Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

Отчет по лабораторной работе №2
По курсу: «Анализ алгоритмов»

По теме: «Алгоритмы умножения матриц»

Студент:
Коротков Андрей Владимирович
Группа: ИУ7-55Б

Преподаватели:
Волкова Лилия Леонидовна
Строганов Юрий Владимирович

Москва, 2019 г.

Содержание

Введение

В данной работе требуется изучить и применить алгоритмы умножения матриц, научиться рассчитывать трудоёмкость алгоритмов, получить навыки в улучшении алгоритмов.

Необходимо реализовать описанные ниже алгоритмы:

1. классический алгоритм умножения матриц;
2. алгоритм Винограда;
3. улучшенный алгоритм Винограда.

1. Аналитическая часть

В данном разделе будет рассмотрено описание алгоритмов умножения матриц.

1.1 Описание алгоритмов

Умножение матриц — одна из основных операций над матрицами. Матрица, получаемая в результате операции умножения, называется произведением матриц. Операция умножения двух матриц выполнима только в том случае, если число столбцов в первом сомножителе равно числу строк во втором; в этом случае говорят, что матрицы согласованы. В частности, умножение всегда выполнимо, если оба сомножителя — квадратные матрицы одного и того же порядка. Таким образом, из существования произведения AB вовсе не следует существование произведения BA.

Эти алгоритмы активно применяются во всех областях, применяющих линейную алгебру, такие как:

- компьютерная графика;
- физика;
- экономика;
- и так далее.

1.1.1 Классический алгоритм умножения матриц

Пусть даны две прямоугольные матрицы A и B размерности $n1 \times m1$ и $n2 \times m2$ соответсвенно:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m_1} \\ a_{21} & a_{22} & \cdots & a_{2m_1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n_1 1} & a_{n_1 2} & \cdots & a_{n_1 m_1} \end{bmatrix}, \quad B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m_2} \\ b_{21} & b_{22} & \cdots & b_{2m_2} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n_2 1} & b_{n_2 2} & \cdots & b_{n_2 m_2} \end{bmatrix}.$$

Тогда матрица C размерностью $l \times n$:

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1m_2} \\ c_{21} & c_{22} & \cdots & c_{2m_2} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n_1 1} & c_{n_1 2} & \cdots & c_{n_1 m_2} \end{bmatrix},$$

в которой:

$$c_{ij} = \sum_{r=1}^{m_1} a_{ir} b_{rj} \quad \left(i = 1,2,\ldots n_1;\ j = 1,2,\ldots m_2\right).$$

называется их произведением.

$$A = \begin{bmatrix} 3 & -1 & 2 \\ 4 & 2 & 0 \\ -5 & 6 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 8 & 1 \\ 7 & 2 \\ 2 & -3 \end{bmatrix}, C = A \times B = \begin{bmatrix} 21 & -5 \\ 46 & 8 \\ 4 & 4 \end{bmatrix}.$$

1.1.2 Алгоритм Винограда

Рассмотрим два вектора V = (v1, v2, v3, v4) и W = (w1, w2, w3, w4). Их скалярное произведение равно:

$$V \bullet W = v1w1 + v2w2 + v3w3 + v4w4 \quad (1)$$

Это равенство можно переписать в виде:

$$V \bullet W = (v1 + w2)(v2 + w1) + (v3 + w4)(v4 + w3) - v1v2 - v3v4 - w1w2 - w3w4 \quad (2)$$

Может показаться, что выражение (2) задает больше работы, чем (1): вместо четырех умножений мы насчитываем их шесть, а вместо трех сложений - десять. Менее очевидно, что выражение в правой части последнего равенства допускает предварительную обработку: его части можно вычислить заранее и запомнить для каждой строки первой матрицы и для каждого столбца второй. Это означает, что над предварительно обработанными элементами нам придется выполнять лишь первые два умножения и последующие пять сложений, а также дополнительно два сложения.

1.1.3 Вывод

Были рассмотрены поверхностно алгоритмы классического умножения матриц и алгоритм Винограда, принципиальная разница которого — наличие предварительной обработки, а также уменьшение количества операций умножения.

2. Конструкторская часть

В данном разделе будут рассмотрены схемы алгоритмов:

- классического умножения;
- Винограда;
- улучшенный алгоритм Винограда.

2.1 Разработка алгоритмов

Ниже на рисунках (1) - (3) приведены схемы алгоритмов умножения матриц:

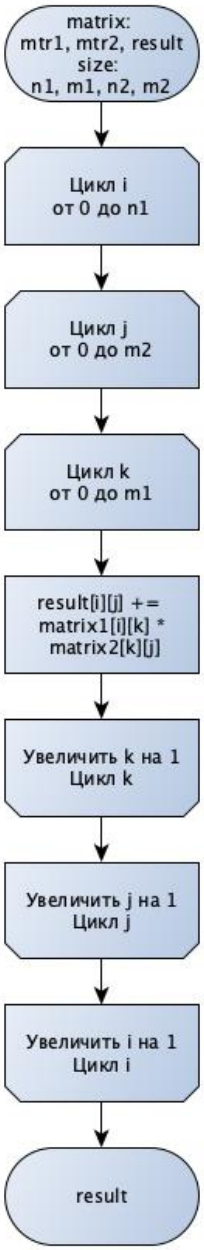

Рисунок 1. Схема алгоритма классического перемножения матриц.

