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 Алгоритмы умножения матриц

Студент Артюхин Н.П.

Группа ИУ7-51Б

Преподаватели Волкова Л.Л., Строганов Ю.В.

Москва — 2022 г.

Оглавление

Введение

Цель лабораторной работы – изучение способов оптимизации алгоритмов на примере алгоритмов умножения матриц.

Термин «матрица» применяется в различных областях, но основное значение данный термин имеет в математике.

Матрица – математический объект, записываемый в виде прямоугольной таблицы элементов кольца или поля (например, целых или комплексных чисел), которая представляет собой совокупность строк и столбцов, на пересечении которых находятся её элементы. Количество строк и столбцов матрицы задают размер матрицы [1].

Матрицы часто применяются в математике для компактной записи систем линейных алгебраических или дифференциальных уравнений (количество строк матрицы соответствует числу уравнений, а количество столбцов – количеству неизвестных).

Таким образом, решение систем линейных уравнений сводится к операциям над матрицами, среди которых встречается умножение.

Произведением двух матриц A и B называется матрица C, элемент которой, находящийся на пересечении i-й строки и j-го столбца, равен сумме произведений элементов i-й строки матрицы A на соответствующие (по порядку) элементы j-го столбца матрицы B [2].

Умножение матриц A и B – это операция вычисления этой матрицы C.

Для достижения поставленной цели требуется решить следующие задачи:

1) изучение двух алгоритмов умножения матриц: стандартный алгоритм, алгоритм Винограда;

2) разработка трех алгоритмов умножения матриц: стандартный, Винограда и Винограда с оптимизациями;

3) реализация трех алгоритмов умножения матриц: стандартный, Винограда и Винограда с оптимизациями;

4) выполнение замеров процессорного времени работы реализаций алгоритмов умножения матриц: стандартный, Винограда и Винограда

с оптимизациями;

5) сравнительный анализ трудоемкости реализаций разработанных алгоритмов умножения матриц на основе теоретических расчетов;

6) выполнение оценки затрат алгоритмов умножения матриц по памяти и сравнительный анализ;

7) сравнительный анализ процессорного времени работы реализаций разработанных алгоритмов умножения матриц на основе экспериментальных данных.

1 Аналитическая часть

В данном разделе будут представлены описания алгоритмов умножения матриц: стандартный, Винограда.

1.1 Стандартный алгоритм умножения матриц

Пусть даны две прямоугольные матрицы $A[M \times N]$ и $B[N \times Q]$:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1q} \\ b_{21} & b_{22} & \cdots & b_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nq} \end{bmatrix}$$

Тогда матрица $C[M \times Q]$ – произведение матриц A и B:

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1q} \\ c_{21} & c_{22} & \cdots & c_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mq} \end{bmatrix},$$

в которой каждый элемент вычисляется по следующей формуле:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}, \quad (i = \overline{1, m}; j = \overline{1, q}) \tag{1.1}$$

Стандартный алгоритм реализует данную формулу.

1.2 Алгоритм Винограда умножения матриц

Если посмотреть на результат умножения двух матриц, то видно, что каждый элемент в нем представляет собой скалярное произведение соответствующих строки и столбца исходных матриц. Можно заметить также, что такое умножение допускает предварительную обработку, позволяющую часть работы выполнить заранее.

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 (1.2):

$$V \cdot W = (v_1+w_2)(v_2+w_1)+(v_3+w_4)(v_4+w_3)-v_1v_2-v_3v_4-w_1w_2-w_3w_4. \quad (1.2)$$

Кажется, что второе выражение задает больше работы, чем первое: вместо четырех умножений мы получаем шесть, а вместо трех сложений – десять. Однако выражение в правой части последнего равенства допускает предварительную обработку, а именно его части можно вычислить заранее и запомнить для каждой строки первой матрицы и для каждого столбца второй. На практике это означает, что над предварительно обработанными элементами нам придется выполнять лишь первые два умножения и последующие пять сложений, а также дополнительно два сложения [3].

В конце нужно проверить кратность общей размерности матриц двум. Если она не кратна двум, то нужно добавить к каждому элементу результирующей матрицы произведение последних элементов соответствующих строки и столбца.

Вывод

В данном разделе были рассмотрены основные идеи, лежащие в основе рассматриваемых алгоритмов умножения матриц – стандартного алгоритма и алгоритма Винограда.

2 Конструкторская часть

В данном разделе будут представлены схемы алгоритмов умножения матриц (стандартный, Винограда, оптимизированный Винограда) и вычисления трудоемкости данных алгоритмов.

2.1 Стандартный алгоритм умножения матриц

На рисунке 2.1 приведена схема стандартного алгоритма умножения матриц.

