Министерство науки и высшего образования 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"

Тема Алгоритм Копперсмита-Винограда

Студент Козлова И.В.

Группа ИУ7-52Б

Оценка (баллы) ______

Преподаватель Волкова Л.Л.

Москва — 2021 г.

Оглавление

Введение

Термин «матрица» применяется во множестве разных областей: от программирования до кинематографии.

Матрица в математике – это таблица чисел, состоящая из определенного количества строк (m) и столбцов (n).

Мы встречаемся с матрицами каждый день, так как любая числовая информация, занесенная в таблицу, уже в какой-то степени считается матрицей.

Целью работы работы является изучение и реализация алгоритмов умножения матриц, вычисление трудоёмкости этих алгоритмов. В данной лабораторной работе рассматривается стандартный алгоритм умножения матриц, алгоритм Винограда и модифицированный алгоритм Винограда.

Для достижения цели ставятся следующие задачи.

1. Изучить классический алгоритм умножения матриц, алгоритм Винограда и модифицированный алгоритм Винограда.
2. Реализовать классический алгоритм умножения матриц, алгоритм Винограда и модифицированный алгоритм Винограда.
3. Дать оценку трудоёмкости алгоритмов.
4. Замерить время работы алгоритмов.
5. Описать и обосновать полученные результаты в отчете о выполненной лабораторной работе, выполненном как расчётно-пояснительная записка к работе.

1 Аналитическая часть

В этом разделе будут представлены описания алгоритмов стандартного умножения матриц и алгоритм Копперсмита-Винограда.

1.1 Матрица

Матрица [1] – математический объект, эквивалентный двумерному массиву. Числа располагаются в матрице по строкам и столбцам. Две матрицы одинакового размера можно поэлементно сложить или вычесть друг из друга.

Если число столбцов в первой матрице совпадает с числом строк во второй, то эти две матрицы можно перемножить. У произведения будет столько же строк, сколько в первой матрице, и столько же столбцов, сколько во второй. Например, при умножении матрицы размером 3x4 на матрицу размером 4x7 мы получаем матрицу размером 3x7.

Умножение матриц некоммутативно: оба произведения AB и BA двух квадратных матриц одинакового размера можно вычислить, однако результаты, вообще говоря, будут отличаться друг от друга.

1.2 Стандартный алгоритм

Пусть даны две прямоугольные матрицы

$$A_{lm} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{l1} & a_{l2} & \dots & a_{lm} \end{pmatrix}, \quad B_{mn} = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{pmatrix}, \tag{1.1}$$

тогда матрица C

$$C_{ln} = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{l1} & c_{l2} & \dots & c_{ln} \end{pmatrix}, \tag{1.2}$$

где

$$c_{ij} = \sum_{r=1}^{m} a_{ir} b_{rj} \quad (i = \overline{1,l}; j = \overline{1,n}) \tag{1.3}$$

будет называться произведением матриц A и B.

Стандартный алгоритм реализует данную формулу.

1.3 Алгоритм Копперсмита-Винограда

Алгоритм Копперсмита-Винограда [2] — алгоритм умножения квадратных матриц, предложенный в 1987 году Д. Копперсмитом и Ш. Виноградом. В исходной версии асимптотическая сложность алгоритма составляла $O(n^{2,3755})$, где n — размер стороны матрицы. Алгоритм Копперсмита — Винограда, с учетом серии улучшений и доработок в последующие годы, обладает лучшей асимптотикой среди известных алгоритмов умножения матриц [3].

Рассмотрим два вектора $V = (v_1, v_2, v_3, v_4)$ и $W = (w_1, w_2, w_3, w_4)$. Их скалярное произведение равно: $V \cdot W = v_1w_1 + v_2w_2 + v_3w_3 + v_4w_4$, что эквивалентно (1.4):

$$V \cdot W = (v_1 + w_2)(v_2 + w_1) + (v_3 + w_4)(v_4 + w_3) - v_1v_2 - v_3v_4 - w_1w_2 - w_3w_4. \tag{1.4}$$

Несмотря на то, что второе выражение требует вычисления большего количества операций, чем стандартный алгоритм: вместо четырех умножений - шесть, а вместо трех сложений - десять, 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, что позволит для каждого элемента выполнять лишь два умножения и пять сложений, складывая затем только лишь с 2 предварительно посчитанными суммами соседних элементов текущих строк и столбцов. Из-за того, что операция сложения быстрее операции умножения в ЭВМ, на практике алгоритм должен работать быстрее стандартного.

