Алгоритмы и структуры данных в системах искусственного интеллекта		
Кафедра ЭВМ и С Лабораторная работа №2 «Умножение матриц на примере сравнения обычного алгоритма и блочного алгоритма Штрассена»	ФИО студента	Рюмин Д.Л.
	Группа	ЭВМ 1.2
	Дата выполнения	02.05.2023
	Дата отчета	
	Оценка (баллы)	
	Подпись преподавателя с расшифровкой	

Задание:

1. Реализовать умножение матриц "в лоб" за O(n^3).

2. Определить время работы для матриц, размерности которых яввляются степенями двойки : 16,

32, 64, 128, 256, 512, 1024, 2048, 4096. Можно выборочно (небольшие, средние, большие).

3. Реализовать алгоритм Штрассена и исследовать его быстродействие на тех же размерностях.

Выполнение:

	Время (с.)	
	Multiple	**Strassen**
4	0,000005	0,0000021
8	0,0000025	0,000003
16	0,000017	0,000015
32	0,00014	0,00012
64	0,0011	0,00087
128	0,0094	0,0083
256	0,082	0,072
512	0,82	0,86
1 024	8,08	5,79
2 048	87	40,17
4 094	775	356,47

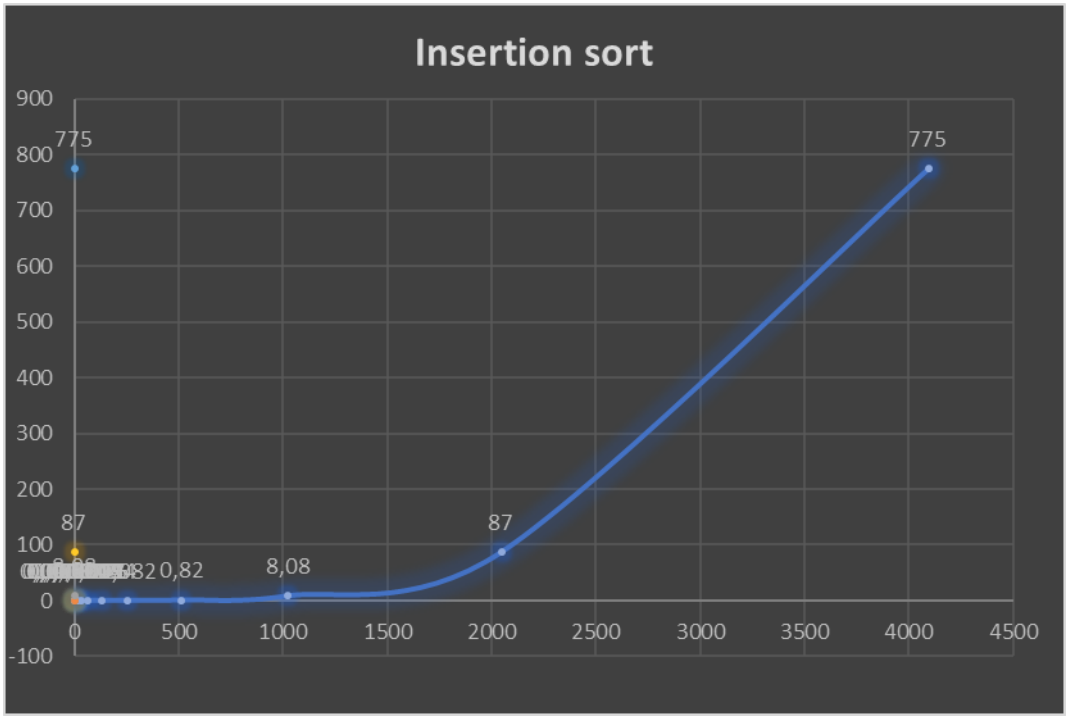

Рисунок 1. Зависимость времени выполнения программы (умножение матриц) от размерности матрицы.