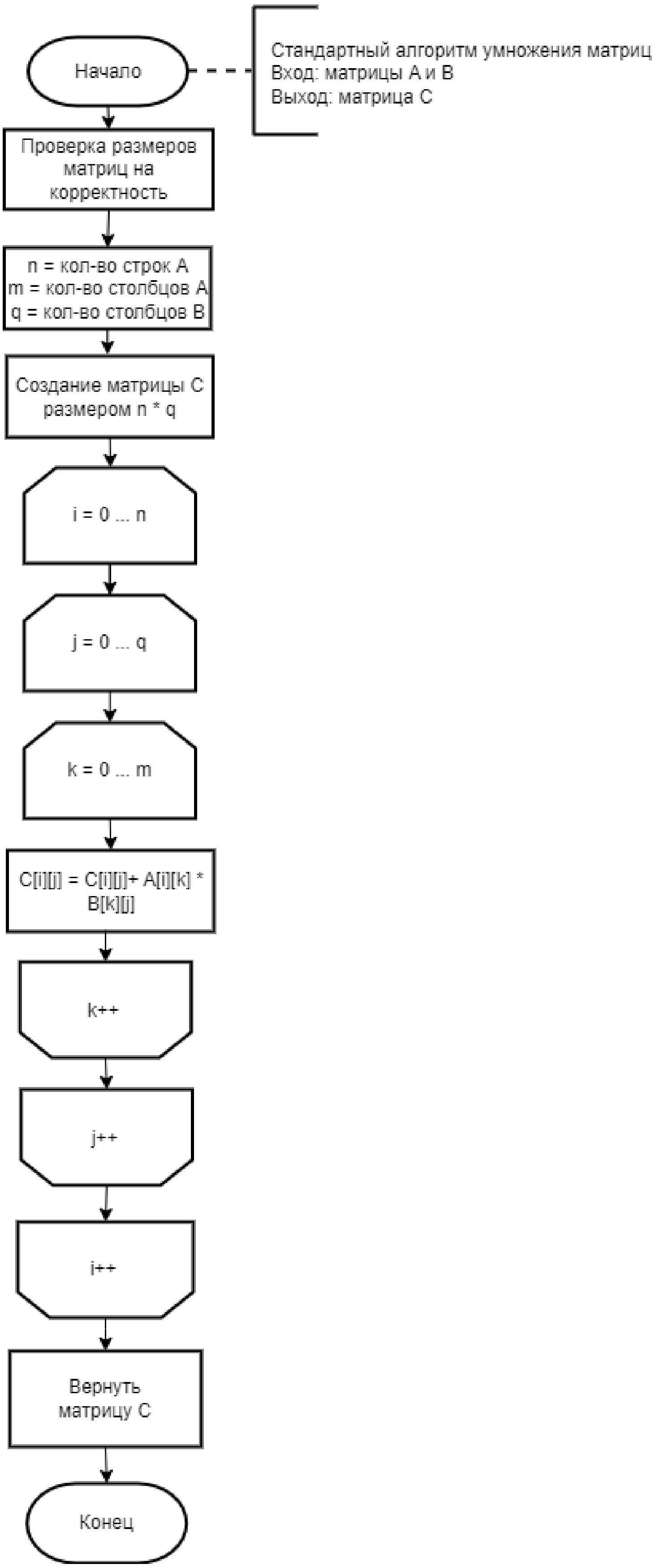

Рисунок 2.1 – Схема стандартного алгоритма умножения матриц

2.2 Алгоритм Винограда умножения матриц

На рисунке 2.2 приведена схема алгоритма Винограда умножения матриц. Можно заметить, что для алгоритма Винограда умножения матриц худшим случаем являются матрицы с нечётной общей размерностью, а лучшим – с чётной, так как последний цикл в этом случае не задействуется.

