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"

Алгоритмы умножения матриц

Студент:

Турсунов Жасурбек Рустамович

Группа: ИУ7-56Б

Преподователи:

Волкова Лилия Леонидовна
Строганов Юрий Владимирович

Москва, 2020 г.

Содержание

Введение

Целью данной лабораторной работы является изучение алгоритмов умножения матриц. В данной лабораторной работе рассматривается стандартный алгоритм умножения матриц, алгоритм Винограда и модифицированный алгоритм Винограда. Также требуется изучить рассчет сложности алгоритмов, получить навыки в улучшении алгоритмов.

В ходе лабораторной предстоит выполнить следующие задачи:

1. изучить алгоритмы умножения матриц; стандартный и алгоритм Винограда;
2. оптимизировать алгоритм Винограда;
3. дать теоретическую оценку базового алгоритма умножения матриц, алгоритма Винограда и улучшенного алгоритма Винограда;
4. реализовать три алгоритма умножения матриц на одном из языков программирования;
5. сравнить алгоритмы умножения матриц.

1 Аналитическая часть

Матрицей A размера [m * n] называется прямоугольная таблица чисел, функций или алгебраических выражений, содержащая m строк и n столбцов. Числа m и n определяют размер матрицы. Если число столбцов в первой матрицы совпадает с числом строк во второй матрице, то эти матрицы можно перемножить. У произведения будет столько же строк, сколько в первой матрице и столько же столбцов, сколько во второй.

Пусть даны две прямоугольные матрицы A и B размеров [m * n] и [n * k] соответственно. В результате произведения матриц A и B получим матрицу C размера [m * k].[2]

$c_{i,j} = \sum_{r=1}^{n} a_{i,r} \cdot b_{r,j}$ называется произведением матриц A и B.

1.1 Алгоритм Винограда

Подход Алгоритма Винограда является иллюстрацией общей методологии, начатой в 1979-ч годах на основе билинейных и трилинейных форм, благодоря которым большинство усовершенствований для умножения матриц были получены с помощью разложения скалярного произведения векторов. $v1$

Рассмотрим два вектора $V = (v1,\ v2,\ v3,\ v4)$ и $W = (w1,\ w2,\ w3,\ w4)$.

Их скалярное произведение равно (1.1)

$$V \cdot W = v_1 \cdot w_1 + v_2 \cdot w_2 + v_3 \cdot w_3 + v_4 \cdot w_4 \tag{1.1}$$

Равенство (1.1) можно переписать в виде (1.2)

$$V \cdot W = (v_1 + w_2) \cdot (v_2 + w_1) + (v_3 + w_4) \cdot (v_4 + w_3) - v_1 \cdot v_2 - v_3 \cdot v_4 - w_1 \cdot w_2 - w_3 \cdot w_4 \tag{1.2}$$

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 выполнить лишь первые два умножения и последующие пять сложений, а также дополнительно два сложения.

1.2 Вывод

Были рассмотрены алгоритмы классического умножения матриц и алгоритм Винограда.[1]
Главное отличие которых - наличие предварительной обработки, а также количество операций умножения.

2 Конструкторская часть

Требования к вводу: На вход подаются две матрицы.

1. на вход подаются две строки;

2. строчные и заглавные буквы считаются разными;

Требования к программе: На вход подаются две матрицы.

1. корректное умножение двух матриц;

2. при матрицах разных размеров программа не должнв аварийно завершаться.

2.1 Разработка алгоритмов

В данном разделе будут рассмотрены схемы алгоритмов:

1. классический алгоритм умножения матриц;

2. алгоритм Винограда;

3. оптимизированный алгоритм Винограда.

На рисунке 1 представлена схема классического алгоритма умножения матриц.

