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»

ДИСЦИПЛИНА «Анализ алгоритмов»

Лабораторная работа № 2

Тема «Алгоритмы умножения матриц»

Студент Чалый А.А.

Группа ИУ7 – 52Б

Оценка (баллы) _____

Преподаватели Волкова Л.Л.
Строганов Ю.В.

Москва.
2020 г.

Оглавление

Введение....3
1. Аналитическая часть....4
1.1 Описание алгоритмов....4
1.2 Классический алгоритм умножения матриц....4
1.3 Алгоритм Винограда....5
2. Конструкторская часть....6
2.1 Разработка алгоритмов....6
2.2 Оптимизации алгоритма Винограда....8
2.3 Трудоемкость алгоритмов....9
2.4 Вывод....11
3.Технологическая часть....12
3.1 Требования к программному обеспечению....12
3.2 Средства реализации....12
3.4 Тестирование....14
3.5 Вывод....16
4. Экспериментальная часть....17
4.1 Постановка эксперимента....17
4.2 Сравнительный анализ на материале экспериментальных данных....17
4.3 Вывод....18
Заключение....19
Список использованных источников....20

Введение

Целью данной работы является изучение алгоритмов умножения матриц, получение практических навыков реализации алгоритмов умножения матриц, приобретение навыков расчета трудоемкости алгоритмов, а также получение навыков в улучшении алгоритмов.

Требуется 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 [2].

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

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

1.2 Классический алгоритм умножения матриц

Пусть даны две прямоугольные матрицы A и B размерности n1 × m1 и n2 × m2 соответственно:

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

Тогда матрица C размерностью l × n называется их произведением:

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

в которой:

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

.

1.3 Алгоритм Винограда

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

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

Это равенство можно переписать в виде:

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

Может показаться, что выражение (2) задает больше работы, чем (1): вместо четырех умножений совершается шесть, а также сложений, вместо трех – 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 дополнительные два сложения. Важным моментом данного алгоритма считается то, что при нечетном количестве столбцов первой матрицы требуется к каждому элементу результирующей матрицы добавить результат произведения последнего столбца первой матрицы на последнюю строку второй матрицы.

1.4 Вывод

Были рассмотрены 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

На рис. 1-3 приведены схемы указанных алгоритмов.