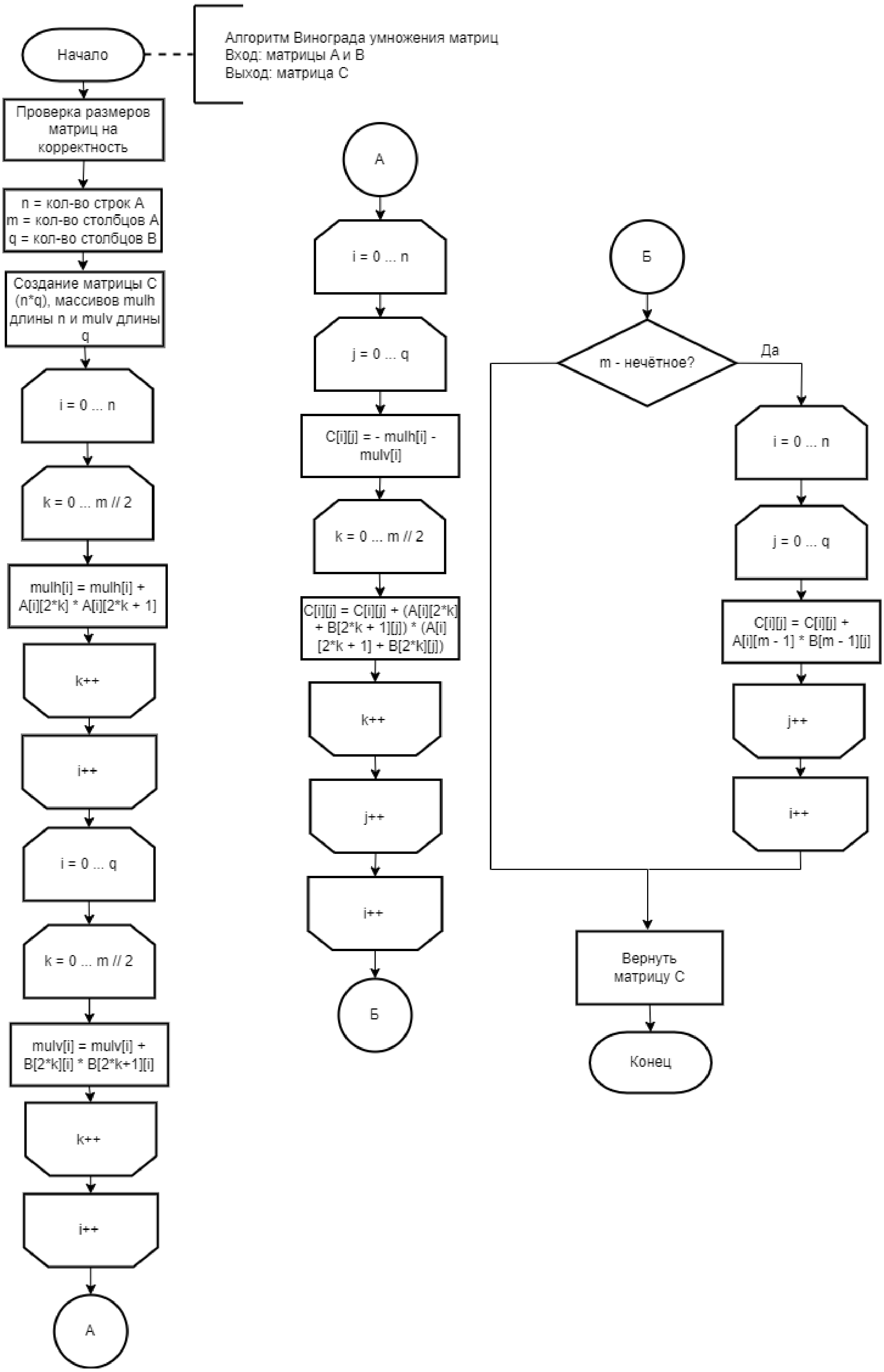

Рисунок 2.2 – Схема алгоритма Винограда умножения матриц

2.3 Оптимизированный алгоритм Винограда умножения матриц

Алгоритм Винограда можно оптимизировать несколькими способами.

1) Заменить все выражения вида x = x + k на x += k;

2) Использовать побитовый сдвиг влево на 1 бит («) вместо умножения на 2;

3) Четвертый (последний) цикл для нечётных элементов алгоритма Винограда объединить с третьим, нечетность общей размерности входных матриц проверить до третьего цикла и установить соответствующий флаг, если флаг содержит значение истина, то выполняем дополнительные операции в третьем цикле (предвычисляем некоторые слагаемые для алгоритма).

На рисунке 2.3 приведена схема оптимизированного алгоритма Винограда умножения матриц.

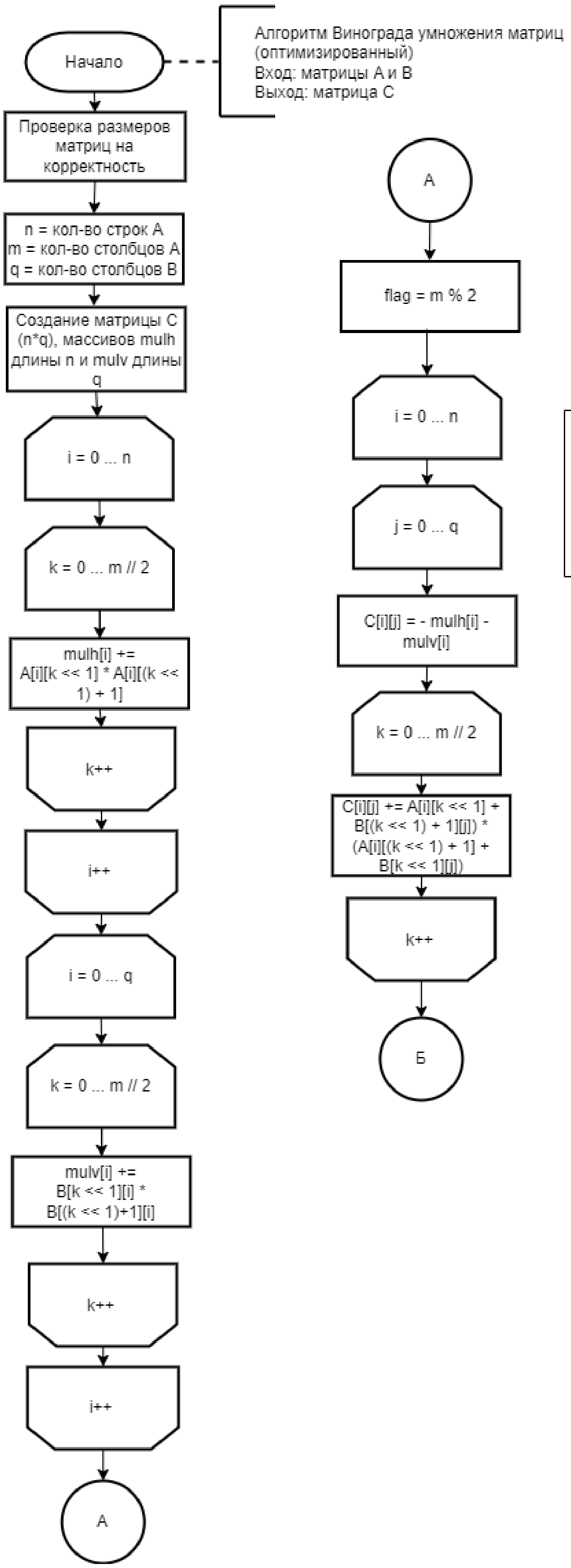

Рисунок 2.3 – Оптимизированный алгоритм Винограда умножения матриц

2.4 Оценка трудоемкости алгоритмов умножения матриц

Для последующего вычисления трудоемкости необходимо ввести следующую модель вычислений:

- базовые операции с трудоемкостью 2: /, /=, *, *=, %, %=;
- базовые операции с трудоемкостью 1: +, ++, +=, -, −−, -=, ==, !=, <, >, <=, >=, «, », [];
- трудоемкость цикла: Fцикла = Fиниц. + Fсравн. + M * (Fтела + Fинк. + Fсравн.), где Fиниц., Fсравн., Fтела, Fинк. – трудоемкости инициализации, проверки условия цикла, тела цикла и инкрементирования соответственно, а M – количество итераций;
- трудоемкость условного опeратроа: F_{if} = Fсравн. +

$$+\left[\begin{array}{l} min(f1, f2) - \text{лучший случай} \\ max(f1, f2) - \text{худший случай} \end{array}\right., \tag{2.1}$$

  где Fсравн., f1, f2 – трудоемкости проверки условия, первого блока и второго блока, соответственно.

Обозначим во всех последующих вычислениях размерности входных матриц (A - N x M, B - M x Q) как N, M, Q.

Во всех рассматриваемых алгоритмах умножения матриц не будем учитывать проверку размерностей входных матриц A и B на корректность для операции умножения и инициализацию матрицы C, в которую записывается результат, так как данные действия являются общими для всех алгоритмов и имеют незначительную для всего алгоритма трудоемкость.

2.4.1 Стандартный алгоритм умножения матриц

Трудоёмкость стандартного алгоритма умножения матриц состоит из:

- внешнего цикла по $i \in [1..N]$, трудоёмкость которого: $f = 2 + N \cdot (2 + f)$;

- внутреннего цикла по $j \in [1..Q]$, трудоёмкость которого: $f = 2 + Q \cdot (2 + f)$;

- внутреннего цикла по $k \in [1..M]$, трудоёмкость которого: $f = 2 + 14M$;

Суммарная трудоёмкость стандартного алгоритма умножения матриц:

$$f_{standard} = 2 + N \cdot (4 + Q \cdot (4 + 14M)) = 2 + 4N + 4NQ + 14NMQ \approx 14NMQ \tag{2.2}$$

2.4.2 Алгоритм Винограда умножения матриц

Трудоёмкость алгоритма Винограда умножения матриц состоит из:

- создания и инициализации массивов MulH и MulV, трудоёмкость которого:

$$f_{init} = N + Q; \tag{2.3}$$

- заполнения массива MulH, трудоёмкость которого:

$$f_{MulH} = 2 + N \cdot (3 + \frac{M}{2} \cdot 15); \tag{2.4}$$

- заполнения массива MulV, трудоёмкость которого:

$$f_{MulV} = 2 + Q \cdot (3 + \frac{M}{2} \cdot 15); \tag{2.5}$$

- цикла заполнения для чётных размеров, трудоёмкость которого:

$$f_{for3} = 2 + N \cdot (4 + Q \cdot (12 + \frac{M}{2} \cdot 29)); \tag{2.6}$$

- цикла, для дополнения результата умножения суммой последних нечётных строки и столбца, если общая размерность входных матриц нечётная, трудоемкость которого:

$$f_{for4} = 3 + \begin{cases} 0, & \text{л.с.} \\ 4 + M \cdot (4 + 16N), & \text{х.с.} \end{cases} \tag{2.7}$$

Трудоемкость в худшем случае (нечётная общая размерность матриц):

$$f = f_{MulH} + f_{MulV} + f_{for3} + f_{for4} \approx 14.5 \cdot NMQ \tag{2.8}$$

Трудоемкость в лучшем случае (чётная общая размерность матриц):

$$f = f_{MulH} + f_{MulV} + f_{for3} + f_{for4} \approx 14.5 \cdot NMQ \tag{2.9}$$

2.4.3 Оптимизированный алгоритм Винограда умножения матриц

Трудоёмкость оптимизированного алгоритма Винограда умножения матриц состоит из:

- создания и инициализации массивов MulH и MulV, трудоёмкость которого:
$$f_{init} = N + Q; \tag{2.10}$$
- заполнения массива MulH, трудоёмкость которого:
$$f_{MulH} = 2 + N \cdot (3 + \frac{M}{2} \cdot 12); \tag{2.11}$$
- заполнения массива MulV, трудоёмкость которого:
$$f_{MulV} = 2 + Q \cdot (3 + \frac{M}{2} \cdot 12); \tag{2.12}$$
- цикла заполнения для чётных размеров, трудоёмкость которого:
$$f_{for3} = 2 + N \cdot (4 + Q \cdot (12 + \frac{M}{2} \cdot 24)); \tag{2.13}$$

- условие, для дополнения умножения суммой последних нечётных строки и столбца, если общая размерность входных матриц нечётная, трудоемкость которого:

$$f_{if} = NQ + \begin{cases} 0, & \text{л.с.} \\ N \cdot (4 + 12Q), & \text{х.с.} \end{cases} \tag{2.14}$$

Трудоемкость в худшем случае (нечётная общая размерность матриц)::

$$f = f_{MulH} + f_{MulV} + f_{for3} + f_{if} \approx 12NMQ \tag{2.15}$$

Трудоемкость в лучшем случае (чётная общая размерность матриц):

$$f = f_{MulH} + f_{MulV} + f_{for3} + f_{if} \approx 12NMQ \tag{2.16}$$

Вывод

В данном разделе были разработаны схемы алгоритмов умножения матриц (стандартный, Винограда, оптимизированный Винограда). Для каждого из них были оценены трудоемкости в лучшем и худшем случаях.