1.4 Оптимизированный алгоритм Винограда

Оптимизированный алгоритм Винограда представляет собой обычный алгоритм Винограда, за исключением следующих оптимизаций:

- вычисление происходит заранее;
- используется битовый сдвиг, вместо деления на 2;
- последний цикл для нечётных элементов включён в основной цикл, используя дополнительные операции в случае нечётности N.

Вывод

Были рассмотрены алгоритмы классического умножения матриц и алгоритм Винограда, основное отличие которого от классического алгоритма — наличие предварительной обработки, а также количество операций умножения.

2 Конструкторская часть

В этом разделе будут приведены требования к вводу и программе, а также схемы алгоритмов умножения матриц.

2.1 Разработка алгоритмов

На рисунке 2.1 приведена схема стандартного алгоритма умножения матриц.

На рисунках 2.2 и 2.3 приведена схема алгоритма Винограда умножения матриц.

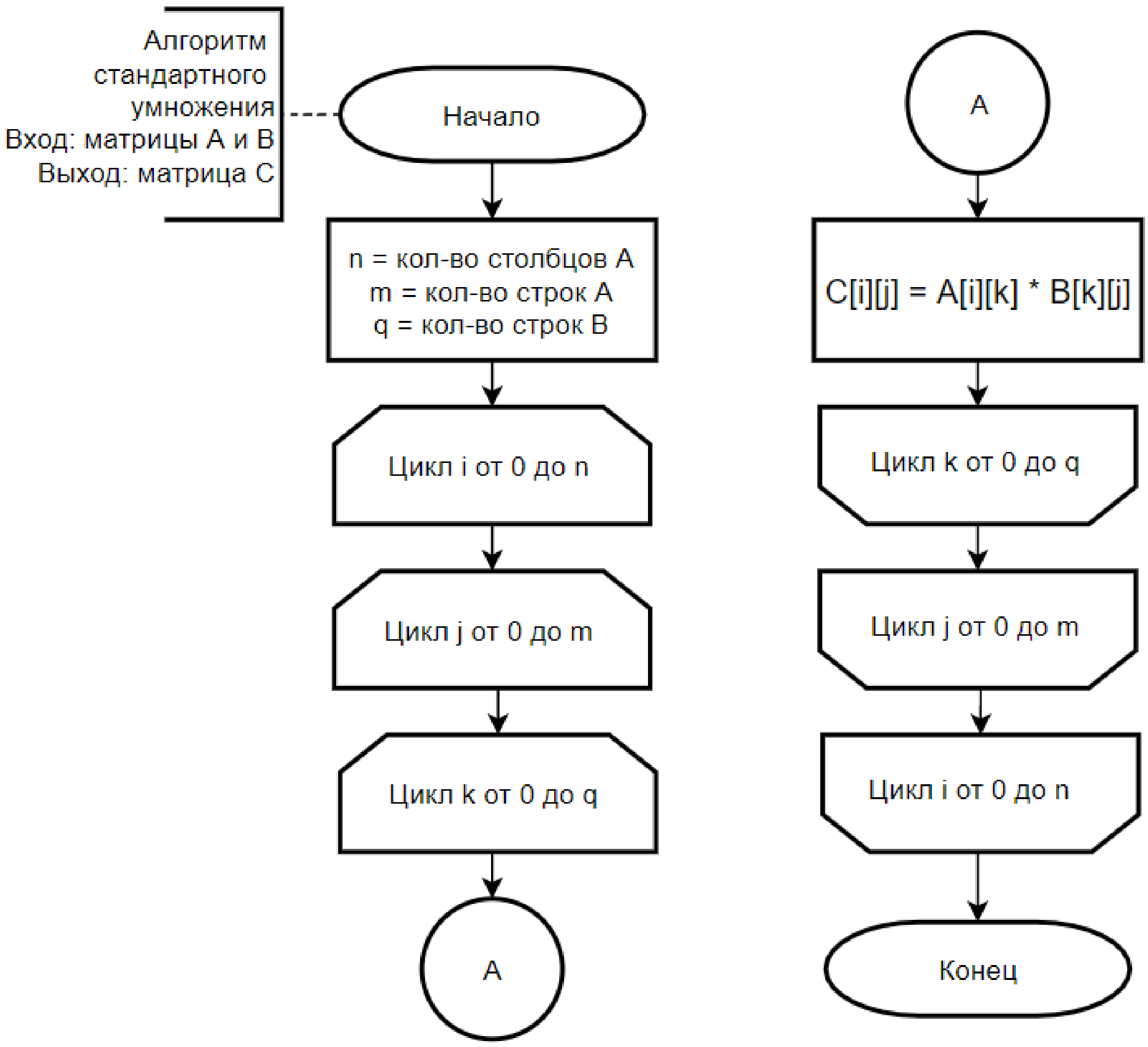

Рисунок 2.1 – Схема стандартного алгоритма умножения матриц

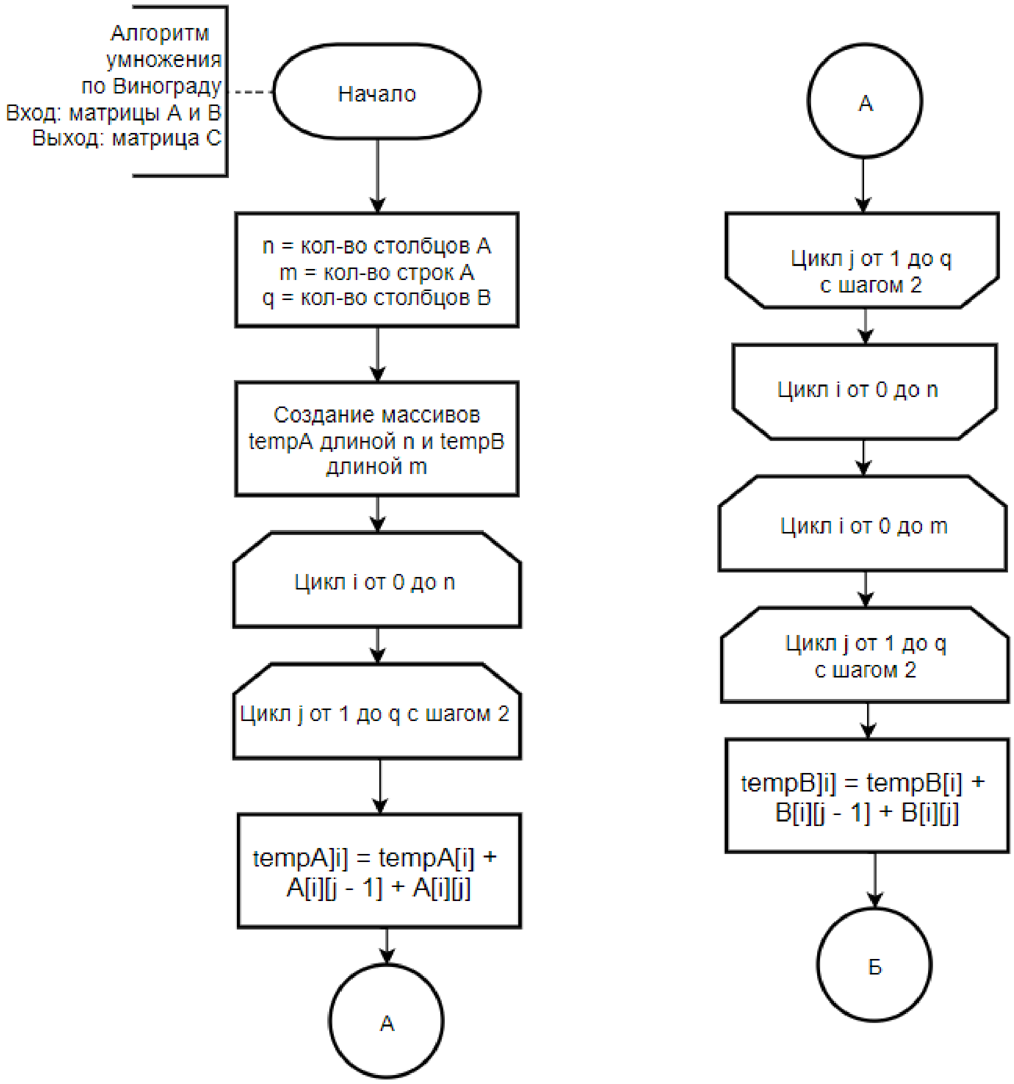

Рисунок 2.2 – Схема алгоритма Винограда умножения матриц

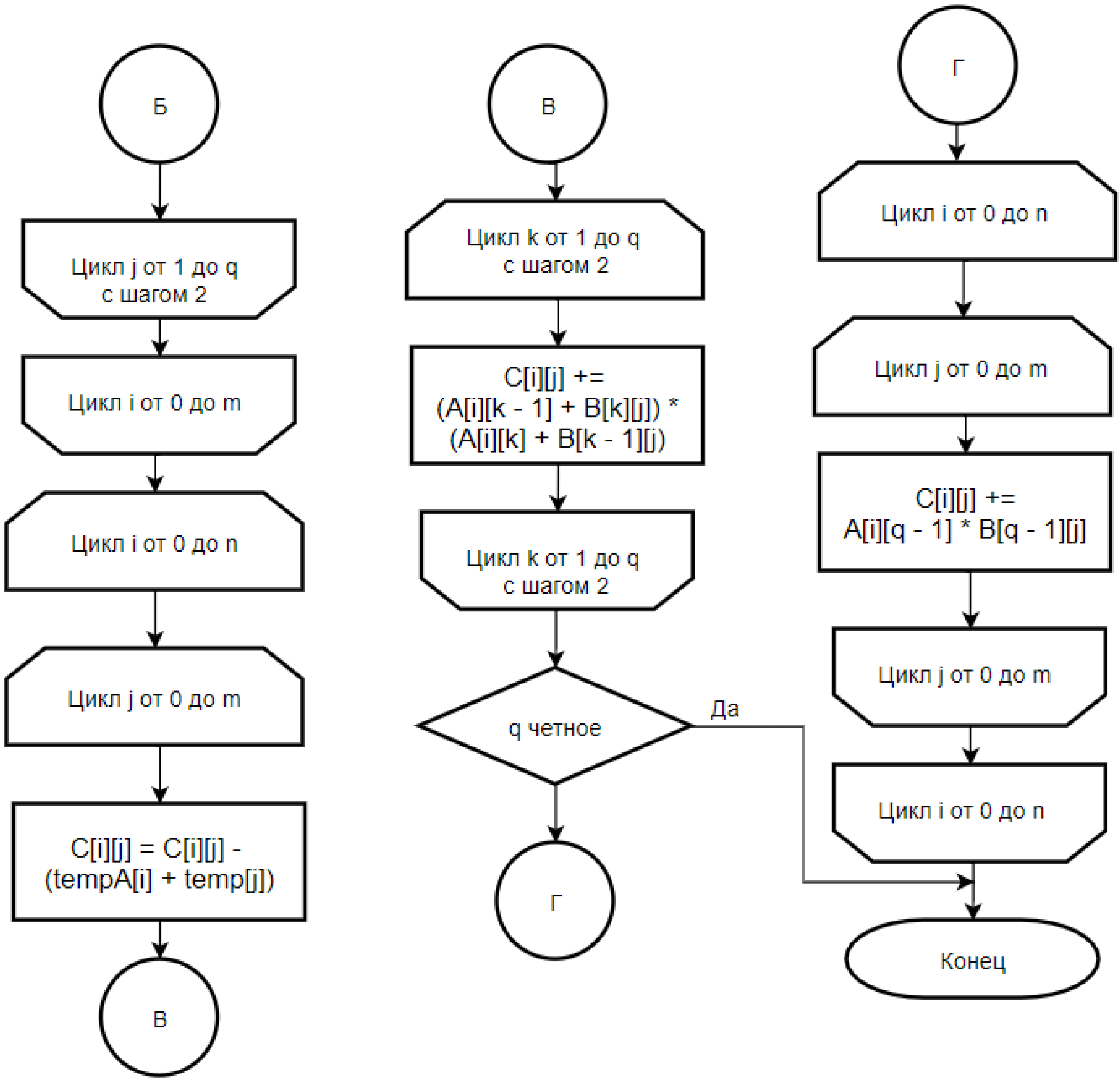

Рисунок 2.3 – Схема алгоритма Винограда умножения матриц(продолжение)

2.2 Модель вычислений

2.3 Трудоемкость алгоритмов

2.3.1 Стандартный алгоритм умножения матриц

2.3.2 Алгоритм Копперсмита — Винограда

2.3.3 Оптимизированный алгоритм Копперсмита — Винограда

3 Технологическая часть

В данном разделе будут приведены требования к программному обеспечению, средства реализации и листинги кода.

3.1 Требования к ПО

К программе предъявляется ряд требований.

- Входными данными являются две матрицы A и B. Количество столбцов матрицы A долджно быть равно количеству строк матрицы B.
- На выходе получается результат умножения, введенных пользователем, матриц.

3.2 Средства реализации

В качестве языка программирования для реализации данной лабораторной работы был выбран ЯП Python [4].

Данный язык достаточно удобен и гибок в использовании.