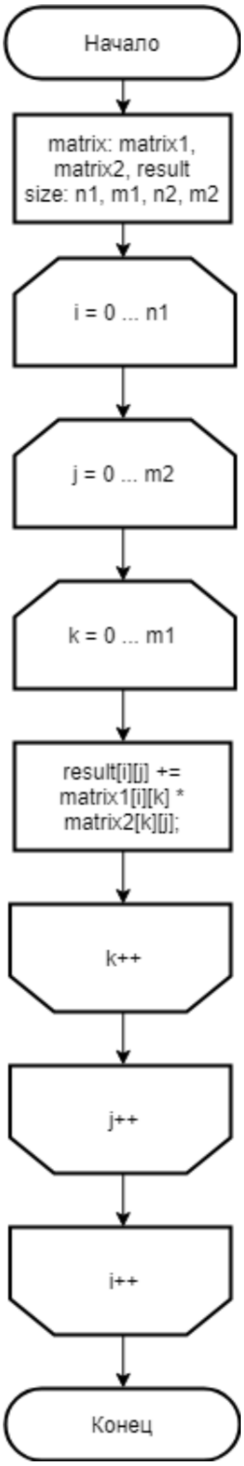

Рис. 1: Схема классического алгоритма умножения матриц

На рисунке 2 представлена схема алгоритма Винограда.