3 Технологическая часть

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

3.1 Требования к программному обеспечению

Вход: две матрицы A и B, количество строк и стобцов в данных матрицах, причем количество столбцов в матрице A должно быть равно количеству строк в матрице B.;

Выход: матрица C – результат умножения матриц A и B.

Результат умножения введенных пользователем матриц должен быть выведен для каждого из реализованных алгоритмов умножения матриц (стандартный, Винограда, оптимизированный Винограда).

3.2 Выбор средств реализации

В качестве языка программирования для реализации данной лабораторной работы был выбран язык программирования Python [4]. Данный язык ускоряет процесс разработки и удобен в использовании.

Процессорное время реализованных алгоритмов было замерено с помощью функции process_time() из библиотеки time [5].

В качестве среды разработки был выбран PyCharm Professional [6]. Данная среда разработки является кросс-платформенной, предоставляет функциональный отладчик, средства для рефакторинга кода и возможность быстрой установки необходимых библиотек при необходимости.

3.3 Реализация алгоритмов

В листингах 3.1 – 3.3 представлены реализации различных алгоритмов умножения матриц: стандартный, Винограда, оптимизированный Винogрада.

Листинг 3.1 – Функция стандартного алгоритма умножения матриц

```
def standart_mult_matr(matr1, matr2):
    if len(matr1[0]) != len(matr2):
        print("Incorrect size of matrices!")
        return -1

    n = len(matr1)
    m = len(matr1[0])
    q = len(matr2[0])

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

    for i in range(n):
        for j in range(q):
            for k in range(m):
                res[i][j] = res[i][j] + matr1[i][k] * matr2[k][j]

    return res
```

Листинг 3.2 – Функция алгоритма Винограда умножения матриц

```
def vinograd_mult_matr(matr1, matr2):
    if len(matr1[0]) != len(matr2):
        print("Incorrect size of matrices!")
        return -1

    n = len(matr1)
    m = len(matr1[0])
    q = len(matr2[0])

    res = [[0] * q for i in range(n)]
    # vector of rows matrix A
    mulh = [0] * n
    for i in range(n):
        for k in range(m // 2):
            mulh[i] = mulh[i] + matr1[i][2 * k] * matr1[i][2 * k +
                1]

    # vector of columns matrix B
    mulv = [0] * q
    for i in range(q):
```

```
        for k in range(m // 2):
            mulv[i] = mulv[i] + matr2[2 * k][i] * matr2[2 * k +
                1][i]

    # filling matrix C
    for i in range(n):
        for j in range(q):
            res[i][j] = - mulh[i] - mulv[j]
            for k in range(m // 2):
                res[i][j] = res[i][j] + (matr1[i][2 * k] + matr2[2
                    * k + 1][j]) * \
                            (matr1[i][2 * k + 1] + matr2[2 * k][j
                                ])

    if m % 2 == 1:
        for i in range(n):
            for j in range(q):
                res[i][j] = res[i][j] + matr1[i][m - 1] * matr2[m
                    - 1][j]

    return res
```

Листинг 3.3 – Функция оптимизированного алгоритма Винограда умножения матриц

```
def vinograd_optimized_mult_matrix(matr1, matr2):
    if len(matr1[0]) != len(matr2):
        print("Incorrect size of matrices!")
        return -1

    n = len(matr1)
    m = len(matr1[0])
    q = len(matr2[0])

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

    # vector of rows matrix A
    mulh = [0] * n
    for i in range(n):
        for k in range(m // 2):
            mulh[i] += matr1[i][k << 1] * matr1[i][(k << 1) + 1]

```

```
    # vector of columns matrix B
    mulv = [0] * q
    for i in range(q):
        for k in range(m // 2):
            mulv[i] += matr2[k << 1][i] * matr2[(k << 1) + 1][i]

    flag = m % 2

    # filling matrix C
    for i in range(n):
        for j in range(q):
            res[i][j] = - mulh[i] - mulv[j]
            for k in range(m // 2):
                res[i][j] += (matr1[i][k << 1] + matr2[(k << 1) + 1][j]) * \
                             (matr1[i][(k << 1) + 1] + matr2[k << 1][j])
            if flag == 1:
                res[i][j] += matr1[i][m - 1] * matr2[m - 1][j]
    return res
```

3.4 Тестирование

В таблице 3.1 приведены функциональные тесты для стандартного алгоритма умножения матриц, алгоритма Винограда и оптимизированного алгоритма Винограда. Тесты пройдены успешно.

