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 Зайцева А.А.

Группа ИУ7-52Б

Преподаватели Волкова Л.Л.

Москва — 2021 г.

Оглавление

Введение

Целью данной лабораторной работы является изучение способов оптимизации алгоритмов на примере алгоритмов умножения матриц.

Основное значение термин «матрица» 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 широко применяются в математике для компактной записи систем линейных алгебраических или дифференциальных уравнений. В этом случае, количество строк матрицы соответствует числу уравнений, а количество столбцов — количеству неизвестных. В результате решение систем линейных уравнений сводится к операциям над матрицами, в том числе - умножению.

Умножение матриц A и B — это операция вычисления матрицы C, элементы которой равны сумме произведений элементов в соответствующей строке первого множителя и столбце второго [2].

В рамках выполнения работы необходимо решить следующие задачи:

1) изучить алгоритмы умножения матриц;
2) разработать и реализовать 3 алгоритма умножения матриц: стандартный, Винограда и Винограда с оптимизациями;
3) оценить трудоемкость реализаций алгоритмов;
4) провести сравнительный анализ процессорного времени выполнения реализаций алгоритмов.

1 Аналитическая часть

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:

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

в которой каждый элемент вычисляется по формуле 1:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}, \ (i = 1, 2, \dots l; j = 1, 2, \dots n) \quad (1)$$

Стандартный алгоритм действует именно по этой формуле.

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 (1.1):

$$V \cdot W = (v_1+w_2)(v_2+w_1)+(v_3+w_4)(v_4+w_3)-v_1v_2-v_3v_4-w_1w_2-w_3w_4. \quad (1.1)$$

Кажется, что второе выражение задает больше работы, чем первое: вместо четырех умножений мы насчитываем их шесть, а вместо трех сложений - 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. 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 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 идеи, лежащие в основе рассматриваемых алгоритмов умножения матриц - стандартного и алгоритма Винограда.

2 Конструкторская часть

2.1 Разработка алгоритмов

На рисунках 2.1 - 2.3 приведены схемы алгоритмов, соотвтетственно, стандартного умножения матриц, умножения матриц по Винограду и оптимизированного умножения матриц по Винограду.

На схеме на рисунке 2.2 видно, что для алгоритма Винограда худшим случаем являются матрицы с нечётным общим размером, а лучшим - с чётным, так как отпадает необходимость в последнем цикле.

Алгоритм Винограда можно оптимизировать несколькими способами.

1) Заменить сравнение k $< \frac{n}{2}$ и инкремент k на сравнение k $<$ n и прибавление 2 к k в заголовках циклов. При этом в телах циклов отпадает необходимость в умножении k на 2, а из n нужно заранее вычесть единицу, в последнем цикле изменить условие входа в него и убрать вычитание 1 из n.

2) Заменить все выражения вида a = a + c на a += c.

3) Заменить в mulh += на -=, и тогда в третьем цикле c[i][j] = mulh[i] - mulv[j].