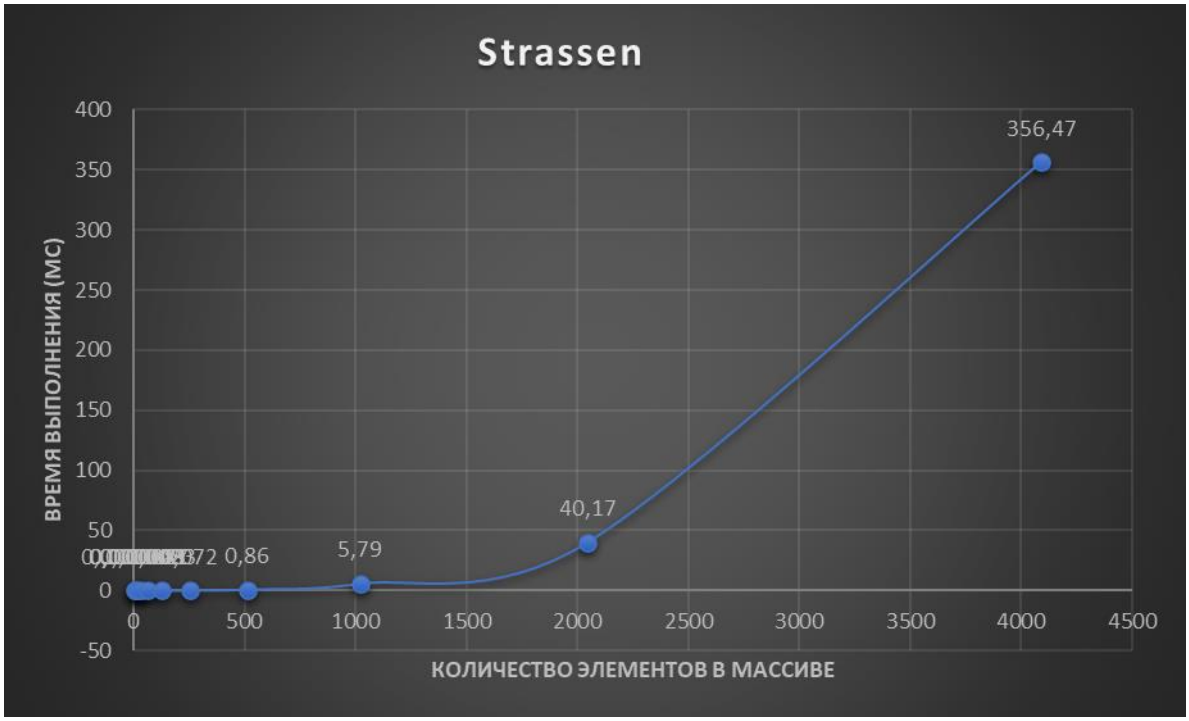

Рисунок 2. Зависимость времени выполнения программы (умножение матриц методом Штрассена) от размерности матрицы.