Таблица 3.1 – Тестирование функций умножения матриц

Матрица A	Матрица B	Ожидаемый результат
$\begin{pmatrix}1 & 1 & 1\\ 2 & 2 & 2\\ 3 & 3 & 3\end{pmatrix}$	$\begin{pmatrix}1\\ 1\\ 1\end{pmatrix}$	$\begin{pmatrix}3\\ 6\\ 9\end{pmatrix}$
$\begin{pmatrix}1 & 1\end{pmatrix}$	$\begin{pmatrix}1\\ 1\end{pmatrix}$	$\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix}$
$\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix}$	$\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix}$	$\begin{pmatrix}2 & 2\\ 2 & 2\end{pmatrix}$
$\begin{pmatrix}7\end{pmatrix}$	$\begin{pmatrix}7\end{pmatrix}$	$\begin{pmatrix}49\end{pmatrix}$
$\begin{pmatrix}1 & 2 & 3 & 4\\ 5 & 6 & 7 & 8\\ 9 & 1 & 2 & 3\end{pmatrix}$	$\begin{pmatrix}6 & 7 & 9\\ -1 & -3 & 4\\ 3 & 7 & 9\\ 17 & -5 & 6\end{pmatrix}$	$\begin{pmatrix}81 & 2 & 68\\ 181 & 26 & 180\\ 110 & 59 & 121\end{pmatrix}$
$\begin{pmatrix}1\\ 2\end{pmatrix}$	$\begin{pmatrix}0\end{pmatrix}$	$\begin{pmatrix}0\\ 0\end{pmatrix}$
$\begin{pmatrix}1 & 2 & 3\end{pmatrix}$	$\begin{pmatrix}4 & 5\\ 6 & 7\end{pmatrix}$	Некорректные размеры матриц!

Вывод

В данном разделе были представлены требования к программному обеспечению и средства реализации, реализованы и протестированы алгоритмы умножения матриц: стандартный, Винограда, оптимизированный Винограда.

4 Исследовательская часть

В текущем разделе будут представлены примеры работы разработанного программного обеспечения, постановка эксперимента и сравнительный анализ реализованных алгоритмов.

4.1 Пример работы программного обеспечения

На рисунке 4.1 представлен результат работы программы. На вход программы подаются две матрицы размерностей 3 на 4 и 4 на 3 соответственно. Первая матрица – [[1, 2, 3, 4], [5, 6, 7, 8], [9, 1, 2, 3]], вторая матрица – [[6, 7, 9], [-1, -3, 4], [3, 7, 9], [17, -5, 6]]. В результате выполнения программа выводит матрицу, полученную умножением двух введенных пользователем матриц различными алгоритмами (стандартный, Винограда, оптимизированный Винограда).

```
Ввод матрицы A
Введите кол-во строк: 3
Введите кол-во столбцов: 4
1 2 3 4
5 6 7 8
9 1 2 3
Ввод матрицы B
Введите кол-во строк: 4
Введите кол-во столбцов: 3
6 7 9
-1 -3 4
3 7 9
17 -5 6
Результат (стандартный алгоритм умножения матриц):
[81, 2, 68]
[181, 26, 180]
[110, 59, 121]
Результат (алгоритм Винограда умножения матриц):
[81, 2, 68]
[181, 26, 180]
[110, 59, 121]
Результат (оптимизированный алгоритм Винограда умножения матриц):
[81, 2, 68]
[181, 26, 180]
[110, 59, 121]
```

Рисунок 4.1 – Пример работы программы

4.2 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование:

- операционная система: Windows 10 [7];
- оперативная память: 16 Гб;
- процессор: Intel® Core™ i5 10300H 2.5 ГГц.

Во время тестирования ноутбук был включен в сеть питания и нагружен только встроенными приложениями окружения и системой тестирования.

4.3 Время выполнения реализаций алгоритмов

Замеры процессорного времени реализованных алгоритмов сортировки (гномья сортировка, поразрядная сортировка, сортировка выбором) проводились с помощью функции process_time() из библиотеки time языка Python.

Функция process_time() возвращает время в секундах (сумму системного и пользовательского процессорного времени).

Замеры времени для каждой четной размерности входных матриц (от 100 x 100 до 500 x 500 с шагом 100) и нечетной (от 101 x 101 до 501 x 501 с шагом 100) проводились 10 раз для всех трех реализованных алгоритмов умножения матриц. В качестве результата бралось среднее время работы алгоритма на каждой длине массива.

На рисунке 4.2 представлено сравнение процессорного времени работы реализаций алгоритмов умножения матриц на матрицах четной размерности (n * n).

На графике видно, что оптимизированный алгоритм Винограда является самым эффективным по времени среди реализованных алгоритмов и его преимущество растет с увеличением размера матрицы, на втором месте по эффективности – алгоритм Винограда в классической реализации, самый неэффективный по времени – стандартный алгоритм умножения матриц, но он практически не уступает по времени алгоритму Винограда без оптимизации.