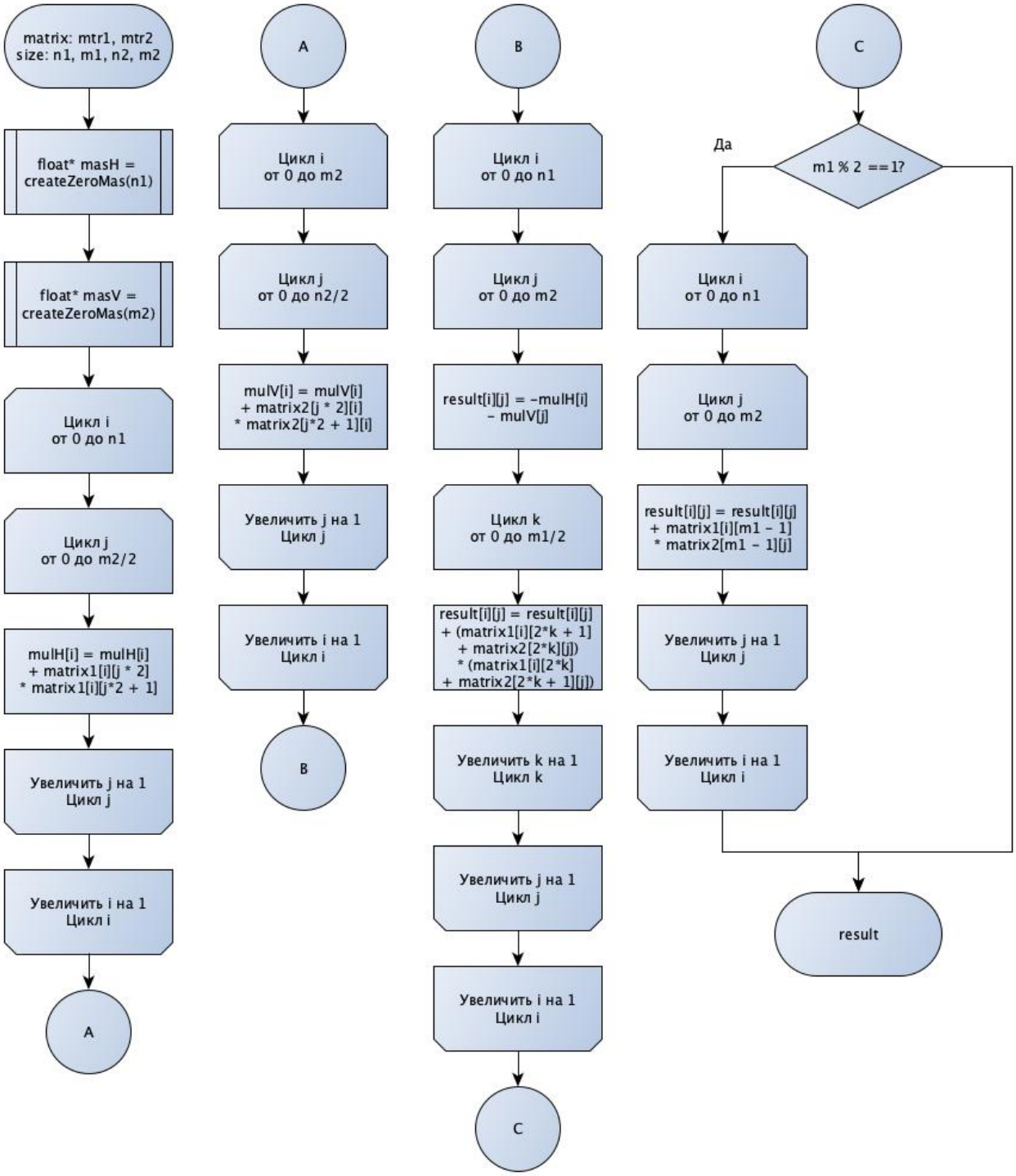

Рисунок 2. Схема алгоритма Винограда по умножению матриц.