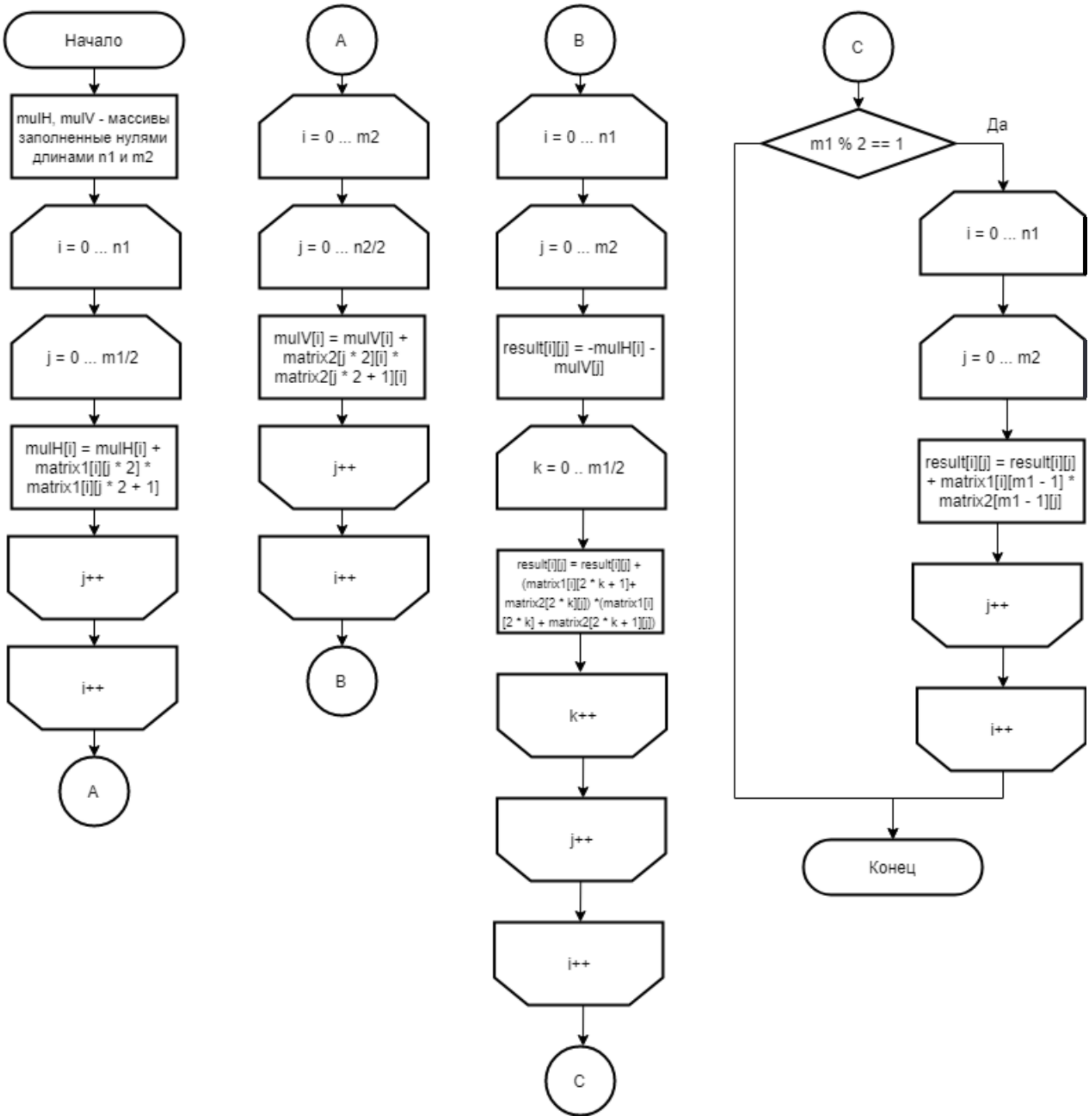

Рис. 2: Схема алгоритма Винограда

На рисунке 3 представлена схема оптимизированного алгоритма Винограда.