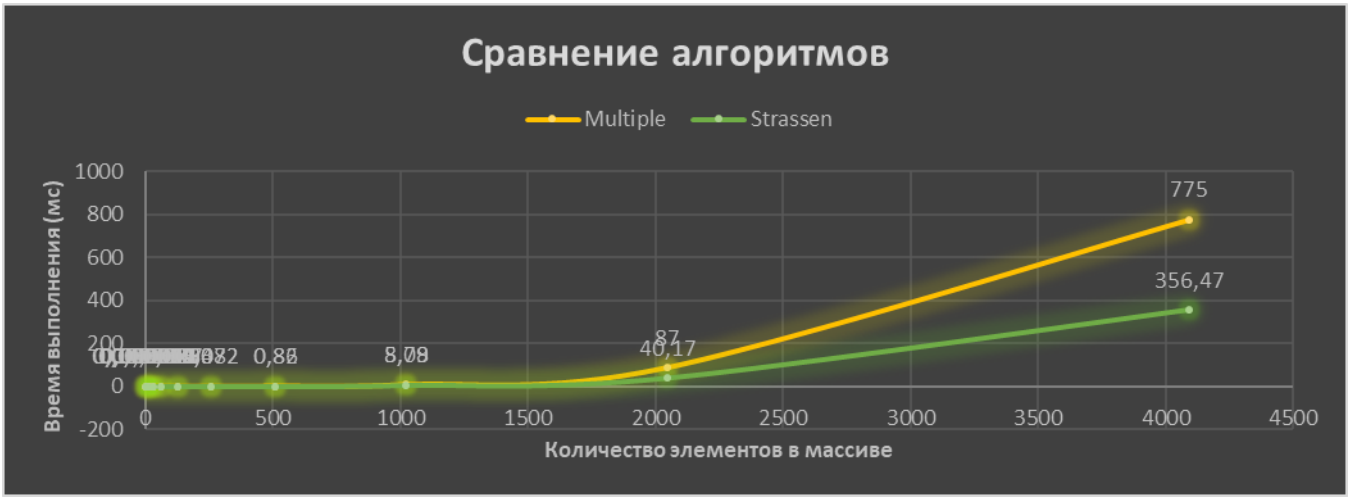

Рисунок 3. Сравнение обычного алгоритма умножения матриц и алгоритма умножение матриц методом Штрассена.

Код программы:

Matr.cpp

```
#include <string>
#include <iostream>
#include <ctime>
#include <chrono>

using namespace std;

int N = 4096;

void multiply(int** A, int** B, int** result)
{
    int i, j, k;

    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            result[i][j] = 0;
            for (k = 0; k < N; k++)
                result[i][j] += A[i][k] * B[k][j];
        }
    }

    /*

    cout << endl << "Результат умножения матриц A x B: " << endl << endl;
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++)
            cout << result[i][j] << " ";
        cout << endl;
    }

    */
}


int main()
{
    setlocale(LC_ALL, "RUS");

    srand(time(NULL));

    int** MatrA = new int* [N];
    int** MatrB = new int* [N];
    int** result = new int* [N];

    for (int j = 0; j < N; j++) {
        MatrA[j] = new int[N];
        MatrB[j] = new int[N];
        result[j] = new int[N];
        for (int k = 0; k < N; k++) {
            MatrA[j][k] = rand() % 100;
            MatrB[j][k] = rand() % 100;
        }
    }

    /*

    cout << "Матрица A (" << N << "x" << N << "): " << endl << endl;
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++)
            cout << MatrA[i][j] << " ";
        cout << endl;
    }

    cout << endl;
```

```
    cout << "Матрица B (" << N << "x" << N << "): " << endl << endl;
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++)
            cout << MatrB[i][j] << " ";
        cout << endl;
    }

    */

    auto begin = chrono::steady_clock::now();

    multiply(MatrA, MatrB, result);

    auto end = chrono::steady_clock::now();

    auto elapsed_ms = chrono::duration_cast<chrono::nanoseconds>(end - begin);


    cout << "\nВремя выполнения: " << elapsed_ms.count() << " ns\n";
}
```

