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»

Лабораторная работа №2
по дисциплине «Анализ алгоритмов»

Тема Поиск в массиве

Студент Звягин Д.О.

Группа ИУ7-53Б

Преподаватель Волкова Л.Л., Строганов Д.В.

Москва, 2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Алгоритм Винограда — это алгоритм умножения матриц, являющийся альтернативой стандартному. В данной лабораторной работе будет рассмотрен этот алгоритм, а также проведено его сравнение со стандартным алгоритмом.

Целью этой лабораторной работы является исследование умножение матриц этими алгоритмами, а также сравнение их временн*ы*х характеристик

Для достижения поставленной цели необходимо выполнить следующие задачи:

— рассмотреть существующие алгоритмы умножения матриц;

— разработать рассмотренные алгоритмы умножения матриц;

— реализовать разработанные алгоритмы;

— проанализировать временные характеристики полученных реализаций.

1 Аналитическая часть

В данном разделе будут рассмотрены следующие алгоритмы умножения матриц:
— классический алгоритм;
— алгоритм Винограда.

1.1 Формализация понятия матрицы

Матрица — это таблица чисел, записанная в форме (1.1) [5]

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \dots & \dots & \ddots & \dots \\ a_{M1} & a_{M2} & \dots & a_{MN} \end{pmatrix} \tag{1.1}$$

Числа M и N — называются размерностями матрицы и определяют количество столбцов и строк соответственно.

Путь A — матрица. Тогда Обозначение A_{ij} или a_{ij} означает элемент матрицы A, расположенный на i-й строке и j-м столбце.

1.2 Определение умножения матриц

Для данной лабораторной работы также необходимо определить операцию умножения матриц.

Пусть A, B и C - матрицы.

Выполнить операцию умножения матриц можно только если число столбцов первой матрицы равно количеству строк второй матрицы. Пусть матрицы A и B удовлетворяют этому условию.

Пусть матрица C задаётся отношением (1.2)

$$C = \left(c_{i,j}\right), \quad c_{ij} = \sum_{k=1}^{N} B_{ik} \times Akj, \quad i,j = \overline{1..N} \tag{1.2}$$

Где N - количество столбцов первой матрицы.

Тогда уравнение (1.3) есть определение произведения матриц A и B [5]

$$C = AB \tag{1.3}$$

1.3 Классический алгоритм умножения матриц

Классический алгоритм умножения матриц является вычислением каждой ячейки выходной матрицы по формуле (1.2)

1.3.1 Преимущества

Преимуществом данного алгоритма является относительная простота написания - для его работы не требуется дополнительной обработки матриц и подготовки каких-либо данных.

1.3.2 Недостатки

Главным недостатком этого алгоритма является скорость его выполнения.

1.4 Алгоритм Копперсмита-Винограда

Алгоритм Копперсмита-Винограда — алгоритм, основанный на идее подготовки части данных для более эффективного умножения матриц [6]. В первом издании (1990 года) авторам удалось достичь асимптотической сложности $O(n^{2.3755})$, где n - размер сторон матрицы.

Идея алгоритма основывается на следующем наблюдении: каждый элемент выходной матрицы является скалярным произведением строки первой и столбца второй матрицы. Скалярное произведение можно преобразовать и использовать преобразованную форму для вычисления произведения матрицы.

Тогда аналогом формулы (1.2) для алгоритма Копперсмита-Винограда будет формула (1.4)

$$c_{ij} = \sum_{k=1}^{q/2} (a_{i,2k-1} + b_{2k,j})(a_{i,2k} + b_{2k-1,j}) - \sum_{k=1}^{q/2} (a_{i,2k-1} \times a_{i,2k}) + (b_{2k-1,j} \times b_{2k,j}) \tag{1.4}$$

Операнды со знаком из уравнения (1.4) могут быть вычислены заранее для каждого столбца и строки матрицы и переиспользованы в процессе вычисления вычисления выходной матрицы, что позволяет значительно увеличить скорость расчёта.

1.4.1 Преимущества

Главным преимуществом данного алгоритма является скорость выполнения. Особенно заметным прирост становится при матрицах большого размера [6].

1.4.2 Недостатки

Недостатками этого алгоритма являются:

— необходимость в предварительной обработке матриц;

— необходимость в выделении дополнительной памяти для хранения рассчитанных значений;

— дополнительная обработка тех случаев, когда количество строк и столбцов в матрице является нечётным.