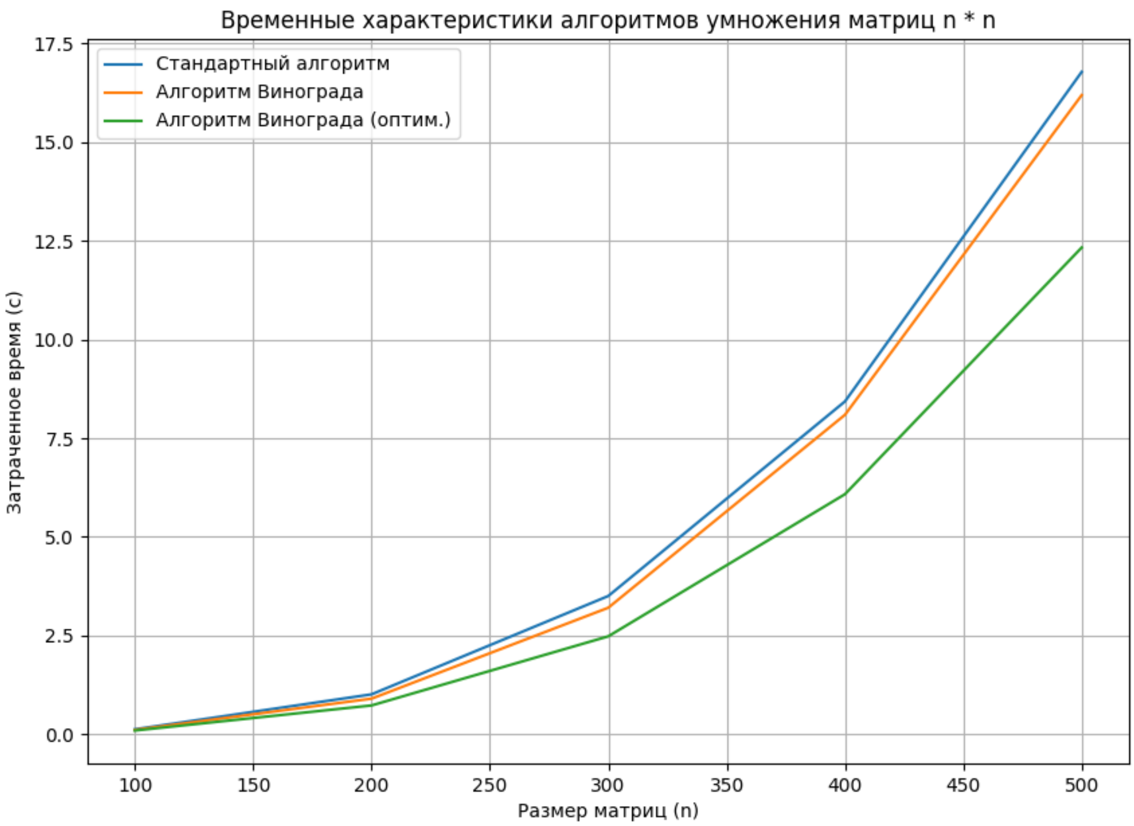

Рисунок 4.2 – Сравнение процессорного времени работы реализаций алгоритмов умножения матриц на квадратных матрицах четной размерности (n * n)

На рисунке 4.3 представлено сравнение процессорного времени работы реализаций алгоритмов умножения матриц на матрицах нечетной размерности (n + 1) * (n + 1). На графике видно, что оптимизированный алгоритм Винограда является самым эффективным по времени среди реализованных алгоритмов и его преимущество растет с увеличением размера матрицы (несмотря на то, что это худший случай для данного алгоритма), на втором месте по эффективности – алгоритм Винограда в классической реализации, самый неэффективный по времени – стандартный алгоритм умножения матриц, но он практически не уступает по времени алгоритму Винограда без оптимизации.