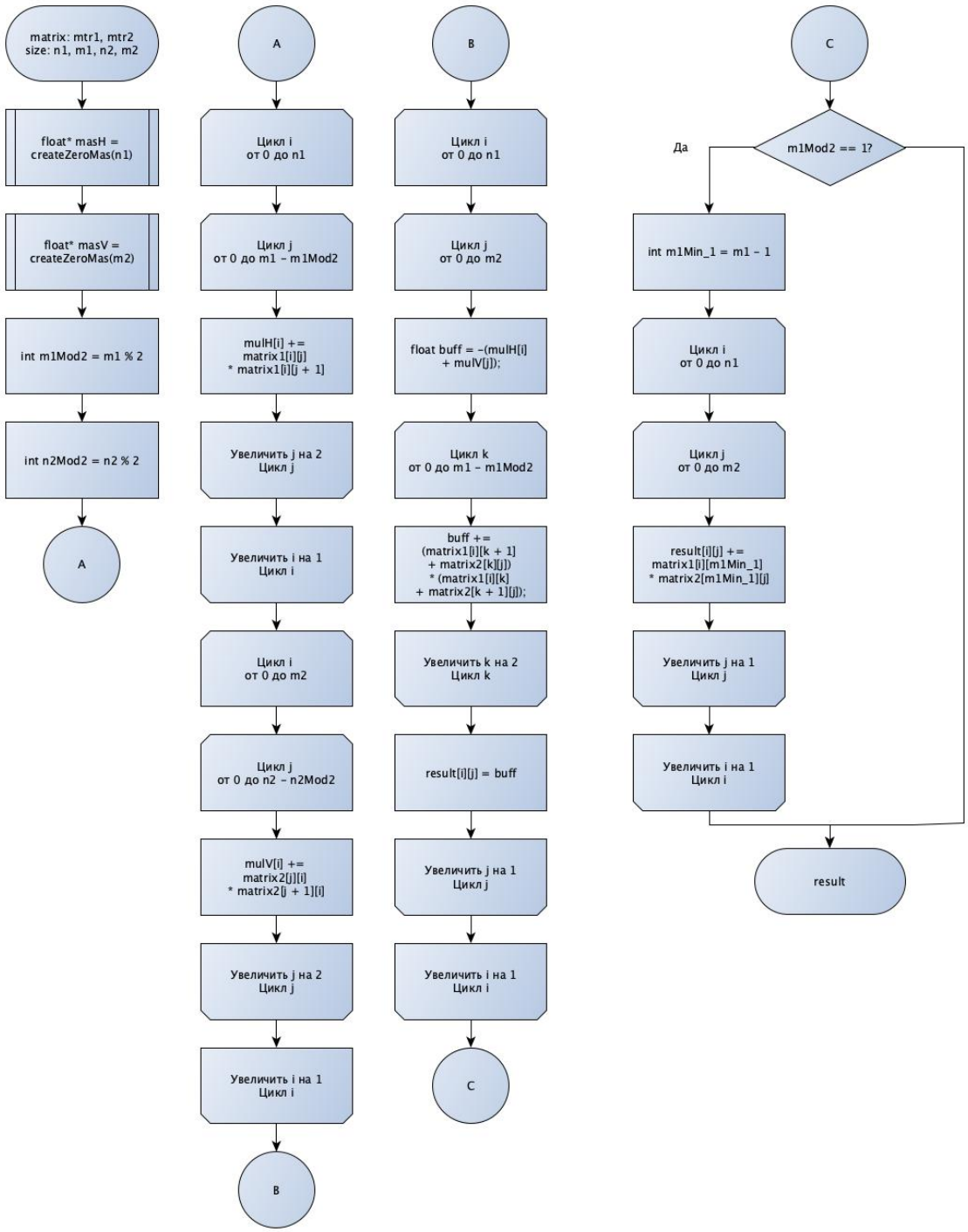

Рисунок 3. Схема усовершенствованного алгоритма Винограда по умножению матриц.

2.2 Оптимизации алгоритма Винограда

Ниже представлены оптимизации для алгоритма Винограда, позволяющие уменьшить трудоемкость алгоритма:

1. Заменить j < m1/2; j++ на j < m1; j+= 2 и j*2 на j. Таким образом можно избавиться от постоянного деления в цикле.
2. Заменить mulH[i] = mulH[i] + … на mulH[i] += …

3. Аналогичны замены для подсчета mulV[i].
4. Внутри тройного цикла накапливать результат в буфер, а вне цикла сбрасывать буфер в ячейку матрицы.

2.3 Трудоемкость алгоритмов

Введем модель трудоемкости для оценки алгоритмов:

1) базовые операции стоимостью 1 — +, -, *, /, =, ==, <=, >=, !=, +=, [];
2) оценка трудоемкости цикла:

Листинг 1. Общий пример цикла в Си.

```
for (int i = 0; i <= N; i++)
{
	//тело цикла
}
```

Fц = 2 + N*(2 + Fтела)

3) стоимость условного перехода возьмем за 0, стоимость вычисления условия остаётся.

В таблицах (1) - (3) приведены оценки трудоемкости алгоритмов.

Таблица 1. Оценка трудоемкости классического алгоритма.

Трудоемкость	Оценка трудоемкости
Fклассич.	2+N1(2 + 2 + M2(2 + 2 + M1(2+Fтела)))
Fтела	8
Fклассич	$(10 * N1 * M2 * M1) + (4 * N1 * M2) + (4 * N1) + 2$

Таблица 2. Оценка трудоемкости алгоритма Винограда.