Strassen.cpp

```
#include <string>
#include <iostream>
#include <ctime>
#include <chrono>

using namespace std;
int N = 4094;

int* multiply(int* A, int* B, int n) {
    int* C = new int[n * n];
    for (int i = 0; i < n * n; i = i + n) {
        for (int j = 0; j < n; j++) {
            C[i + j] = 0;
            for (int k = 0; k < n; k++) {
                C[i + j] += A[i + k] * B[k * n + j];
            }
        }
    }
    return C;
}

int* add(int* A, int* B, int n) {
    int* C = new int[n * n];
    for (int i = 0; i < n * n; i = i + n) {
        for (int j = 0; j < n; j++) {
            C[i + j] = A[i + j] + B[i + j];
        }
    }
    return C;
}

int* subtract(int* A, int* B, int n) {
    int* C = new int[n * n];
    for (int i = 0; i < n * n; i = i + n) {
        for (int j = 0; j < n; j++) {
            C[i + j] = A[i + j] - B[i + j];
        }
    }
    return C;
}

void splitMatrix(int* A, int* A11, int* A12, int* A21, int* A22, int n)
{
    int k = n / 2;
```

```
    for (int i = 0; i < k; i++) {
        for (int j = 0; j < k; j++) {
            A11[i * k + j] = A[i * n + j];
            A12[i * k + j] = A[i * n + j + k];
            A21[i * k + j] = A[(i + k) * n + j];
            A22[i * k + j] = A[(i + k) * n + j + k];
        }
    }
}

int* collectMatrix(int* A11, int* A12, int* A21, int* A22, int n) {
    int* A = new int[n * n];
    int k = n / 2;
    for (int i = 0; i < k; i++) {
        for (int j = 0; j < k; j++) {
            A[i * n + j] = A11[i * k + j];
            A[i * n + j + k] = A12[i * k + j];
            A[(i + k) * n + j] = A21[i * k + j];
            A[(i + k) * n + j + k] = A22[i * k + j];
        }
    }
    return A;
}

int* strassen(int* A, int* B, int n) {
    int* C = new int[n * n];
    if (n < 1024) {
        C = multiply(A, B, n);
        return C;
    }
    int k = n / 2;

    int* A11 = new int[k * k];
    int* A12 = new int[k * k];
    int* A21 = new int[k * k];
    int* A22 = new int[k * k];

    int* B11 = new int[k * k];
    int* B12 = new int[k * k];
    int* B21 = new int[k * k];
    int* B22 = new int[k * k];

    int* P1 = new int[k * k];
    int* P2 = new int[k * k];
    int* P3 = new int[k * k];
    int* P4 = new int[k * k];
    int* P5 = new int[k * k];
    int* P6 = new int[k * k];
    int* P7 = new int[k * k];

    int* C11 = new int[k * k];
    int* C12 = new int[k * k];
    int* C21 = new int[k * k];
    int* C22 = new int[k * k];

    splitMatrix(A, A11, A12, A21, A22, n);
    splitMatrix(B, B11, B12, B21, B22, n);

    P1 = strassen(add(A11, A22, k), add(B11, B22, k), k);
    P2 = strassen(add(A21, A22, k), B11, k);
    P3 = strassen(A11, subtract(B12, B22, k), k);
    P4 = strassen(A22, subtract(B21, B11, k), k);
    P5 = strassen(add(A11, A12, k), B22, k);
    P6 = strassen(subtract(A21, A11, k), add(B11, B12, k), k);
    P7 = strassen(subtract(A12, A22, k), add(B21, B22, k), k);

    C11 = add(subtract(add(P1, P4, k), P5, k), P7, k);
    C12 = add(P3, P5, k);
    C21 = add(P2, P4, k);
```

```
    C22 = add(subtract(P1, P2, k), add(P3, P6, k), k);

    C = collectMatrix(C11, C12, C21, C22, n);

    return C;
}


void printMatr(int* A) {
    for (int i = 0; i < N * N; i = i + N) {
        for (int j = 0; j < N; j++) {
            cout << A[i + j] << " ";
        }
        cout << endl;
    }
}

int main() {

    setlocale(LC_ALL, "RUS");
    srand(time(NULL));

    int* MatrA = new int [N * N];
    int* MatrB = new int [N * N];

    for (int i = 0; i < N * N; i = i + N)
    {
        for (int j = 0; j < N; j++)
        {
            MatrA[i + j] = rand() % 100;
            MatrB[i + j] = rand() % 100;
        }
    }

    /*

    cout << "Матрица A (" << N << "x" << N << "): " << endl << endl;
    printMatr(MatrA);
    cout << endl;

    cout << "Матрица B (" << N << "x" << N << "): " << endl << endl;
    printMatr(MatrB);
    cout << endl;

    */

    auto start = chrono::steady_clock::now();
    int* result = strassen(MatrA, MatrB, N);
    auto end = chrono::steady_clock::now();

    auto duration = chrono::duration_cast<chrono::nanoseconds>(end - start);

    /*

    cout << endl << "Результат умножения матриц A x B методом Винограда-Штрассена: " << endl << endl;
    printMatr(result);

    */

    cout << endl;
    cout << "Время выполнения: " << duration.count() << " ns\n";

}
```