1.5 Вывод

Были рассмотрены классический алгоритм умножения матриц, и алгоритм Копперсмита-Винограда.

2 Конструкторская часть

2.1 Требования к программному обеспечению

К разрабатываемой программе предъявлен ряд требований:

— на вход подаются две матрицы A и B;

— на выход подаётся матрица C равная произведению матрицы A на матрицу B;

— под матрицей в программе понимается двумерный массив чисел;

— индексация в массивах начинается с 0;

— в случае если произведение матрицы A на матрицу B не определено (количество столбцов первой матрицы и рядов второй не совпадает), должно быть выведено сообщение об ошибке;

2.2 Разработка алгоритмов

В алгоритмах подразумевается, что матрица C уже инициализирована и требуется только её заполнение.

Классический алгоритм умножения матриц изображён на рисунке 2.1

Алгоритм умножения матриц Копперсмита-Винограда изображён на рисунках 2.2 и 2.3

2.3 Вывод

В результате конструкторской части были определены требования к ПО, а также разработаны алгоритмы классического умножения матриц и алгоритм Копперсмита-Винограда.

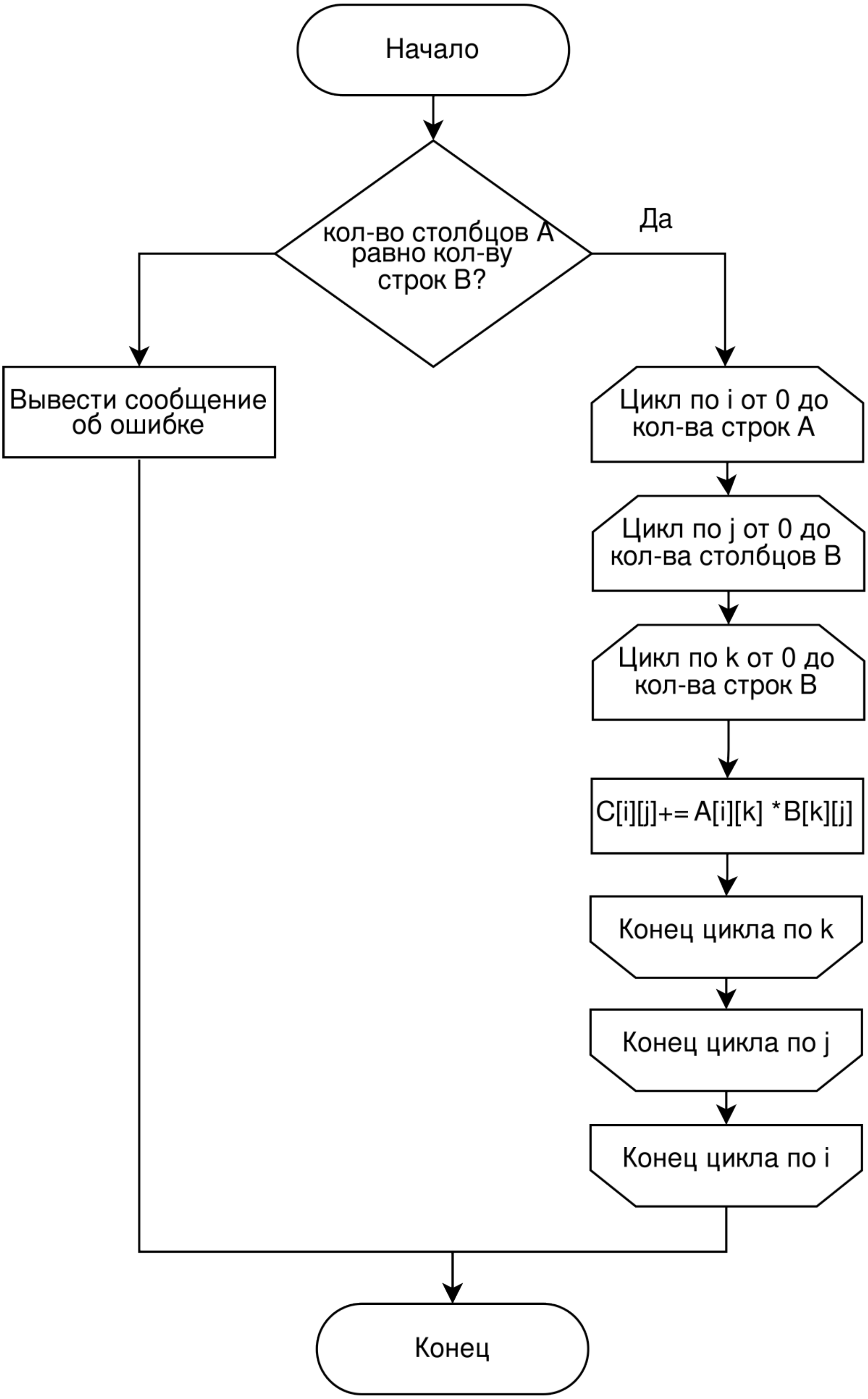

Рисунок 2.1 — Классический алгоритм умножения матриц

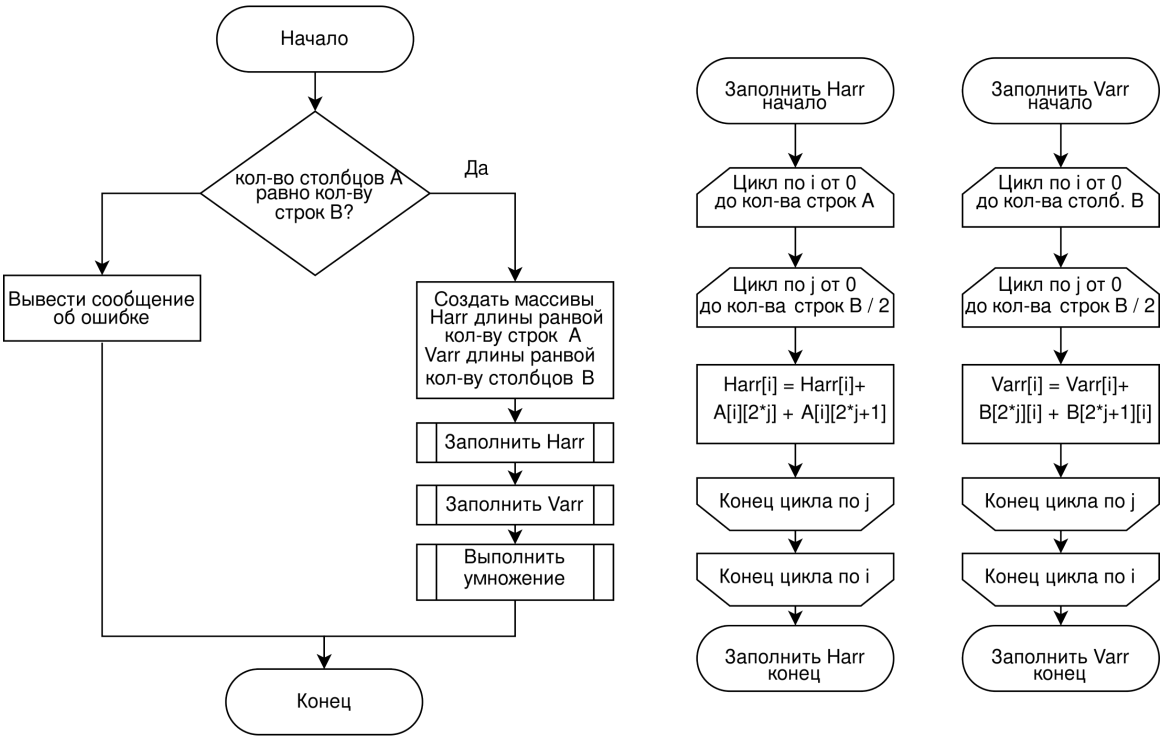

Рисунок 2.2 — Алгоритм умножения матриц Копперсмита-Винограда

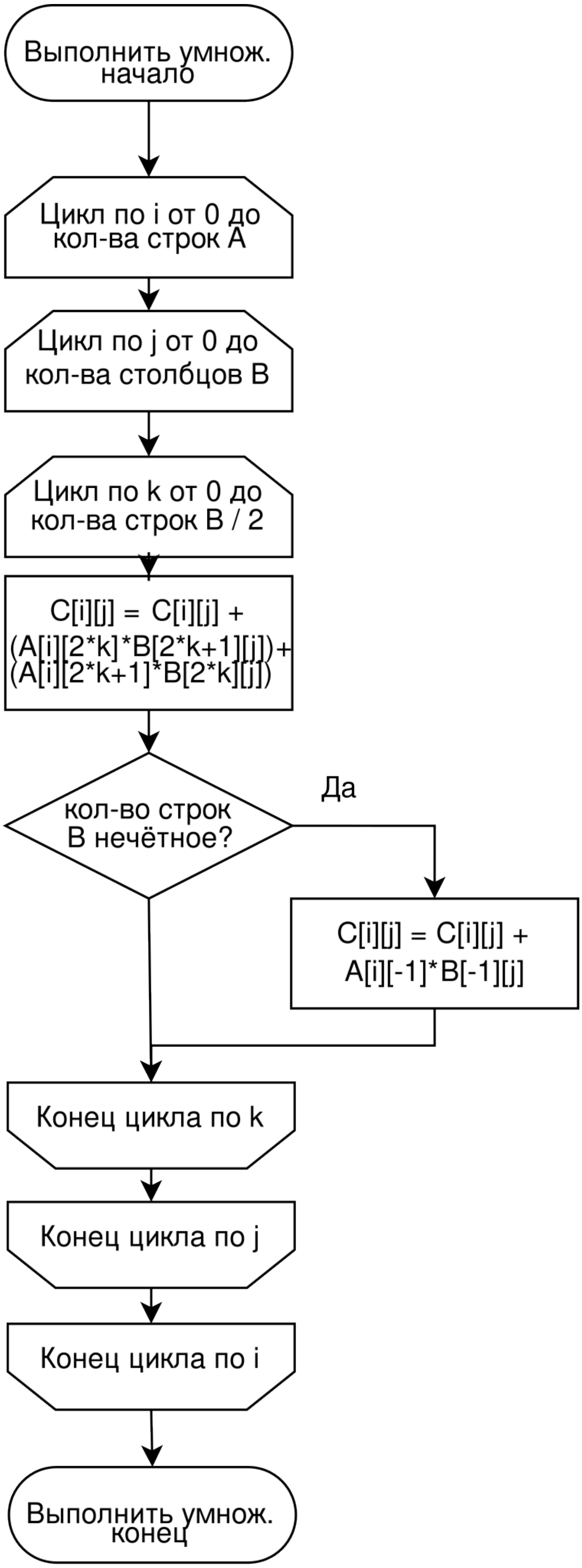

Рисунок 2.3 — Вторая часть алгоритма умножения матриц Копперсмита-Винограда

3 Технологическая часть

3.1 Средства разработки

В качестве языка программирования был выбран python3 [1], так как его стандартная библиотека достаточна для реализации данных алгоритмов, а также данный язык обладает множеством инструментов для визуализации и работы с данными и таблицами.

В качестве основного файла был выбран инструмент jupyter notebook [2], так как он позволяет организовать код в виде блоков, а также выводить данные и графики прямо в нём, что позволяет наглядно продемонстрировать все замеры.

Для построения графиков использовалась библиотека matplotlib [3].

Для замеров процессорного времени использовалась библиотека time [4]

3.2 Реализация алгоритмов

Листинг 3.1 — Классический алгоритм умножения матриц

```
def multiply(A: list[list[float]], B: list[list[float]]) -> list[list[
    float]]:
    if len(A[0]) != len(B):
        raise ValueError("Size mismatch")

    res: list[list[float]] = [
        [
            0. for _ in range(len(B[0]))
        ] for _ in range(len(A))
    ]

    for i in range(len(A)):
        for j in range(len(B[0])):
            for k in range(len(B)):
                res[i][j] += A[i][k] * B[k][j]

    return res
```

При реализации алгоритма Копперсмита-Винограда были применены следующие оптимизации:

— инкремент счётчика наиболее вложенного цикла на 2;

— объединение III и IV частей алгоритма Винограда;

— вынос начальной итерации из каждого внешнего цикла.