Трудоемкость	Оценка трудоемкости
Fвиноград	2 + N1(4 + M1/2(3 + Fтела1)) + 2 + M2(4 + N2/2(3 + Fтела2)) + 2 + N1(2 + 2 + M2(2 + Fтела3)) + 2 + Fусловия
Fусловия	$\left[\begin{matrix}2\\2+N1*[2+2+M2*(2+F_{\text{тела}}5)]\end{matrix}\right]$
Fтела1	12
Fтела2	12
Fтела3	2 + 7 + M1/2(3 + Fтела4)
Fтела4	23
Fтела5	13
Fвиноград	8 + 8*N1 + 15/2*N1*M1 + 4*M2 +15/2*M2*N2 + 11*M2*N1 + 13*M1*M2*N1 + $+\left[\begin{matrix}2\\2+N1*4+N1*M2*15\end{matrix}\right]$

Таблица 3. Оценка трудоемкости улучшенного алгоритма Винограда.

Трудоемкость	Оценка трудоемкости
Fулучш.	2 + N1(4 + M1/2(3 + Fтела1)) + 2 + M2(4 + N2/2(3 + Fтела2)) + 2 + N1(2 + 2 + M2(2 + Fтела3)) + 2 + Fусловия
Fусловия	$\left[\frac{1}{1 + N1 * [2 + 2 + M2 * (2 + F_{\text{тела}}5)]}\right]$
Fтела1	8
Fтела2	8
Fтела3	2 + 8 + M1/2(3 + Fтела4)
Fтела4	14
Fтела5	8
Fулучш.	8 + 8*N1 + 11/2*M1*N1 + + 4*M2 + 11/2*N2*M2 + + 12*M2*N1+ 17/2*M1*M2*N1 + $+ \left[\frac{1}{1 + 4 * N1 + 10 * M2 * N1}\right]$

2.4 Вывод

Так как оценка дается по самому быстрому растущему слагаемому, в нашем случае это куб линейного размера матриц. В классическом алгоритме коэффициент равен 10, у Винограда — 13, в улучшенном — 8.5. Можно сделать вывод о том, что улучшенный метод Винограда выигрывает по скорости работы у классического алгоритма умножения матриц.

3. Технологическая часть

В данном разделе будут рассмотрены требования к программному обеспечению, средства реализации и представлен листинг кода.

3.1 Требования к программному обеспечению

Входные данные: mtr1 - первая матриц, mtr2 - вторая матрица.

Выходные данные: произведение двух матриц.

На рис. 4 приведена функциональная схема умножения матриц:

Рисунок 4. IDEF0-диаграмма, описывающая алгоритм умножения двух матриц.

3.2 Средства реализации

В данной работе используется язык программирования «C». Проект выполнен в среде Xcode. Для отключения оптимизации компилятора указан флаг «-o0».

Для замера процессорного времени используется функция, возвращающая количество тиков.

Листинг 2. Функция замера времени.

```
unsigned long long getTicks(void)
{
    unsigned long long d;
    __asm__ __volatile__ ("rdtsc" : "=A" (d) );
    return d;
}
```

3.3 Листинг кода

В данном пункте представлен листинг кода, а именно:

- классический алгоритм умножения матриц;
- алгоритм Винограда;
- улучшенный алгоритм Винограда.

Ниже приведен листинг трех алгоритмов умножения матриц:

Листинг 3. Классический алгоритм умножения матриц.

```
    float** multMatrix(float** matrix1, int n1, int m1, float** matrix2, int
n2, int m2)
    {
        float** result = NULL;
        if (matrix1 && matrix2 && m1 == n2)
        {
            result = createZeroMatrix(n1, m2);
            for (int i = 0; i < n1; i++)
                for (int j = 0; j < m2; j++)
                    for (int k = 0; k < m1; k++)
                        result[i][j] += matrix1[i][k] * matrix2[k][j];
        }

        return result;
    }
```

Листинг 4. Алгоритм умножения матриц Винограда.

```
    float** multMatrixVinogradov(float** matrix1, int n1, int m1, \
                                 float** matrix2, int n2, int m2)
    {
        float* mulH = createZeroMas(n1);
        float* mulV = createZeroMas(m2);
        float** result = createZeroMatrix(n1, m2);

        if (result && mulH && mulV && m1 == n2)
        {
            for (int i = 0; i < n1; i++)
                for (int j = 0; j < m1/2; j++)
                    mulH[i] = mulH[i] + matrix1[i][j * 2]
                                      * matrix1[i][j*2 + 1];

            for (int i = 0; i < m2; i++)
                for (int j = 0; j < n2/2; j++)
                    mulV[i] = mulV[i] + matrix2[j * 2][i]
                                      * matrix2[j*2 + 1][i];

            for (int i = 0; i < n1; i++)
                for (int j = 0; j < m2; j++)
                {
                    result[i][j] = -mulH[i] - mulV[j];
                    for (int k = 0; k < m1/2; k++)
                        result[i][j] = result[i][j] + (matrix1[i][2*k + 1]
                                                    + matrix2[2*k][j])
                                                    * (matrix1[i][2*k]
                                                    + matrix2[2*k + 1][j]);
                }

            if (m1 % 2)
                for (int i = 0; i < n1; i++)
                    for (int j = 0; j < m2; j++)
                        result[i][j] = result[i][j] + matrix1[i][m1 - 1]
                                                    * matrix2[m1 - 1][j];
        }
        else
            freeMemMatrix(result, n1, m2);

        freeMemMas(mulH);
        freeMemMas(mulV);

        return result;
    }
```