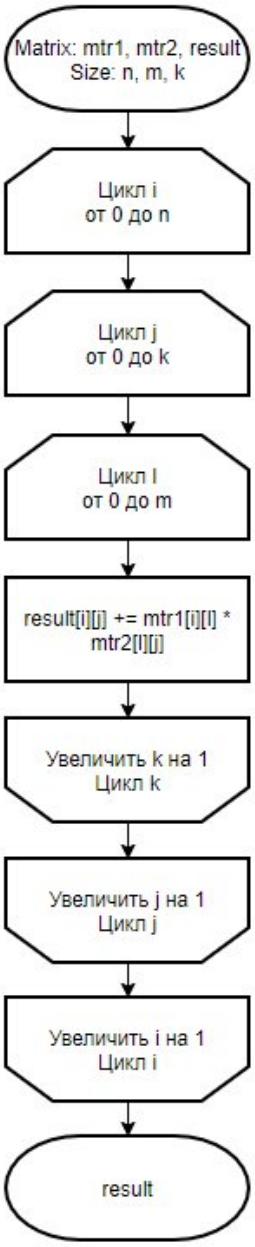

Рисунок 1. Схема алгоритма классического перемножения матриц

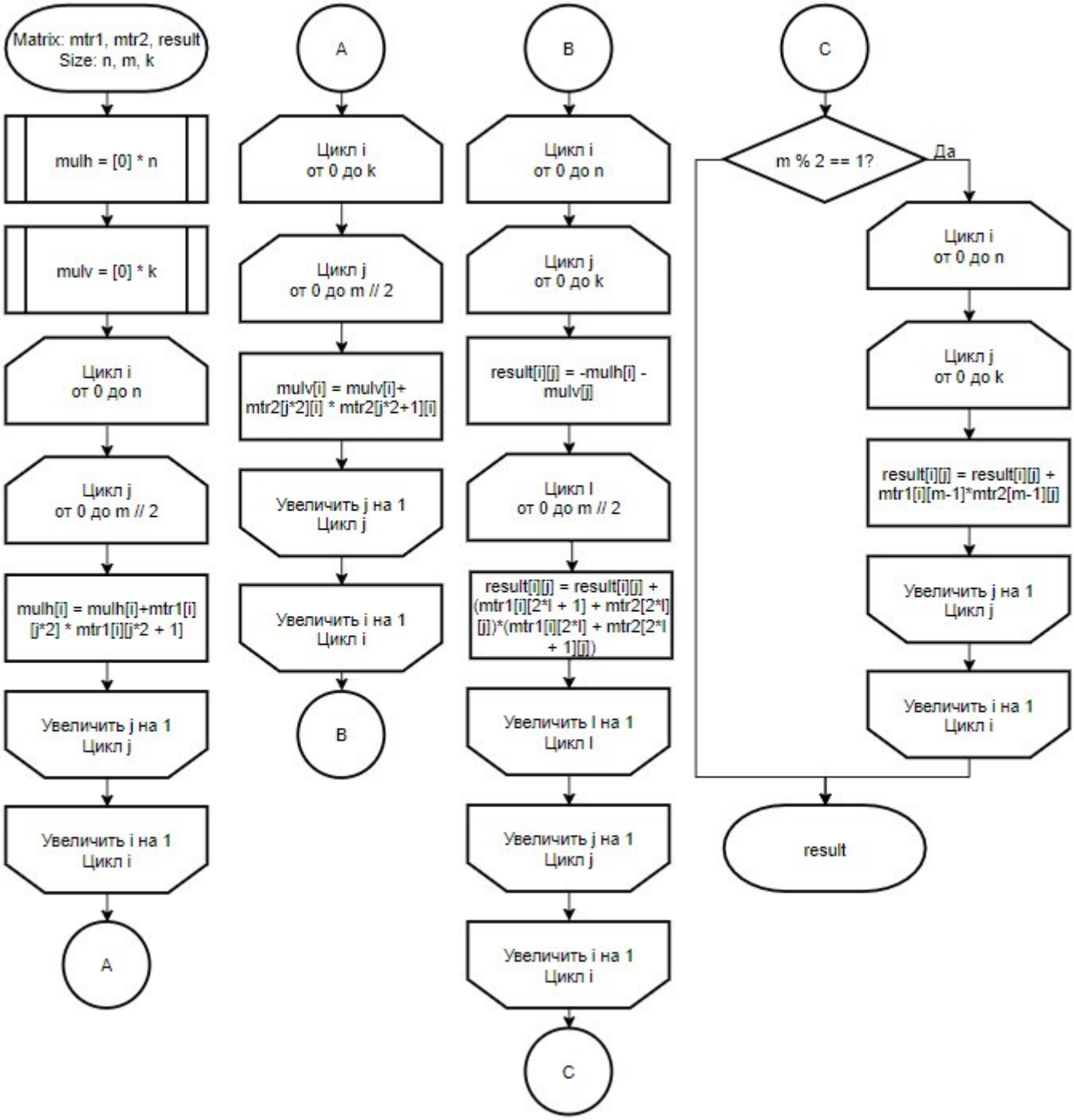

Рисунок 2. Схема алгоритма Винограда по умножению матриц

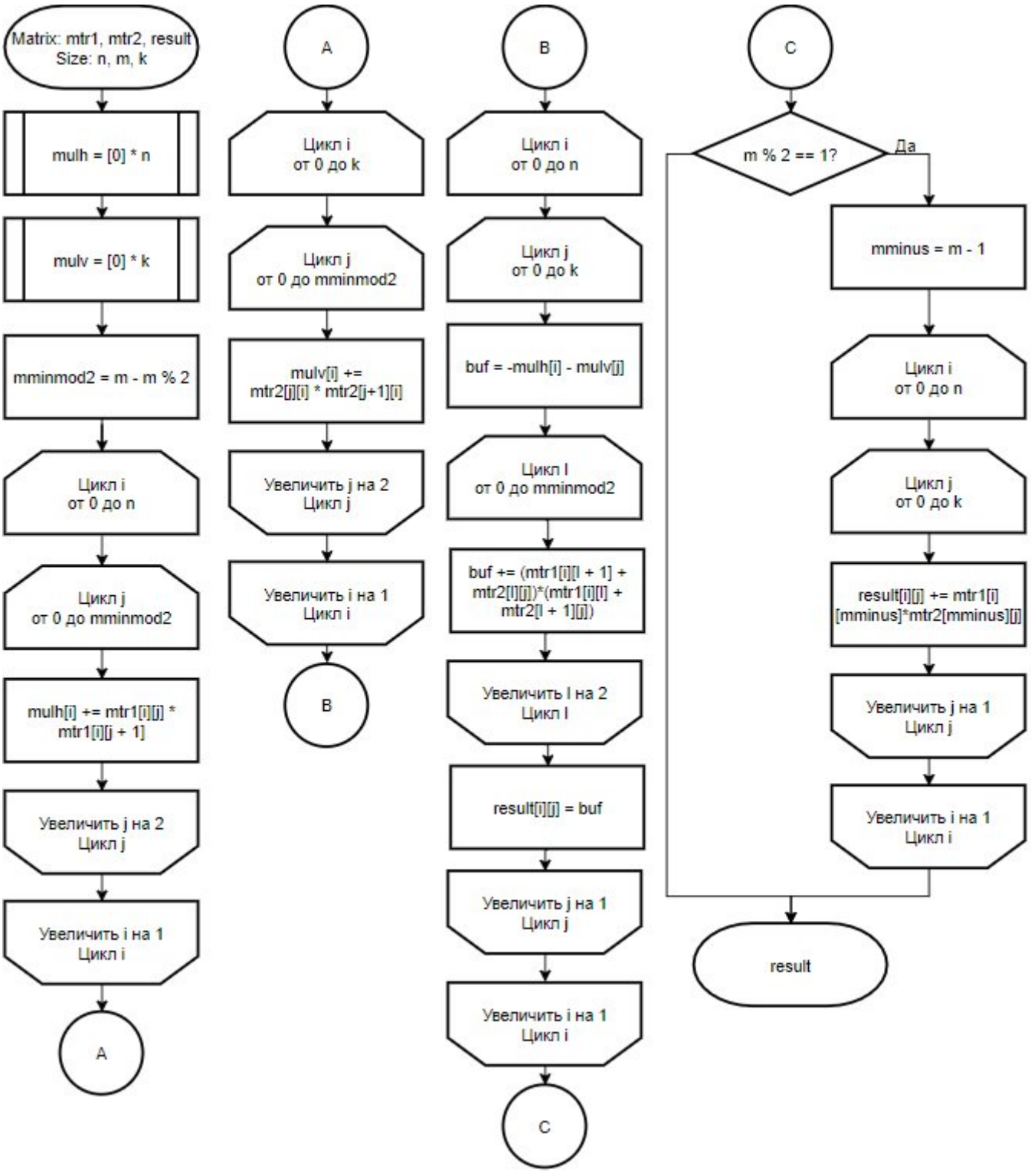

Рисунок 3. Схема усовершенствованного алгоритма Винограда по умножению матриц

2.2 Оптимизации алгоритма Винограда

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

1) Заменить конец цикла j с m//2, на переменную, инициализированную заранее mminmod2 = m - m % 2 и j += 1 изменить на j += 2. Таким образом можно избавиться от постоянного деления в цикле.
2) Заменить все выражения типа mulh[i] = mulh[i] + ... на mulh[i] +=
3) Внутри тройного цикла накапливать результат в буфер, а вне цикла сбрасывать буфер в ячейку матрицы.
4) Заменить m - 1 в цикле, работающем при нечетном количестве столбцов первой матрицы, на переменную mminus1 = m - 1. Таким образом можно избавиться от постоянного вычитания в цикле.

2.3 Трудоемкость алгоритмов

Введем модель трудоемкости для оценки алгоритмов:

1. базовые операции стоимостью 1 — +, -, *, /, =, ==, <=, >=, !=, +=, [];
2. оценка трудоемкости цикла $F_{\text{ц}} = 2 + N * (2 + F_{\text{тела}})$
3. Стоимость условного перехода возьмем за 0, стоимость вычисления условия остаётся.

В приведенных ниже таблицах N, M, K — константы для матриц A с размерами NxM и B с размерами MxK. Также в таблицах 2 и 3 выражение в [] принимает значение сверху, если M четное, а а нижнее — иначе.