Время работы алгоритмов было замерено с помощью функции process_time() из библиотеки time [5]

3.3 Сведения о модулях программы

Программа состоит из двух модулей:

1. main.py - главный файл программы, в котором располагаются коды всех алгоритмов и меню;
2. test.py - файл с замерами времени для графического изображения результата;

3.4 Листинг кода

В листинге 3.1 приведена реализация алгоритма стандартного алгоритма матриц.

В листингах 3.2, 3.3 приведена реализация алгоритма Копперсмита-Винограда.

Листинг 3.1 – Реализация стандарного умножения матриц

```
def simple_matrix_mult(matrix1, matrix2):
    if (len(matrix1) != len(matrix2[0])):
        print("Error size matrix!")
        return -1

    n = len(matrix1)
    m = len(matrix1[0])
    q = len(matrix2[0])

    matrix_res = [[0] * q for i in range(n)]

    for i in range(n):
        for j in range(q):
            for k in range(m):
                matrix_res[i][j] = matrix_res[i][j] + \
                    matrix1[i][k] * matrix2[k][j]

    return matrix_res
```

Листинг 3.2 – Реализация алгоритма Копперсмита-Винограда

```
def winograd_matrix_mult(matrix1, matrix2):
    if (len(matrix2) != len(matrix1[0])):
        print("Error size matrix!")
        return -1

    n = len(matrix1)
    m = len(matrix1[0])
    q = len(matrix2[0])

    matrix_res = [[0] * q for i in range(n)]

    row_factor = [0] * n
```

```
    for i in range(n):
        for j in range(0, m // 2, 1):
            row_factor[i] = row_factor[i] + \
                matrix1[i][2 * j] * matrix1[i][2 * j + 1]

    column_factor = [0] * q
    for i in range(q):
        for j in range(0, m // 2, 1):
            column_factor[i] = column_factor[i] + \
                matrix2[2 * j][i] * matrix2[2 * j + 1][i]

    for i in range(n):
        for j in range(q):
            matrix_res[i][j] = -row_factor[i] - column_factor[j]
            for k in range(0, m // 2, 1):
                matrix_res[i][j] = matrix_res[i][j] +\
                    (matrix1[i][2 * k + 1] + matrix2[2 * k][j]) * \
                    (matrix1[i][2 * k] + matrix2[2 * k + 1][j])

    if m % 2 == 1:
        for i in range(n):
            for j in range(q):
                matrix_res[i][j] = matrix_res[i][j] +\
                    matrix1[i][m - 1] * matrix2[m - 1][j]

    return matrix_res
```

Листинг 3.3 – Реализация алгоритма Копперсмита-Винограда (оптимизированный)

```
def winograd_matrix_mult_opim(matrix1, matrix2):
    if (len(matrix2) != len(matrix1[0])):
        print("Error size matrix!")
        return -1

    n = len(matrix1)
    m = len(matrix1[0])
    q = len(matrix2[0])

    matrix_res = [[0] * q for i in range(n)]

```

```
    row_factor = [0] * n
    for i in range(n):
        for j in range(1, m, 2):
            row_factor[i] += matrix1[i][j] * matrix1[i][j - 1]

    column_factor = [0] * q
    for i in range(q):
        for j in range(1, m, 2):
            column_factor[i] += matrix2[j][i] * matrix2[j - 1][i]

    flag = n % 2
    for i in range(n):
        for j in range(q):
            matrix_res[i][j] = -(row_factor[i] + column_factor[j])
            for k in range(1, m, 2):
                matrix_res[i][j] += (matrix1[i][k - 1] + matrix2[k][j]) * \
                        (matrix1[i][k] + matrix2[k - 1][j])
            if (flag):
                matrix_res[i][j] += matrix1[i][m - 1] \
                    * matrix2[m - 1][j]

    return matrix_res
```

3.5 Функциональные тесты

В таблице 3.1 приведены тесты для функций, реализующих стандартный алгоритм умножения матриц, алгоритм Винограда и оптимизированный алгоритм Винограда. Тесты пройдены успешно.

Вывод

В этом разделе была представлена реализация алгоритмов классического умножения матриц, алгоритма Винограда, оптимизирвоанного алгоритма Винограда. Тестирование показало, что алгоритмы реализованы правильно и работают корректно.