Листинг 5. Улучшенный алгоритм умножения матриц Винограда.

```
     float** multMatrixVinogradovFixed(float** matrix1, int n1, int m1, float**
matrix2, int n2, int m2)
     {
         float* mulH = createZeroMas(n1);
         float* mulV = createZeroMas(m2);
         float** result = createZeroMatrix(n1, m2);

         if (result && mulH && mulV && m1 == n2)
         {
             int m1Mod2 = m1 % 2;
             int n2Mod2 = n2 % 2;

             for (int i = 0; i < n1; i++)
                 for (int j = 0; j < (m1 - m1Mod2); j += 2)
                     mulH[i] += matrix1[i][j] * matrix1[i][j + 1];

             for (int i = 0; i < m2; i++)
                 for (int j = 0; j < (n2 - n2Mod2); j += 2)
                     mulV[i] += matrix2[j][i] * matrix2[j + 1][i];

             for (int i = 0; i < n1; i++)
                 for (int j = 0; j < m2; j++)
                 {
                     float buff = -(mulH[i] + mulV[j]);
                     for (int k = 0; k < (m1 - m1Mod2); k += 2)
                         buff += (matrix1[i][k + 1] + matrix2[k][j]) *
                                 (matrix1[i][k] + matrix2[k + 1][j]);

                     result[i][j] = buff;
                 }

             if (m1Mod2)
             {
                 int m1Min_1 = m1 - 1;
                 for (int i = 0; i < n1; i++)
                     for (int j = 0; j < m2; j++)
                         result[i][j] += matrix1[i][m1Min_1]
                                       * matrix2[m1Min_1][j];
             }
         }
         else
             freeMemMatrix(result, n1, m2);

         freeMemMas(mulH);
         freeMemMas(mulV);

         return result;
     }
```

3.4 Вывод

В данном разделе были представлены реализации классического алгоритма, алгоритма Винограда, улучшенного алгоритма Винограда.

4. Экспериментальная часть

В данном разделе будут рассмотрены примеры работы программы.
Также будет проведен эксперимент и сравнительный анализ данных.

4.1 Примеры работы

Проверки работы алгоритмов:

- нулевые матрицы;
- единичные матрицы;
- матрицы, размерностью 0x0.

В листингах (6) - (11) представлены результаты работы программы на разных входных данных:

Листинг 6. Умножение случайных матриц размерностью 2x2.

```
Matr1:
97.000000 65.000000
50.000000 30.000000

Matr2:
80.000000 84.000000
3.000000 38.000000

--------------------------

Mult classic:
7955.000000 10618.000000
4090.000000 5340.000000

Mult Vinograd:
7955.000000 10618.000000
4090.000000 5340.000000

Mult Vinograd Fixed:
7955.000000 10618.000000
4090.000000 5340.000000
```

Листинг 7. Умножение матриц, заполненных нулями, размерностью 2x2

```
Matr1:
0.000000 0.000000
0.000000 0.000000

Matr2:
0.000000 0.000000
0.000000 0.000000

--------------------------

Mult classic:
0.000000 0.000000
0.000000 0.000000

Mult Vinograd:
0.000000 0.000000
0.000000 0.000000

Mult Vinograd Fixed:
0.000000 0.000000
0.000000 0.000000
```

Листинг 8. Умножение матриц, заполненных единицами, размерностью 2x2.

```
Matr1:
1.000000 1.000000
1.000000 1.000000

Matr2:
1.000000 1.000000
1.000000 1.000000

--------------------------

Mult classic:
2.000000 2.000000
2.000000 2.000000

Mult Vinograd:
2.000000 2.000000
2.000000 2.000000

Mult Vinograd Fixed:
2.000000 2.000000
2.000000 2.000000
```

Листинг 9. Умножение матриц, заполненных единицами, размерностью 10x10.

```
Matr1:
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000

Matr2:
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
--------------------------
Mult classic:
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000

Mult Vinograd:
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000

Mult Vinograd Fixed:
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000
```

Листинг 10. Умножение матриц, размерностью 0x0.

```
Matr1:
Empty matrix

Matr2:
Empty matrix

--------------------------

Mult classic:
Empty matrix

Mult Vinograd:
Empty matrix

Mult Vinograd Fixed:
Empty matrix
```