Листинг 3.2 — Алгоритм Копперсмита-Винограда умножения матриц

```
def multiply_optimized(A: list[list[float]], B: list[list[float]]) -> list
    [list[float]]:
    if len(A[0]) != len(B):
        raise ValueError("Size mismatch")
    n: int = len(B)

    res: list[list[float]] = [
        [
            0. for _ in range(len(B[0]))
        ] for _ in range(len(A))
    ]

    Harr: list[float] = [0. for _ in range(len(A))]
    Varr: list[float] = [0. for _ in range(len(B[0]))]

    for j in range(n // 2):
        Harr[0] = Harr[0] + A[0][2*j] * A[0][2*j + 1]

    for i in range(1, len(A)):
        for j in range(n // 2):
            Harr[i] = Harr[i] + A[i][2*j] * A[i][2*j + 1]

    #

    for j in range(n // 2):
        Varr[0] = Varr[0] + B[2*j][0] * B[2*j + 1][0]

    for i in range(1, len(B[0])):
        for j in range(n // 2):
            Varr[i] = Varr[i] + B[2*j][i] * B[2*j + 1][i]

    #

    for j in range(len(B[0])):
        res[0][j] = -Harr[0] - Varr[j]
        for k in range(0, n - 1, 2):
            res[0][j] = res[0][j] + (A[0][k] + B[k+1][j]) * \
                (A[0][k+1] + B[k][j])
        if n % 2 != 0:
            res[0][j] += A[0][-1] * B[-1][j]
```

```
    for i in range(1, len(A)):
        for j in range(len(B[0])):
            res[i][j] = -Harr[i] - Varr[j]
            for k in range(0, n - 1, 2):
                res[i][j] = res[i][j] + (A[i][k] + B[k+1][j]) * \
                    (A[i][k+1] + B[k][j])
            if n % 2 != 0:
                res[i][j] = res[i][j] + A[i][-1] * B[-1][j]

    return res
```

3.3 Функциональные тесты

A	B	A×B	Результат выполнения алгоритма		
			Классического	Винограда	Винограда с опт.
$\begin{pmatrix}1 & 2\\3 & 4\\5 & 6\end{pmatrix}$	$\begin{pmatrix}7 & 8 & 9\\0 & 1 & 2\end{pmatrix}$	$\begin{pmatrix}7 & 10 & 13\\21 & 28 & 35\\35 & 46 & 57\end{pmatrix}$	$\begin{pmatrix}7 & 10 & 13\\21 & 28 & 35\\35 & 46 & 57\end{pmatrix}$	$\begin{pmatrix}7 & 10 & 13\\21 & 28 & 35\\35 & 46 & 57\end{pmatrix}$	$\begin{pmatrix}7 & 10 & 13\\21 & 28 & 35\\35 & 46 & 57\end{pmatrix}$
$\begin{pmatrix}1 & 2\\3 & 4\end{pmatrix}$	$\begin{pmatrix}5 & 6\\7 & 8\end{pmatrix}$	$\begin{pmatrix}19 & 22\\43 & 50\end{pmatrix}$	$\begin{pmatrix}19 & 22\\43 & 50\end{pmatrix}$	$\begin{pmatrix}19 & 22\\43 & 50\end{pmatrix}$	$\begin{pmatrix}19 & 22\\43 & 50\end{pmatrix}$
$\begin{pmatrix}1 & 2\end{pmatrix}$	$\begin{pmatrix}3 & 4\end{pmatrix}$	—	—	—	—

Все тесты пройдены успешно

Вывод

В ходе работы были реализованы алгоритмы классического умножения матриц и умножения матриц Копперсмита-Винограда, а также было проведено их тестирование.

4 Исследовательская часть

4.1 Технические характеристики

Технические характеристики устройства, на котором проводились замеры:

— операционная система: EndeavourOS x86_64;

— процессор: 13th Gen Intel(R) Core(TM) i53500H (16) С частотой 4.70 ГГц;

— оперативная память: 16 ГБ с частотой 5200 МГц.

4.2 Временные характеристики

Замеры проводились на квадратных матрицах размера N×N.

Время приведено в миллисекундах

Замеры проводились с использованием библиотеки time [4]. Замерялось процессорное время.