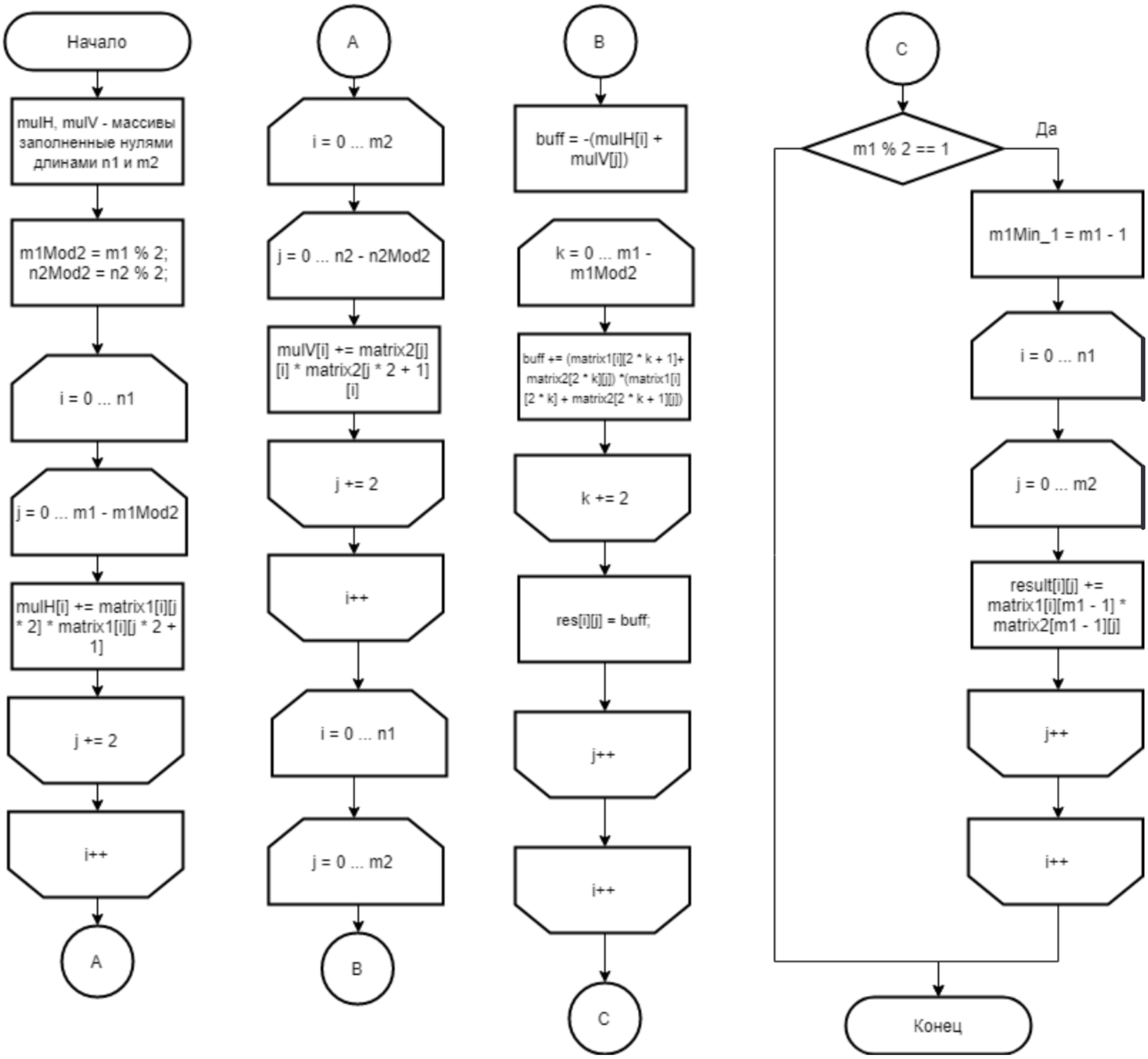

Рис. 3: Схема оптимизированного алгоритма Винограда

2.2 Трудоемкость алгоритмов

Введем модель трудоемкости для оценки алгоритмов:

1. базовые операции стоимостью 1 - +, -, *, /, =, ==, <=, >=, !=, +=, [], получение полей класса;

2. оценка трудоемкости цикла: Fц = init + N * (a + Fтела + post) + a, где a - условие цикла, init - предусловие цикла, post - постусловие цикла;

3. стоимость условного перехода применим за 0, стоимомть вычисления условия остаётся.

Оценим трудоемкость алгоритмов по коду программы.

2.2.1 Классический алгоритм

Рассмотрим трудоемкость первичной проверки на возможность умножения матриц.

Инициализация матрицы результата: $1 + 1 + n_1(1 + 2 + 1) + 1 = 4n_1 + 3$.

Подсчёт:

$1 + n_1(1 + (1 + m_2(1 + (1 + m_1(1 + (8) + 1) + 1) + 1) + 1) + 1) + 1 = n_1(m_2(10m_1 + 4) + 4) + 4) + 2 = 10n_1m_2m_1 + 4n_1m_2 + 4n_1 + 2$.

2.2.2 Алгоритм Винограда

Аналогично рассмотрим трудоемкость алгоритма Винограда.

Первый цикл: $\frac{15}{2}n_1m_1 + 5n_1 + 2$

Второй цикл: $\frac{15}{2}m_2n_2 + 5m_2 + 2$

Третий цикл: $13n_1m_2m_1 + 12n_1m_2 + 4n_1 + 2$

Условный переход: $\begin{bmatrix} 2 & \text{, невыполнение условия} \\ 15n_1m_2 + 4n_1 + 2 & \text{, выполнение условия} \end{bmatrix}$

Итого: $13n_1m_2m_1 + \frac{15}{2}n_1m_1 + \frac{15}{2}m_2n_2 + 12n_1m_2 + 5n_1 + 5m_2 + 4n_1 + 6 + \begin{bmatrix} 2 & \text{, невыполнение условия} \\ 15n_1m_2 + 4n_1 + 2 & \text{, выполнение условия} \end{bmatrix}$

2.2.3 Оптимизированный алгоритм Винограда

Аналогично Рассмотрим трудоемкость оптимизированого алгоритма Винограда:

Первый цикл: $\frac{11}{2}n_1m_1 + 4n_1 + 2$
Второй цикл: $\frac{11}{2}m_2n_2 + 4m_2 + 2$
Третий цикл: $\frac{17}{2}n_1m_2m_1 + 9n_1m_2 + 4n_1 + 2$
Условный переход: $\begin{bmatrix} 1 & \text{, невыполнение условия} \\ 10n_1m_2 + 4n_1 + 2 & \text{, выполнение условия} \end{bmatrix}$