В таблицах 1–3 приведены оценки трудоемкости алгоритмов.

Таблица 1. Оценка трудоемкости классического алгоритма.

Трудоемкость	Оценка трудоемкости
F классич.	$2 + N(2 + 2 + K(2 + 2 + M(2 + F_{\text{тела}})))$
F тела	8
F класич.	(10 * N * M * K) + (4 * N * K) + (4 * N) + 2

Таблица 2. Оценка трудоемкости алгоритма Винограда.

Трудоемкость	Оценка трудоемкости
F инициализации mulH и mulV	2 * 3
F заполнения mulH	2 + N * (2 + 2 + M / 2 * (3 + 6 + 6))
F заполнения mulV	2 + K * (2 + 2 + M / 2 * (3 + 6 + 6))
F подсчета результата	2 + N * (2 + 2 + K * (2 + 7 + 2 + M / 2 * (3 + 23)))
F условия нечетности m	2
F для матриц с нечетным m	2 + N *(2 + 2 + K * (2 + 8 + 5))
F Винограда	13 * M * N * K + 7.5 * M * N + 7.5 * K * M + 11 * K * N + 8 * N + 4 * K + 14 + $\begin{bmatrix} 0 \\ 15 * K * N + 4 * N + 2 \end{bmatrix}$

Таблица 3. Оценка трудоемкости усовершенствованного алгоритма Винограда.

Части алгоритма	Трудоемкость
F иниц. mulH и mulV	2 * 3
Переменная mminmod2	3
F заполнения mulH	2 + N * (2 + 2 + M / 2 * (3 + 6 + 6))
F заполнения mulV	2 + K * (2 + 2 + M / 2 * (3 + 6 + 6))
F подсчета результата	2 + N * (2 + 2 + K * (2 + 5 + 3 + 2 + M / 2 * (2 + 14)))
F условия нечетности m	2
F для матриц с нечетным m	2 + 2 + N * (2 + 2 +K * (2 + 6 + 2))
F усоверш. Винограда	8 * M * N * K + 5 * M * N + 5 * K * M + 12 * K * N + 8 * N + 4 * K + 17 + $\begin{bmatrix} 0 \\ 10 * K * N + 4 * n + 4 \end{bmatrix}$

2.4 Вывод

В данном разделе были построены схемы алгоритмов:

1) классического умножения матриц;
2) умножения матриц Винограда;
3) усовершенствованное умножение матриц Винограда.

Также были описаны оптимизации, которые были применены в целях усовершенствования алгоритма Винограда. Была проведена оценка трудоемкости алгоритмов, в ходе которой было выяснено, что улучшенным 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 матрица, mtr2 – вторая матрица.

Выходные данные: произведение двух матриц.

Функциональная схема процесса перемножения двух матриц в нотации IDEF0 представлена на рис. 4.

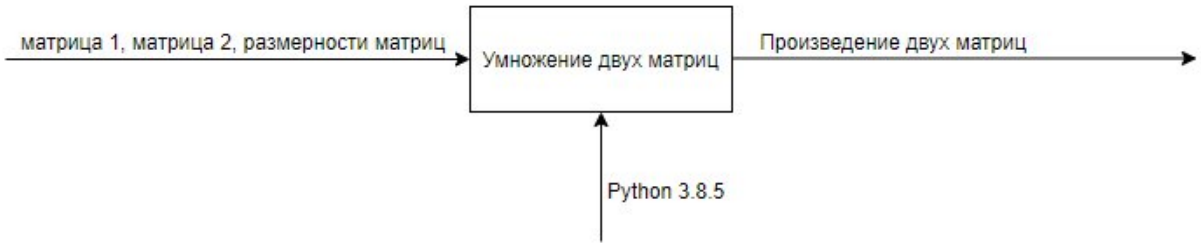

Рисунок 4. Функциональная схема процесса перемножения двух матриц

3.2 Средства реализации

В данной работе был использован язык программирования Python. Проект выполнен в IDE PyCharm с использованием сторонней библиотеки для создания массивов numpy.

Для замера процессорного времени была использована python библиотека time [1].

