0) + 0);

    else{
        int **a11 = new int*[(N / 2)];
        int **a12 = new int*[(N / 2)];
        int **a21 = new int*[(N / 2)];
        int **a22 = new int*[(N / 2)];

        int **b11 = new int*[(N / 2)];
        int **b12 = new int*[(N / 2)];
        int **b21 = new int*[(N / 2)];
        int **b22 = new int*[(N / 2)];

        int **c11 = new int*[(N / 2)];
        int **c12 = new int*[(N / 2)];
        int **c21 = new int*[(N / 2)];
        int **c22 = new int*[(N / 2)];

        int **temp1 = new int*[(N / 2)];
        int **temp2 = new int*[(N / 2)];
```

```
        int **m1 = new int*[(N / 2)];
        int **m2 = new int*[(N / 2)];
        int **m3 = new int*[(N / 2)];
        int **m4 = new int*[(N / 2)];
        int **m5 = new int*[(N / 2)];
        int **m6 = new int*[(N / 2)];
        int **m7 = new int*[(N / 2)];

        for (int i = 0; i < (N / 2); ++i){
            a11[i] = new int[(N / 2)];
            a12[i] = new int[(N / 2)];
            a21[i] = new int[(N / 2)];
            a22[i] = new int[(N / 2)];

            b11[i] = new int[(N / 2)];
            b12[i] = new int[(N / 2)];
            b21[i] = new int[(N / 2)];
            b22[i] = new int[(N / 2)];

            c11[i] = new int[(N / 2)];
            c12[i] = new int[(N / 2)];
            c21[i] = new int[(N / 2)];
            c22[i] = new int[(N / 2)];

            temp1[i] = new int[(N / 2)];
            temp2[i] = new int[(N / 2)];

            m1[i] = new int[N / 2];
            m2[i] = new int[N / 2];
            m3[i] = new int[N / 2];
            m4[i] = new int[N / 2];
            m5[i] = new int[N / 2];
            m6[i] = new int[N / 2];
            m7[i] = new int[N / 2];
        }

        for (int i = 0; i < (N / 2); ++i)
            for (int j = 0; j < (N / 2); ++j){
                a11[i][j] =  *(*(A + i) + j );
                a12[i][j] = A[i][j + (N / 2)];
                a21[i][j] = A[i + (N / 2)][j];
                a22[i][j] = A[i + (N / 2)][j + (N / 2)];

                b11[i][j] =  *(*(B + i) + j );
                b12[i][j] = B[i][j + (N / 2)];
                b21[i][j] = B[i + (N / 2)][j];
                b22[i][j] = B[i + (N / 2)][j + (N / 2)];
            }


        adition(a11, a22, temp1, (N / 2));
        adition(b11, b22, temp2, (N / 2));
        strassen_algorithm(temp1, temp2, m1, (N / 2));

        adition(a21, a22, temp1, (N / 2));
        strassen_algorithm(temp1, b11, m2, (N / 2));

        substraction(b12, b22, temp1, (N / 2));
        strassen_algorithm(a11, temp1, m3, (N / 2));

        substraction(b21, b11, temp1, (N / 2));
        strassen_algorithm(a22, temp1, m4, (N / 2));

        adition(a11, a12, temp1, (N / 2));
        strassen_algorithm(temp1, b22, m5, (N / 2));

        substraction(a21, a11, temp1, (N / 2));
```

```
        adition(b11, b12, temp2, (N / 2));
        strassen_algorithm(temp1, temp2, m6, (N / 2));

        substraction(a12, a22, temp1, (N / 2));
        adition(b21, b22, temp2, (N / 2));
        strassen_algorithm(temp1, temp2, m7, (N / 2));

        adition(m1, m4, temp1, N / 2);
        adition(temp1, m7, temp2, N / 2);
        substraction(temp2, m5, c11, N / 2);

        adition(m3, m5, c12, N / 2);

        adition(m2, m4, c21, N / 2);

        adition(m1, m3, temp1, N / 2);
        adition(temp1, m6, temp2, N / 2);
        substraction(temp2, m2, c22, N / 2);