Итого: $\frac{17}{2}n_1m_2m_1 + \frac{11}{2}n_1m_1 + \frac{11}{2}m_2n_2 + 9n_1m_2 + 8n_1 + 4m_2 + 6 + \begin{bmatrix} 1 & \text{, невыполнение условия} \\ 10n_1m_2 + 4n_1 + 2 & \text{, выполнение условия} \end{bmatrix}$

2.3 Вывод

В данном разделе были рассмотрены схемы алгоритмов умножения матриц, введена модель оценки трудоемкости алгоритма, были расчитаны трудоемкости алгоритмов в соответствии с этой моделью.

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: две матрицы.

Выходные данные: матрица полученная в результате умножения двух матриц.

На рисунке 4 представлена IDEF0-диаграмма, описывающая функциональную схему умножения матриц.

Рис. 4: IDEF0-диаграмма, описывающая функциональную схему умножения матриц.

3.2 Средства реализации

В данной работе используется язык программирования Python, так как ЯП позволяет написать программу за кратчайшее время. Проект выполнен в среде разработки Visual Studio Code.

3.3 Листинг кода

В данном пункте представлен листинг кода, а именно:

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

На листинге 1 представлен код классического алгоритма умножения матриц.

```
def classicMul(a, b):
    if len(b) != len(a[0]):
        print("Different dimension of the matrics")
        return
    n = len(a)
    m = len(a[0])
    k = len(b[0])
    res = [[0 for i in range(0, k)] for j in range(0, n)]
    for i in range(n):
        for j in range(m):
            for t in range(k):
                res[i][t] += a[i][j] * b[j][t]
    return res

```

Листинг 1: Классический алгоритм умножения матриц

На листинге 2 представлен код алгоритма Винограда.

```
    def classicVinograd(G, H):
        a = len(G)
        b = len(H)
        c = len(H[0])
        if b != len(G[0]):
            print("Different dimension of the matrics")
            return
        d = b // 2
        row = [0 for i in range(a)]
        col = [0 for i in range(c)]
        for i in range(a):
            for j in range(d):
                row[i] += G[i][2 * j] * G[i][2 * j + 1]
        for i in range(c):
            for j in range(d):
                col[i] += H[2 * j][i] * H[2 * j + 1][i]

        answer = [[0 for i in range(c)] for j in range(a)]
        for i in range(a):
            for j in range(c):
                answer[i][j] = - row[i] - col[j]
                for k in range(d):
                    answer[i][j]+=((G[i][2*k] +
                                H[2*k+1][j])*(G[i][2*k+1]+H[2*k][j]))
        if b % 2:
            for i in range(a):
                for j in range(c):
                    answer[i][j] += G[i][b - 1] * H[b - 1][j]

        return answer

```

Листинг 2: Алгоритм Винограда

На листинге 3 представлен код оптимизированного алгоритма Винограда.

```
def imprvVinograd(G, H):
a = len(G)
b = len(H)
c = len(H[0])

if b != len(G[0]):
    print("Different dimension of the matrics")
    return

d = b // 2

row = [0 for i in range(a)]
col = [0 for i in range(c)]

for i in range(a):
    row[i]=sum(G[i][2*j]*G[i][2*j+1] for j in range(d))

for i in range(c):
    col[i]=sum(H[2*j][i]*H[2*j+1][i] for j in range(d))

answer = [[0 for i in range(c)] for j in range(a)]
for i in range(a):
    for j in range(c):
        answer[i][j]=sum((G[i][2 k]+H[2*k+1][j])*
            (G[i][2*k+1]+H[2*k][j]) for k in range(d))
                                        -row[i]-col[j]

if b % 2:
    for i in range(a):
      answer[i][j]=sum(G[i][b-1]*H[b-1][j] for j in range(c))

return answer

```

Листинг 3: Оптимизированный алгоритм Винограда

3.4 Вывод

В данном разделе была представлена структура ПО и листинги кода программы.