3.3 Листинг кода

В данном пункте представлен листинг кода функций:

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

Листинг 1. Функция классического умножения матриц

```
def classic_mult(resmatr, matr1, matr2, n, m, k):
    i = 0
    while i < n:
        j = 0
        while j < k:
```

```
            l = 0
            while l < m:
                resmatr[i][j] += matr1[i][l] * matr2[l][j]
                l += 1
            j += 1
        i += 1
    return resmatr
```

Листинг 2. Функция умножения матриц Винограда

```
def Vinograd_mult(resmatr, matr1, matr2, n, m, k):
    mulh = [0] * n
    mulv = [0] * k

    i = 0
    while i < n:
        j = 0
        while j < m//2:
            mulh[i] = mulh[i] + matr1[i][j * 2] * matr1[i][j * 2 + 1]
            j += 1
        i += 1

    i = 0
    while i < k:
        j = 0
        while j < m//2:
            mulv[i] = mulv[i] + matr2[j * 2][i] * matr2[j * 2 + 1][i]
            j += 1
        i += 1

    i = 0
    while i < n:
        j = 0
        while j < k:
            resmatr[i][j] = -mulh[i] - mulv[j]
            l = 0
            while l < m // 2:
                resmatr[i][j] = resmatr[i][j] + \
                                (matr1[i][2 * l + 1] + matr2[2 * l][j]) *
(matr1[i][2 * l] + matr2[2 * l + 1][j])
                l += 1
            j += 1
        i += 1

    if m % 2 == 1:
        i = 0
        while i < n:
            j = 0
            while j < k:
                resmatr[i][j] = resmatr[i][j] + matr1[i][m - 1] * matr2[m -
1][j]
                j += 1
            i += 1
    return resmatr
```

Листинг 3. Функция умножения матриц Винограда (улучшенная)

```
def Vinograd_opt(resmatr, matr1, matr2, n, m, k):
    mulh = [0] * n
    mulv = [0] * k

    mminmod2 = m - m % 2
    i = 0
    while i < n:
        j = 0
        while j < mminmod2:
            mulh[i] += matr1[i][j] * matr1[i][j + 1]
            j += 2
        i += 1

    i = 0
    while i < k:
        j = 0
        while j < mminmod2:
            mulv[i] += matr2[j][i] * matr2[j + 1][i]
            j += 2
        i += 1

    i = 0
    while i < n:
        j = 0
        while j < k:
            buf = -mulh[i] - mulv[j]
            l = 0
            while l < mminmod2:
                buf += (matr1[i][l + 1] + matr2[l][j]) * (matr1[i][l] +
matr2[l + 1][j])
                l += 2
            resmatr[i][j] = buf
            j += 1
        i += 1

    if m % 2 == 1:
        i = 0
        mminus = m - 1
        while i < n:
            j = 0
            while j < k:
                resmatr[i][j] += matr1[i][mminus] * matr2[mminus][j]
                j += 1
            i += 1
    return resmatr
```

3.4 Тестирование

В тестах будет рассмотрена проверка работы алгоритмов при:

- нулевой матрице;
- единичной матрице;

- матриц, содержащих отрицательные числа;
- умножении матрицы на вектор (частный случай матрицы).

Тестирование производится методом черного ящика.

На рисунках ниже будут приведены тесты с целью демонстрации корректности работы программы.

```
Матрица A
0 0 0
0 0 0
0 0 0
Матрица B
-2 4 2
0 4 4
1 -1 5
Результат AXB (классический алгоритм)
0 0 0
0 0 0
0 0 0
Результат AXB (алгоритм Винограда)
0 0 0
0 0 0
0 0 0
Результат AXB (улучшенный алгоритм Винограда)
0 0 0
0 0 0
0 0 0
```

Рисунок 5. Проверка работы алгоритмов при нулевой матрице

```
Матрица A
1 0 0
0 1 0
0 0 1
Матрица B
-4 -4 5
0 -1 4
-3 0 -1
Результат AXB (классический алгоритм)
-4 -4 5
0 -1 4
-3 0 -1
Результат AXB (алгоритм Винограда)
-4 -4 5
0 -1 4
-3 0 -1
```

Рисунок 6. Проверка работы алгоритмов при единичной матрице

```
Матрица A
-2 -1 0
2 -1 0
-1 1 3
Матрица B
-2 4 5
1 1 1
3 3 -3
Результат AXB (классический алгоритм)
3 -9 -11
-5 7 9
12 6 -13
Результат AXB (алгоритм Винограда)
3 -9 -11
-5 7 9
12 6 -13
Результат AXB (улучшенный алгоритм Винограда)
3 -9 -11
-5 7 9
12 6 -13
```