Листинг 11. Умножение квадратных матриц, размерностью 3x3.

```
Matr1:
0.000000 1.000000 2.000000
1.000000 2.000000 3.000000
2.000000 3.000000 4.000000

Matr2:
0.000000 1.000000 2.000000
1.000000 2.000000 3.000000
2.000000 3.000000 4.000000

--------------------------

Mult classic:
5.000000 8.000000 11.000000
8.000000 14.000000 20.000000
11.000000 20.000000 29.000000

Mult Vinograd:
5.000000 8.000000 11.000000
8.000000 14.000000 20.000000
11.000000 20.000000 29.000000

Mult Vinograd Fixed:
5.000000 8.000000 11.000000
8.000000 14.000000 20.000000
11.000000 20.000000 29.000000
```

4.2 Постановка эксперимента

В рамках данного проекта были проведены эксперименты, описанные ниже.

1. Сравнение времени выполнения трех алгоритмов. Создаются квадратные матрицы, заполненные случайными числами. Их размерности изменяются в интервале от 100x100 до 1000x1000 с шагом 100. Эксперимент по измерению одной размерности ставился 100 раз.
2. Сравнение времени выполнения трех алгоритмов. Создаются квадратные матрицы, заполненные случайными числами. Их размерности изменяются в интервале от 101x101 до 1001x1001 с шагом 100. Эксперимент по измерению одной размерности ставился 100 раз.

4.3 Сравнительный анализ на материале экспериментальных данных

Ниже на рисунках (5) - (7) представлены графики работы алгоритмов по умножению квадратных матриц с чётной и нечётной размерностью:

Сравнение времени умножения квадратных матриц с четной размерностью:

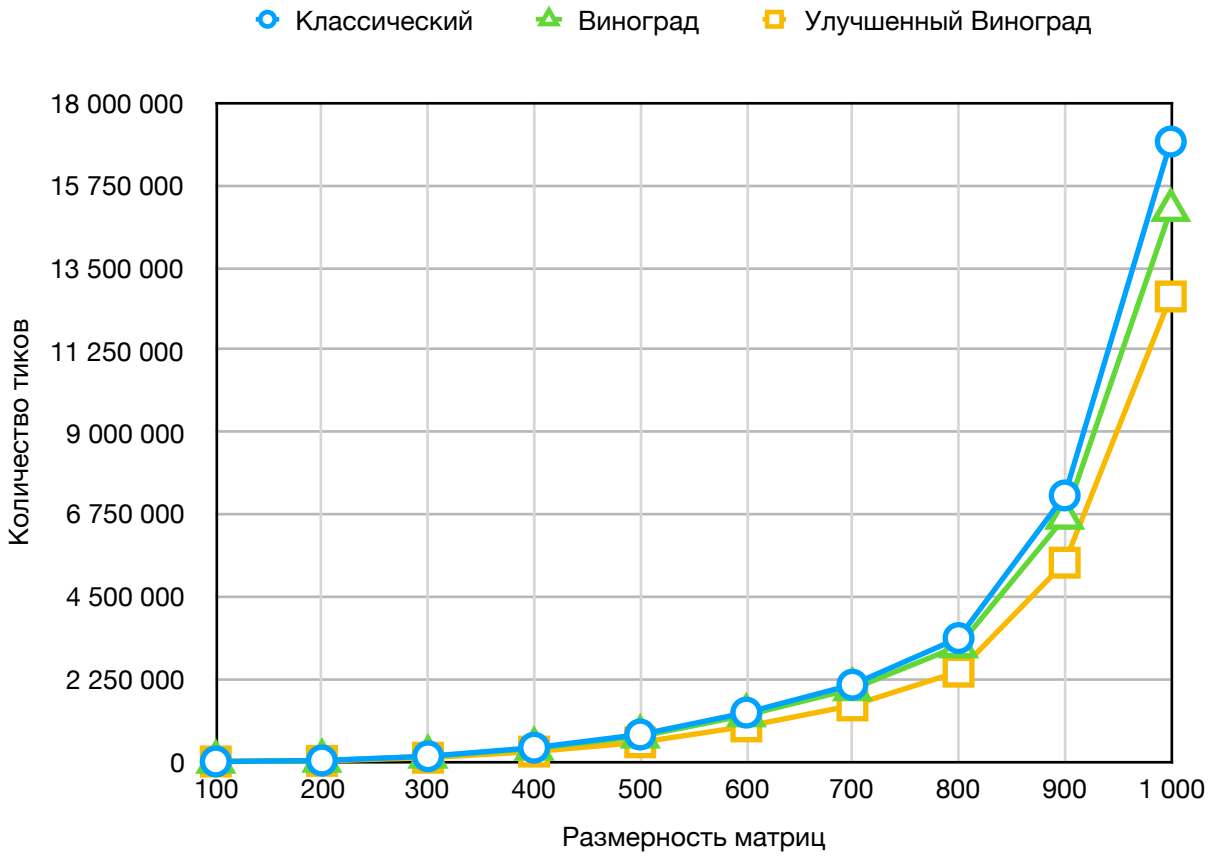

Рисунок 5. График работы алгоритмов по умножению квадратных матриц с четной размерностью (ось абсцисс — время работы алгоритмов (тики), ось ординат — размерность матриц).