Рисунок 2.1 – Стандартный алгоритм умножения матриц

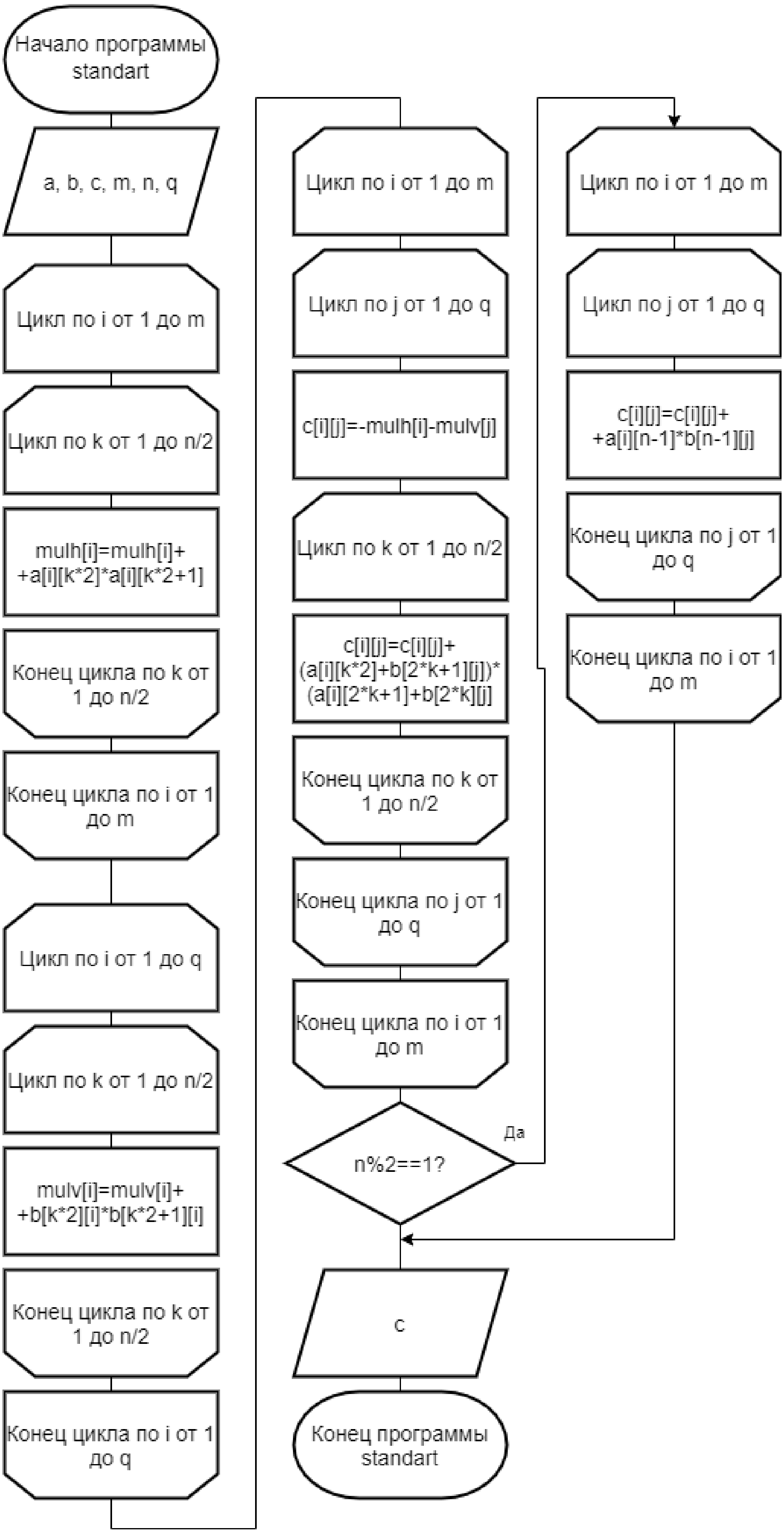

Рисунок 2.2 – Алгоритм умножения матриц по Винограду

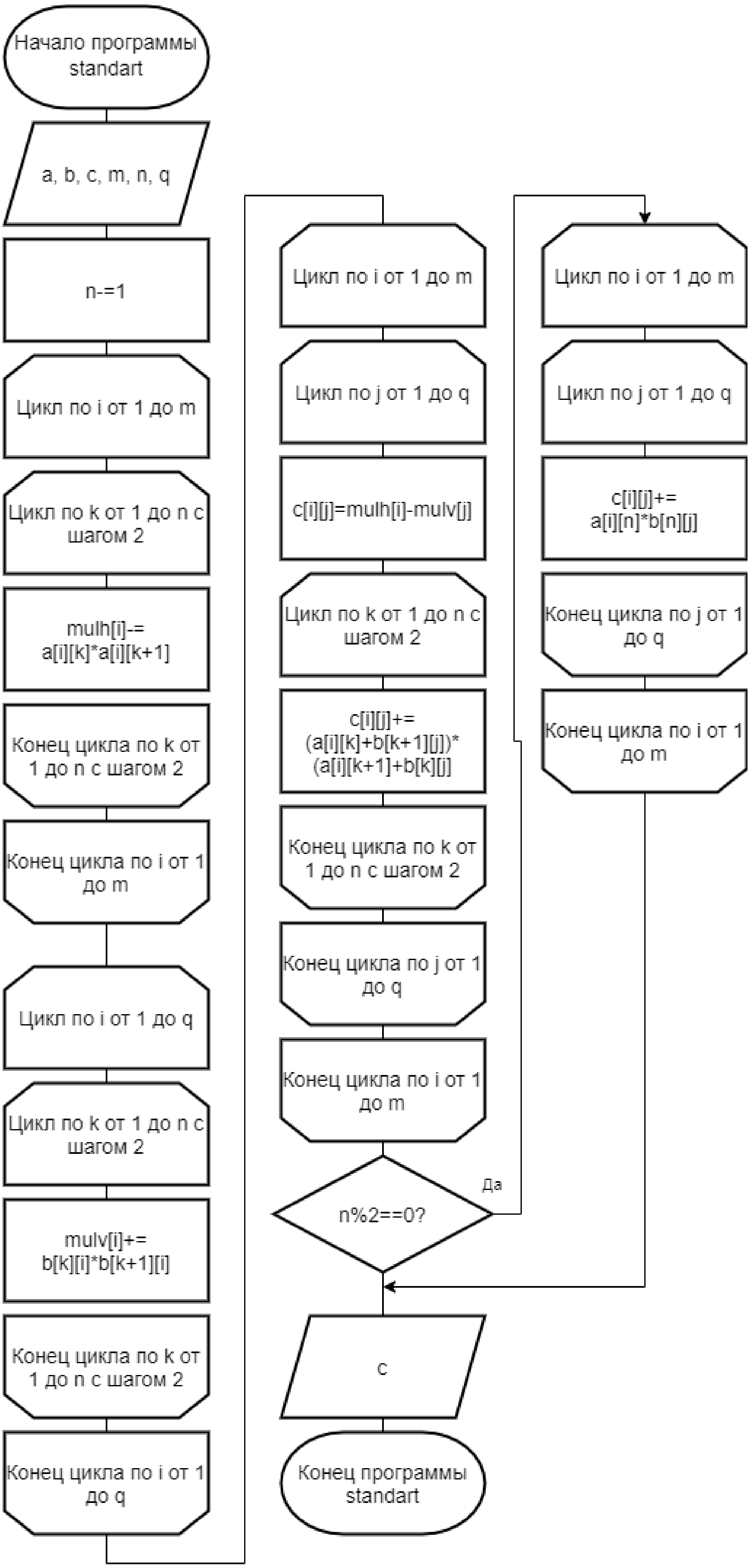

Рисунок 2.3 – Оптимизированный алгоритм умножения матриц по Винограду

2.2 Оценка трудоемкости

Произведем теоретическую оценку трудоемкости алгоритмов умножения матриц.

1. Стандартный алгоритм:

$$f = 2+m(2+2+q(2+2+n(2+8+1+1+1*2))) = 14mnq+4mq+4m+2$$

2. Алгоритм Винограда, складывающийся из 4 циклов:

$$f1 = 2 + m(2 + 4 + \frac{n}{2}(4 + 6 + 1 + 2 + 3 * 2)) = \frac{19}{2}mn + 6m + 2$$

$$f2 = 2 + q(2 + 4 + \frac{n}{2}(4 + 6 + 1 + 2 + 3 * 2)) = \frac{19}{2}qn + 6q + 2$$

$$f3 = 2+m(2+2+q(2+7+4+\frac{n}{2}(4+1+12+5+5*2))) = 16mnq+13mq+4m+2$$

$$f4 = 3 + \left[\begin{array}{c} \text{0, л.с. (n четное)} \\ \text{2+m(2+2+q(2+8+1+3+1*2))=16mq+4m+2, х.с.} \end{array}\right.$$

Итоговая трудоемкость - сумма трудоемкостей каждого цикла:

$$f = 16mnq+13mq+\frac{19}{2}mn+\frac{19}{2}qn+10m+6q+9+\left[\begin{array}{l} \text{0, л.с. (n четное)} \\ \text{16mq+4m+2, х.с.} \end{array}\right.$$

Трудоемкость выше по сравнению со стандартной из-за отсутствия оптимизаций.

3. Оптимизированный алгоритм Винограда расчитывается аналогично, не упуская при этом вычитание из n единицы в самом начале:

$$f1 = 1 + 2 + m(2 + 2 + \frac{n}{2}(2 + 5 + 2 + 1 * 2)) = \frac{11}{2}mn + 4m + 3$$

$$f2 = 2 + q(2 + 2 + \frac{n}{2}(2 + 5 + 2 + 1 * 2)) = \frac{11}{2}qn + 4q + 2$$

$$f3 = 2+m(2+2+q(2+6+2+\frac{n}{2}(2+1+10+4+1*2))) = \frac{19}{2}mnq+10mq+4m+2$$

$$f4 = 3 + \left[\begin{array}{c} \text{0, л.с. (n четное)} \\ \text{2+m(2+2+q(2+6+1+1*2))=13mq+4m+2, х.с.} \end{array} \right.$$