Матрица 1	Матрица 2	Ожидаемый результат
$\begin{pmatrix}1 & 1 & 1\\ 5 & 5 & 5\\ 2 & 2 & 2\end{pmatrix}$	$\begin{pmatrix}1\\ 1\\ 1\end{pmatrix}$	$\begin{pmatrix}3\\ 15\\ 6\end{pmatrix}$
$\begin{pmatrix}1 & 1 & 1\end{pmatrix}$	$\begin{pmatrix}1\\ 1\\ 1\end{pmatrix}$	$\begin{pmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{pmatrix}$
$\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix}$	$\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix}$	$\begin{pmatrix}2 & 2\\ 2 & 2\end{pmatrix}$
$\begin{pmatrix}2\end{pmatrix}$	$\begin{pmatrix}2\end{pmatrix}$	$\begin{pmatrix}4\end{pmatrix}$
$\begin{pmatrix}1 & -2 & 3\\ 1 & 2 & 3\\ 1 & 2 & 3\end{pmatrix}$	$\begin{pmatrix}-1 & 2 & 3\\ 1 & 2 & 3\\ 1 & 2 & 3\end{pmatrix}$	$\begin{pmatrix}0 & 4 & 6\\ 4 & 12 & 18\\ 4 & 12 & 18\end{pmatrix}$
$\begin{pmatrix}1 & 2\end{pmatrix}$	$\begin{pmatrix}1 & 2\end{pmatrix}$	Не верный размер

Таблица 3.1 – Тестирование функций

4 Исследовательская часть

В данном разделе будут приведены примеры работы программ, постановка эксперимента и сравнительный анализ алгоритмов на основе полученных данных.

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование:

- Операционная система: Ubuntu 20.04.3 [6] Linux [7] x86_64.
- Память: 8 GiB.
- Процессор: 11th Gen Intel® Core™ i5-1135G7 @ 2.40GHz [8].

Тестирование проводилось на ноутбуке, включенном в сеть электропитания. Во время тестирования ноутбук был нагружен только встроенными приложениями окружения, а также непосредственно системой тестирования.

4.2 Демонстрация работы программы

На рисунке представлен результат работы программы.

4.3 Время выполнения алгоритмов

Таблица 4.1 – Результаты замеров времени.

Вывод

В результате проведенного эксперимента был получен следующий вывод: оптимизированный алгоритм Винограда работает быстрее классического метода и зачительно быстрее обычного алгоритма Винограда.

Заключение

В ходе выполнения лабораторной работы были решены следующие задачи:

- были изучены и реализованы 3 алгоритма перемножения матриц: обычный, Копперсмита-Винограда, модифицированный Копперсмита-Винограда;
- был произведен анализ трудоёмкости алгоритмов на основе теоретических расчетов и выбранной модели вычислений;
- был сделан сравнительный анализ алгоритмов на основе экспериментальных данных;
- подготовлен отчет о лабораторной работе.

Литература

[1] Матрица – Википедия [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/\T2A\CYRM\T2A\cyra\T2A\cyrt\T2A\cyrr\T2A\cyri\T2A\cyrc\T2A\cyra_(\T2A\cyrm\T2A\cyra\T2A\cyrt\T2A\cyre\T2A\cyrm\T2A\cyra\T2A\cyrt\T2A\cyri\T2A\cyrk\T2A\cyra) (дата обращения: 13.09.2021).

[2] Умножение матриц [Электронный ресурс]. Режим доступа: http://algolib.narod.ru/Math/Matrix.html (дата обращения: 13.09.2021).

[3] Group-theoretic Algorithms for Matrix Multiplication / H. Cohn, R. Kleinberg, B. Szegedy et al. // Proceedings of the 46th Annual Symposium on Foundations of Computer Science. 2005. October. P. 379–388.

[4] Welcome to Python [Электронный ресурс]. Режим доступа: https://www.python.org (дата обращения: 04.09.2021).

[5] time — Time access and conversions [Электронный ресурс]. Режим доступа: https://docs.python.org/3/library/time.html#functions (дата обращения: 04.09.2021).

[6] Ubuntu 20.04.3 LTS (Focal Fossa) [Электронный ресурс]. Режим доступа: https://releases.ubuntu.com/20.04/ (дата обращения: 04.09.2021).

[7] Linux – Википедия [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Linux (дата обращения: 04.09.2021).

[8] Процессор Intel® Core™ i5-1135G7 [Электронный ресурс]. Режим доступа: https://www.intel.ru/content/www/ru/ru/products/sku/208658/intel-core-i51135g7-processor-8m-cache-up-to-4-20-ghz/specifications.html (дата обращения: 04.09.2021).