Самым медленным алгоритмом оказалось классическое умножение матриц, а самый быстрый — улучшенный алгоритм Винограда. Заметим, что оба алгоритма имеют одинаковый характер.

Сравнение времени умножения квадратных матриц с нечетной размерностью:

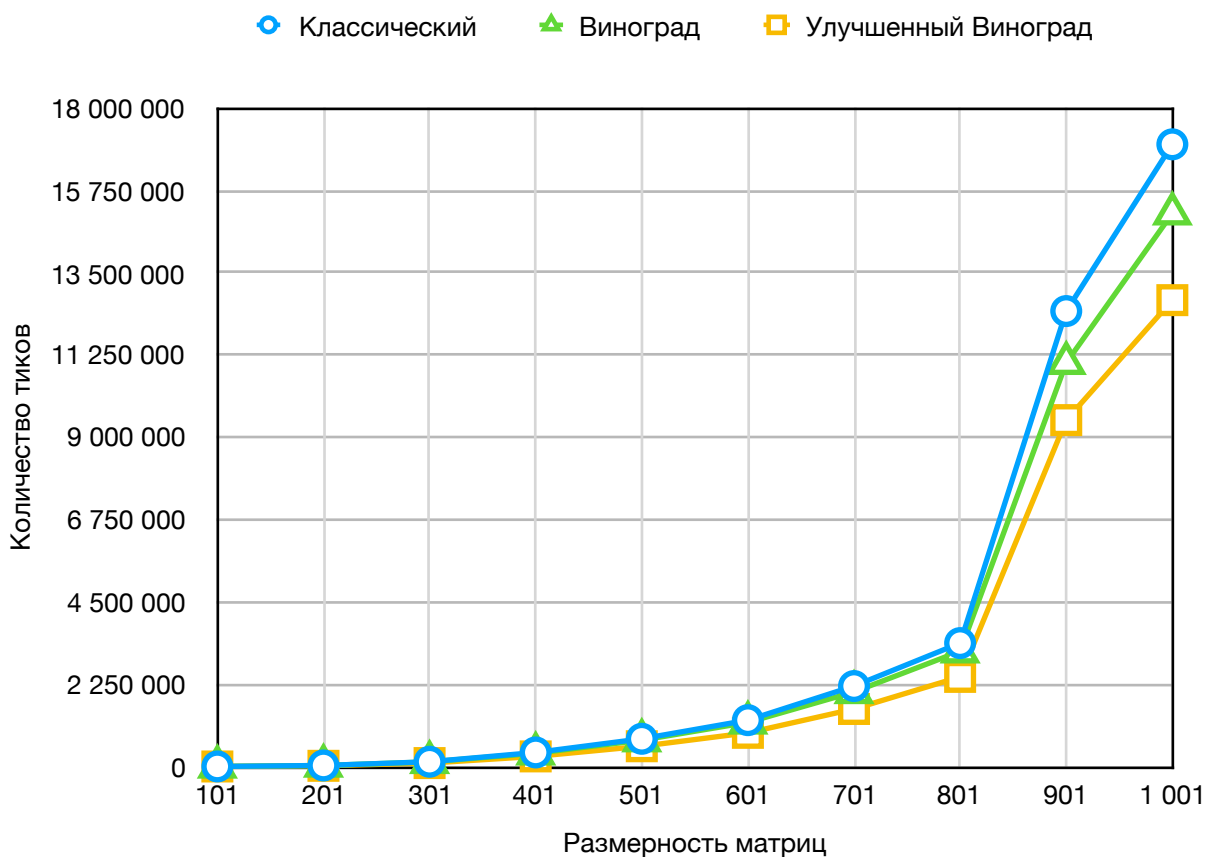

Рисунок 6. График работы алгоритмов по умножению квадратных матриц с нечетной размерностью (ось абсцисс — время работы алгоритмов (тики), ось ординат — размерность матриц).