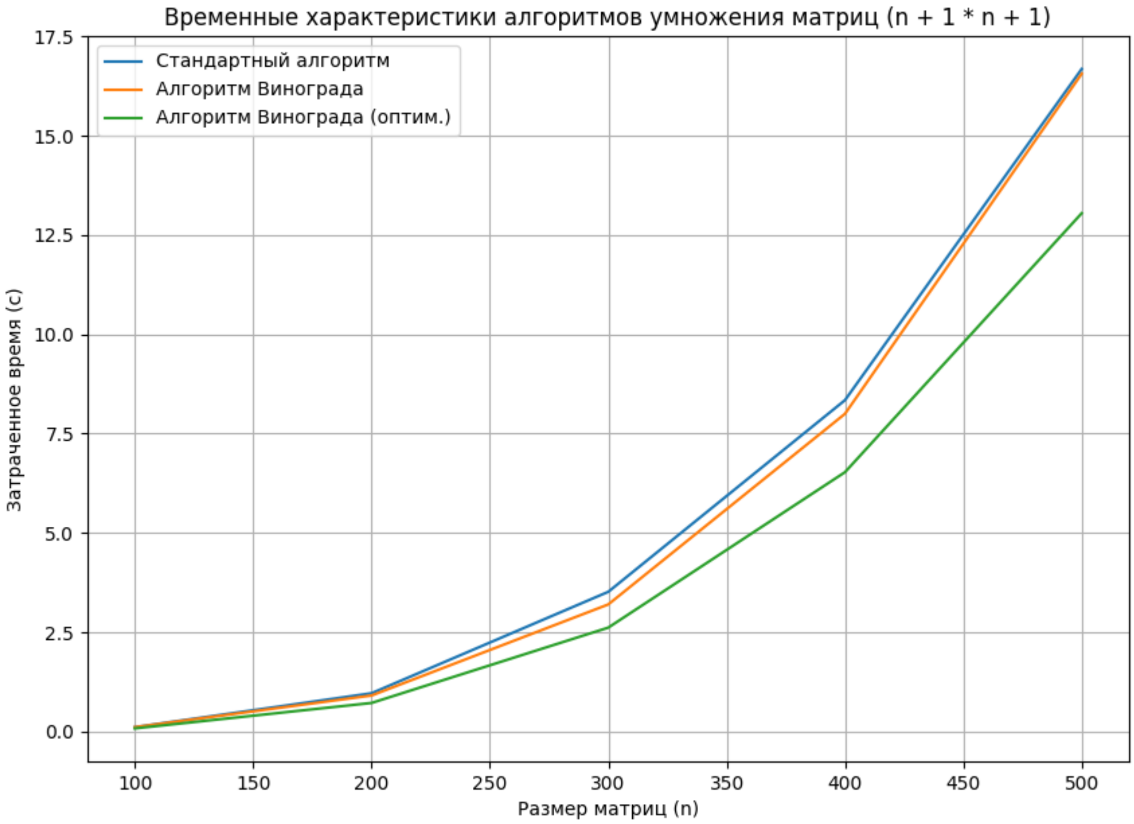

Рисунок 4.3 – Сравнение процессорного времени работы реализаций алгоритмов умножения матриц на квадратных матрицах нечетной размерности (n + 1) * (n + 1)

4.4 Оценка затрат алгоритмов по памяти

Пусть первая матрица (A) имеет N строк и M столбцов, вторая матрица (B) имеет M строк и Q столбцов, тогда затраты памяти на рассматриваемые алгоритмы умножения матриц будут следующими.

Стандартный алгоритм умножения матриц:

- входные матрицы – (N * M + M * Q) * sizeof(int);
- результирующая матрица – (N * Q) * sizeof(int);
- размерности матриц - 3 * sizeof(int).

Суммарные затраты памяти стандартного алгоритма умножения матриц: (N * M + M * Q + N * Q + 3) * sizeof(int) байт.

Алгоритм Винограда умножения матриц:

- входные матрицы – (N * M + M * Q) * sizeof(int);
- результирующая матрица – (N * Q) * sizeof(int);
- размерности матриц – 3 * sizeof(int);
- массивы MulH и MulV – (N + Q) * sizeof(int);
- вспомогательные переменные (флаг) – sizeof(int). (только для оптимизированного алгоритма Винограда)

Суммарные затраты памяти алгоритма Винограда умножения матриц: (N * M + M * Q + N * Q + N + Q + 3) * sizeof(int) байт.

Суммарные затраты памяти оптимизированного алгоритма Винограда умножения матриц: (N * M + M * Q + N * Q + N + Q + 4) * sizeof(int) байт.

Вывод

Таким образом, самой эффективной по времени по экспериментальным данным с учетом работы на всех видах матриц является реализация оптимизированного алгоритма Винограда, стандартный алгоритм и алгоритм Винограда без оптимизации практически одинаковы эффективны по времени из-за того, что неоптимизированный алгоритм Винограда содержит большое количество операций.

Самым эфективным по памяти является стандартный алгоритм умножения матриц, так как в алгоритме Винограда выделяется память под вспомогательные массивы MulH и MulV.

Заключение

В результате выполнения лабораторной работы цель достигнута, а именно изучены способы оптимизации алгоритмов на примере алгоритмов умножения матриц (стандартный и Винограда).

В ходе выполнения данной работы были решены следующие задачи:

- изучены два алгоритма умножения матриц: стандартный алгоритм, алгоритм Винограда;
- разработаны три алгоритма умножения матриц: стандартный, Винограда и Винограда с оптимизациями;
- реализованы алгоритмы умножения матриц: стандартный, Винограда и Винограда с оптимизациями;
- выполнены замеры процессорного времени работы реализаций алгоритмов алгоритмов умножения матриц: стандартный, Винограда и Винограда с оптимизациями;
- проведен сравнительный анализ трудоемкости реализаций разработанных алгоритмов умножения матриц на основе теоретических расчетов;
- выполнена оценка и сравнительный анализ затрат алгоритмов умножения матриц по памяти;
- проведен сравнительный анализ процессорного времени работы реализаций разработанных алгоритмов умножения матриц на основе экспериментальных данных.

В результате лабораторной работы можно сделать вывод, что самой эффективной по времени (на основе полученных экспериментальных данных) является реализация оптимизированного алгоритма Винограда умножения матриц,а самым эффективным по памяти является стандартный алгоритм умножения матриц.

Литература

[1] Матрицы. Виды матриц [Электронный ресурс]. Режим доступа: https://vuzlit.com/833194/matritsy_vidy_matrits (дата обращения: 20.10.2022).

[2] Произведение двух матриц: формула, решения, свойства [Электронный ресурс]. Режим доступа: https://function-x.ru/operations_with_matrices.html (дата обращения: 20.10.2022).

[3] Умножение матриц по Винограду [Электронный ресурс]. Режим доступа: http://algolib.narod.ru/Math/Matrix.html (дата обращения: 20.10.2022).

[4] Лутц Марк. Изучаем Python, том 1, 5-е изд. Пер. с англ. — СПб.: ООО "Диалектика", 2019. с. 832.

[5] time — Time access and conversions [Электронный ресурс]. Режим доступа: https://docs.python.org/3/library/time.html (дата обращения: 20.09.2021).

[6] Узнайте все о PyCharm [Электронный ресурс]. Режим доступа: https://www.jetbrains.com/ru-ru/pycharm/learn/ (дата обращения: 20.09.2022).

[7] Windows 10 [Электронный ресурс]. Режим доступа: https://learn.microsoft.com/ru-ru/windows/ (дата обращения: 20.09.2022).