        for (int i = 0; i < N / 2; ++i)
            for (int j = 0; j < N / 2; ++j){
                *(*(C + i) + j ) = c11[i][j];
                C[i][j + (N / 2)] = c12[i][j];
                C[i + (N / 2)][j] = c21[i][j];
                C[i + (N / 2)][j + (N / 2)] = c22[i][j];
            }



        for (int i = 0; i < (N / 2); ++i){
            delete[] a11[i];
            delete[] a12[i];
            delete[] a21[i];
            delete[] a22[i];

            delete[] b11[i];
            delete[] b12[i];
            delete[] b21[i];
            delete[] b22[i];

            delete[] c11[i];
            delete[] c12[i];
            delete[] c21[i];
            delete[] c22[i];

            delete[] temp1[i];
            delete[] temp2[i];

        }
    }
}

void print(int** matrix, int rows, int columns) {

    for(int i = 0; i < rows; ++i) {
        for(int j = 0; j < columns; ++j) {
            cout << *(*(matrix + i) + j ) << "\t";
        }
        cout<<endl;
    }
}

int main(){
    int **A, **B, **C, m = 4, n = 4, o = 4;

    ifstream matrixA, matrixB;
    matrixA.open("C:/Users/win 10/Documents/SUBLIME/array.txt");
    matrixB.open("C:/Users/win 10/Documents/SUBLIME/arrayB.txt");
```

```
    if(!matrixA.is_open() and !matrixB.is_open()){
        cout << "Error" << endl;
        return 1;
    }

    A = create_matrix(m, n);
    B = create_matrix(n, o);
    C = create_matrix(m, o);

    for(int i = 0; i < m; ++i)
        for(int j = 0; j < n; ++j)
            matrixA >> *(*(A + i) + j );

    for(int i = 0; i < n; ++i)
        for(int j = 0; j < o; ++j)
            matrixB >> *(*(B + i) + j );

    standard_algorithm(A, B, C, m, n, o);
    cout << endl;
    strassen_algorithm(A, B, C, m);


    print(A, m, n);
    cout << endl;
    print(B, n, o);
    cout << endl;
    print(C, m, o);

    for (int i = 0; i < m; ++i)
        delete[] A[i];
    delete[] A;

    for (int i = 0; i < n; ++i)
        delete[] B[i];
    delete[] B;

    for (int i = 0; i < n; ++i)
        delete[] C[i];
    delete[] C;
}
```

```
multiplicacion strassen:
1       2       3       4
5       6       7       8
9       10      11      12
13      14      15      16

16      15      14      13
12      11      10      9
8       7       6       5
4       3       2       1

80      70      60      50
240     214     188     162
400     358     316     274
560     502     444     386
```

```
multiplicacion clasica:
1       2       3
4       5       6
7       8       9
10      11      12

9       8       7       6
5       4       3       2
1       1       2       3

22      19      19      19
67      58      55      52
112     97      91      85
157     136     127     118
```

```
multiplicacion clasica:
1       2       3       4
5       6       7       8
9       10      11      12


9       8       7
6       5       4
3       2       1
1       2       3


34      32      30
110     100     90
186     168     150
```

```
multiplicacion strassen:
1       2       3       4       5       6       7       8
9       10      11      12      13      14      15      16
1       2       3       4       5       6       7       8
9       10      11      12      13      14      15      16
1       2       3       4       5       6       7       8
9       10      11      12      13      14      15      16
1       2       3       4       5       6       7       8
9       10      11      12      13      14      15      16


9       8       7       6       5       4       3       2
1       1       2       3       13      14      7       4
9       8       7       6       5       4       3       2
1       1       2       3       13      14      7       4
9       8       7       6       5       4       3       2
1       1       2       3       13      14      7       4
9       8       7       6       5       4       3       2
1       1       2       3       13      14      7       4
```

164	148	152	156	340	344	188	112
484	436	440	444	916	920	508	304
164	148	152	156	340	344	188	112
484	436	440	444	916	920	508	304
164	148	152	156	340	344	188	112
484	436	440	444	916	920	508	304
164	148	152	156	340	344	188	112
484	436	440	444	916	920	508	304