Сравнение времени умножения квадратных матриц с чётной и нечётной размерностью алгоритмом Винограда:

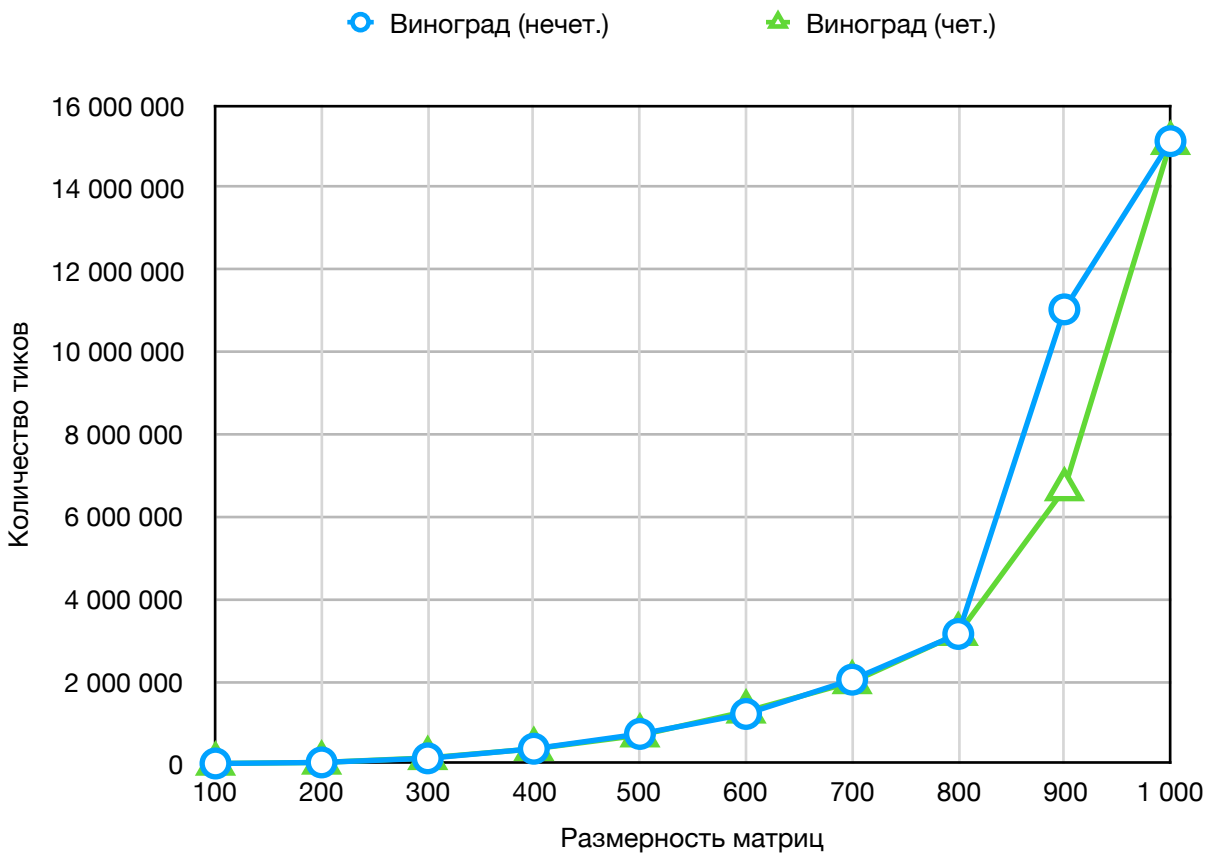

Рисунок 7. График работы алгоритма Винограда по умножению квадратных матриц с нечетной и четной размерностью (ось абсцисс — время работы алгоритмов (тики), ось ординат— размерность матриц).

4.4 Вывод

По результатам тестирования выявлено, что все рассматриваемые алгоритмы реализованы правильно. Самым медленным алгоритмом оказался алгоритм классического умножения матриц, а самым быстрым — улучшенный алгоритм Винограда. Заметим, что оба алгоритма имеют одинаковый характер трудоемкости — кубический от линейного размера матриц.

Заключение

В ходе работы были изучены алгоритмы умножения матриц: классический алгоритм, алгоритм Винограда, улучшенный алгоритм Винограда. Выполнено сравнение всех рассматриваемых алгоритмов. В ходе исследования было установлено, что улучшенный алгоритм Винограда является самым быстрым алгоритмом по трудоемкости и времени выполнения среди рассмотренных алгоритмов умножения матриц. Изучены зависимости времени выполнения алгоритмов от размерности матриц. Также реализован программный код продукта.

Список использованных источников

1. Kakaradov B. Ultra-Fast Matrix Multiplication: An Empirical Analysis of Highly Optimized Vector Algorithms [Электронный ресурс] // cs.stanford.edu: [сайт]. [2004]. URL: https://cs.stanford.edu/people/boyko/pubs/Matrix Mult_SURJ_2004.pdf

2. Stothers A.J. On the Complexity of Matrix [Электронный ресурс] // era.lib.ed.ac.uk: [сайт]. [2010]. URL: https://www.era.lib.ed.ac.uk/bitstream/handle/184 2/4734/Stothers2010.pdf

3. Williams V.V. Multiplying matrices [Электронный ресурс] // http://theory.stanford.edu: [сайт]. [2014]. URL: http://theory.stanford.edu/~virgi/matrixmult-f.pdf

4. Алгоритм Копперсмита — Винограда [Электронный ресурс] // ru.math.wikia.com/: [сайт]. URL: http://ru.math.wikia.com/wiki/Алгоритм_Коппе рсмита_—_Винограда (дата обращения: 4.10.2019).