Рисунок 7. Проверка работы алгоритмов при матрицах, содержащих отрицательные числа

```
Матрица A
-2 4 2
0 -4 5
-2 0 4
Матрица B
2
5
2
Результат AXB (классический алгоритм)
20
-10
4
Результат AXB (алгоритм Винограда)
20
-10
4
Результат AXB (улучшенный алгоритм Винограда)
20
-10
4
```

Рисунок 8. Проверка работы алгоритмов при умножении матрицы на вектор

Все тесты пройдены успешно.

3.5 Вывод

В данном разделе была представлена реализация алгоритмов классического перемножения матриц, Винограда, улучшенного Винограда. Также проведено тестирование разработанных методов по методу чёрного ящика.

4. Экспериментальная часть

В данном разделе будет проведен эксперимент и сравнительный анализ полученных данных.

4.1 Постановка эксперимента

В рамках данной части были проведены эксперименты, описанные ниже.

1) Сравнение времени работы трех алгоритмов. Создаются квадратные матрицы размерностью 4 и 8, заполненные случайными числами. Эксперимент по измерению одной размерности ставился 10000 раз;
2) Сравнение времени работы трех алгоритмов. Создаются квадратные матрицы размерностью 5 и 9, заполненные случайными числами. Эксперимент по измерению одной размерности ставился 10000 раз.

4.2 Сравнительный анализ на материале экспериментальных данных

Сравнение времени работы алгоритмов при квадратных матрицах размерностью 4 и 8 и количестве итераций 10000 изображены на рис. 9. и рис. 10.

```
Количество итераций: 10000
Размер матрицы: 4x4
Классическое умножение =  9.6875e-05
Алгоритм Винограда =  0.000125
Улучшенный алгоритм Винограда =  0.0001046875
```

Рисунок 9. Время работы алгоритмов при квадратных матрицах размерностью 4

```
Размер матрицы: 8x8
Классическое умножение =  0.0007421875
Алгоритм Винограда =  0.0008890625
Улучшенный алгоритм Винограда =  0.0006671875
```

Рисунок 10. Время работы алгоритмов при квадратных матрицах размерностью 8

Сравнение времени работы алгоритмов при квадратных матрицах размерностью 5 и 9 и количестве итераций 10000 изображены на рис. 11. и рис. 12.

```
Размер матрицы: 5x5
Классическое умножение =  0.0001828125
Алгоритм Винограда =  0.0002390625
Улучшенный алгоритм Винограда =  0.0001921875
```

Рисунок 11. Время работы алгоритмов при квадратных матрицах размерностью 5

```
Размер матрицы: 9x9
Классическое умножение =  0.00105625
Алгоритм Винограда =  0.0012328125
Улучшенный алгоритм Винограда =  0.00095625
```

Рисунок 12. Время работы алгоритмов при квадратных матрицах размерностью 9

4.3 Вывод

В данном разделе был поставлен эксперимент по замеру времени выполнения каждого алгоритма. По итогам замеров можно сделать вывод о том, что алгоритм Винограда является самым медленным среди всех, когда как его улучшенная версия самая быстрая при средних и больших размерах матриц, а стандартный алгоритм быстрее. при малых размерностях матрицы Тем не менее, все три 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. Выполнена оценка трудоемкости всех вышеперечисленных алгоритмов, в ходе которой был сделан вывод, что улучшенный алгоритм Винограда является самым быстрым алгоритмом по трудоемкости. Так10же были произведены замеры времени выполнения алгоритмов, в котором улучшенный алгоритм Винограда оказался самым быстрым. Изучены зависимости времени выполнения алгоритмов от размерности матриц. Все поставленные задачи были выполнены. Целью лабораторной работы являлось изучение алгоритмов перемножения матриц, что также было достигнуто.

Список использованных источников

1. time – Time access and conversions: сайт. – URL: https://docs.python.org/3/library/time.html (дата обращения 16.09.2020). – Текст: электронный.
2. Дж. Макконнелл. Анализ алгоритмов. Активный обучающий подход. – М.: Техносфера, 2017. – 267 с.