4 Исследовательская часть

В данном разделе будет проведен эксперимент и сравнительный анализ.

4.1 Системные характеристики

Характеристики компьютера на котором проводился замер времени сортировки массива:

1. операционная система - Windows 10;
2. процессор - Intel(R) Core(TM) i7-10510U CPU @1.80GHz 2.30GHz;
3. оперативная память - 16 ГБ.

4.2 Постановка эксперимента

В рамках данного проекта были проведены эксперименты, описанные ниже:

1. Сравнение времени работы алгоритмов при четном и нечеином размере матрицы.

4.3 Сравнительный анализ на основе замеров времени работы алгоритмов

Был проведен замер времени работы каждого из алгоритмов.

На рисунке 5 показаны результаты первого эксперимента, который производится для лучшего случая на квадратных матрицах размером от 100 х 100 до 1000 х 1000 с шагом 100. Ниже приведена полученная диаграмма:

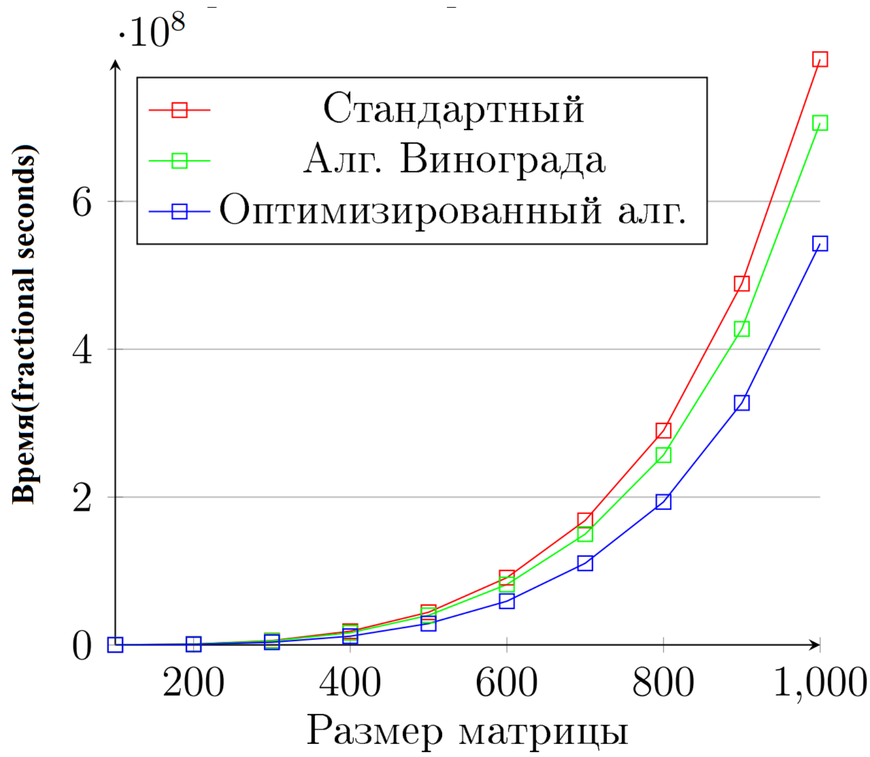

Рис. 5: Сравнение времени работы алгоритмов при четном размере матрицы

На рисунке 6 показаны результаты второго эксперимента, который производится для худшего случая, когда поданы квадратные матрицы с нечетными размерами от 101 x 101 до 1001 x 1001 с шагом 100. Ниже приведена полученная диаграмма:

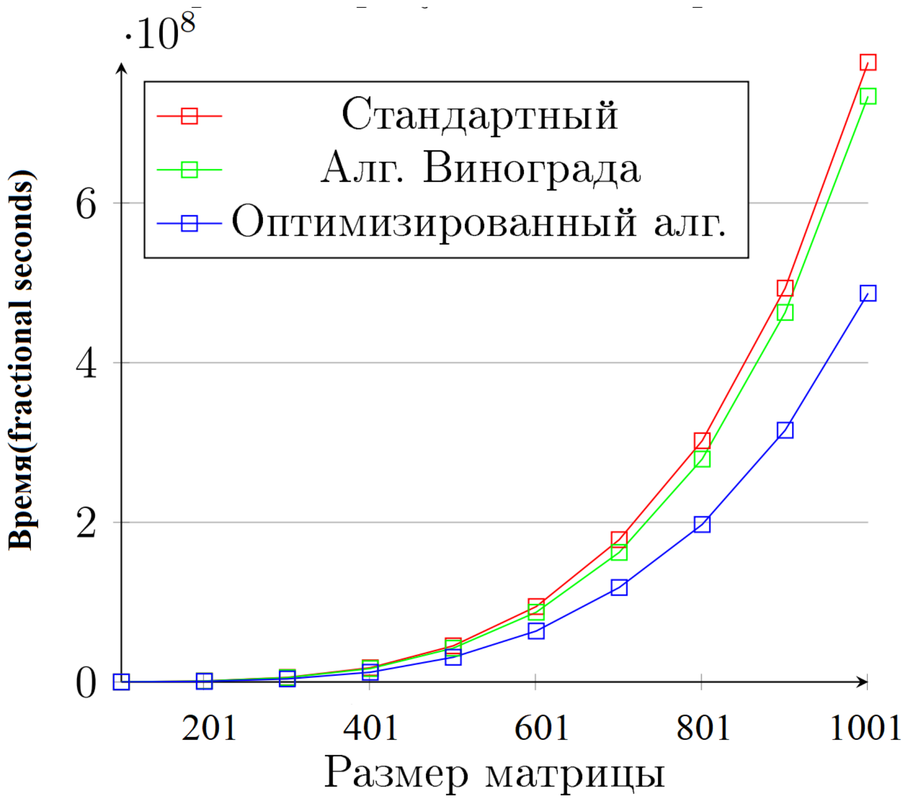

Рис. 6: Сравнение времени работы алгоритмов при нечетном размере матрицы

4.4 Тестирование программы

В данном разделе будут показаны результаты тестирования

Всего было реализовано 7 тестовых случаев:

1. Некорректный размер матриц;

2. Размер матрицы равен 1;

3. Размер матрицы равен 2;

4. Сравнение работы стандартной реализации с алгоритмом Винограда на случайных значениях при четном и нечетном размерах матриц;

5. Сравнение работы стандартной реализации с оптимизированным алгоритмом Винограда на случайных значениях при четном и нечетном размерах матриц;

На рисунке 7 показаны результаты тестирования

```
Number of failed tests:
     Not correct size of matrix: 0
     Size of matrix is 1: 0
     Size of matrix is 2: 0
     Standart Vinograd(even) : 0
     Standart Vinograd(odd) : 0
     Optimized Vinograd(even) : 0
     Optimized Vinograd(odd) : 0
```

Рис. 7: Результаты тестирования

4.5 Вывод

По результатам тестирования все рассматриваемые алгоритмы были реализованы верно. Самым медленным алгоритмом оказался алгоритм классического умножения матриц, а самым быстрым - оптимизированный алгоритм Винограда.

Заключение

В ходе работы были изучены алгоритмы классического умножения матриц, стандартный и оптимизированный алгоритм Винограда. Также была дана теоретическая оценка алгоритмам умножения матриц. Было сделано сравнение этих алгоритмов, также был реализован программный код продукта.

Список литературы

[1] algolib@narod.ru. *Основные алгоритмы умножения матриц[ЭЛ. РЕСУРС] Режим доступа:* URL: http://www.algolib.narod.ru/Math/Matrix.html. (дата обращения: 18.10.2020).

[2] Ермолаев Игорь. *Умножение матриц: эффективная реализация шаг за шагом [ЭЛ. РЕСУРС] Режим доступа:* URL: https://habr.com/ru/post/359272/. (дата обращения: 20.10.2020).