Результаты замеров приведены на рисунке 4.1

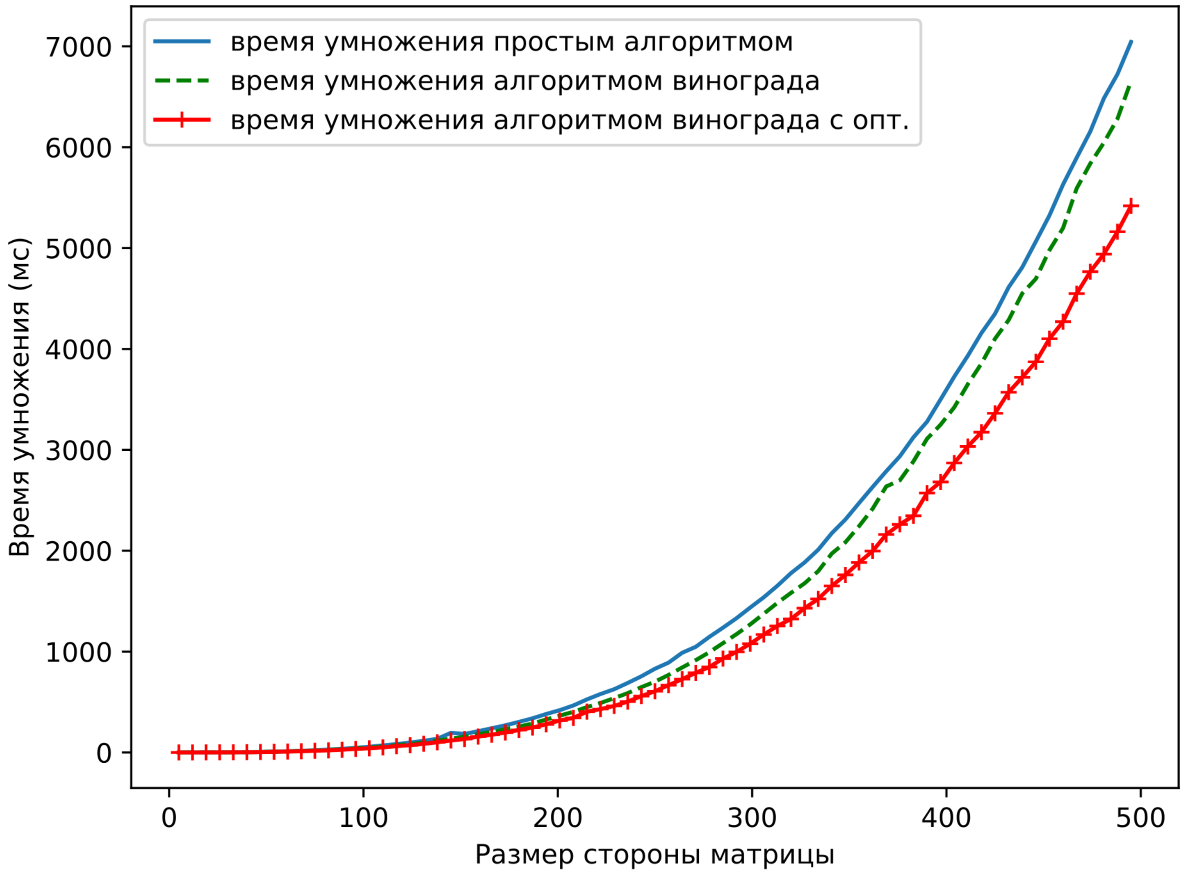

Рисунок 4.1 — Результаты замеров временных затрат алгоритмов

4.3 Вывод

В результате исследования получен следующий результат: временные затраты на умножение двух квадратных матриц алгоритмом Винограда стабильно ниже временных затрат классического алгоритма умножения таких же матриц. Оптимизированная версия алгоритма Винограда даёт видимый прирост в скорости работы алгоритма.

ЗАКЛЮЧЕНИЕ

Были проанализированы и сравнены по временным характеристикам классический алгоритм умножения матриц и алгоритм Копперсмита-Винограда. Оказалось, что алгоритм Копперсмита-Винограда работает быстрее классического алгоритма умножения матриц при умножении двух квадратных матриц.

Были выполнены следующие задачи:

— были рассмотрены существующие алгоритмы умножения матриц;
— рассмотренные алгоритмы умножения матриц были разработаны;
— разработанные алгоритмы были реализованы;
— были проанализированы временные характеристики полученных реализаций.

Цели и задачи лабораторной работы выполнены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация Языка программирования Python / [Электронный ресурс] // Python : [сайт]. — URL: https://www.python.org/ (дата обращения: 25.09.2024).

2. Документация Jupyter Notebook: The Classic Notebook Interface / [Электронный ресурс] // Jupyter : [сайт]. — URL: https://jupyter.org/ (дата обращения: 25.09.2024).

3. Документация библиотеки Matplotlib: Visualization with Python / [Электронный ресурс] // Matplotlib : [сайт]. — URL: https://matplotlib.org/ (дата обращения: 25.09.2024).

4. Документация библиотеки time — Time access and conversions / [Электронный ресурс] // Python 3.12.6 documentation : [сайт]. — URL: https://docs.python.org/3/library/time.html#time.process_time_ns (дата обращения: 25.09.2024).

5. R. Bellman Introduction to Matrix Analysis // 2-е изд. // 1997 год // Филадельфия, SIAM // 426 страниц

6. D. Coppersmith, S. Winograd Matrix multiplication via arithmetic progressions // Journal of Symbolic Computation том 9 номер 3 // 1990 год // страницы с 251 по 280