$$f = \frac{19}{2}mnq+10mq+\frac{11}{2}mn+\frac{11}{2}qn+8m+4q+10+ \left[\begin{array}{l} \text{0, л.с. (n четное)} \\ \text{13mq+4m+2, х.с.} \end{array} \right.$$

Трудоемкость меньше по сравнению со стандартным алгоритмом за счет меньшей доли умножений.

Вывод

Были разработаны схемы алгоритмов, позволяющих с помощью различных подходов находить произведение матриц, а также была дана оценка их трудоемкости.

3 Технологическая часть

3.1 Требования к ПО

На вход программе подаются две матрицы, а на выходе должно быть получено искомое произведение матриц, посчитанное с помощью каждого реализованного алгоритма: стандартный, Винограда и оптимизированный Винограда. Также необходимо вывести затраченное каждым алгоритмом процессорное время.

3.2 Выбор средств реализации

В качестве языка программирования для реализации данной лабораторной работы был выбран язык Python [4]. Он позволяет быстро реализовывать различные алгоритмы без выделения большого времени на проектирование сруктуры программы и выбор типов данных.

Кроме того, в Python есть библиотека time, которая предоставляет функцию process_time для замера процессорного времени [5].

В качестве среды разработки выбран PyCharm. Он является кроссплатформенным, а также предоставляет удобный и функциональнаый отладчик и средства для рефакторинга кода, что позволяет быстро находить и исправлять ошибки [6].

3.3 Листинги кода

В листингах 3.1 - 3.3 представлены реализации рассматриваемых алгоритмов.

Листинг 3.1 – Стандартный алгоритм умножения матриц

```
def standart_mult(a, b, c, m, n, q):
  for i in range(m):
    for j in range(q):
      for k in range(n):
        c[i][j] = c[i][j] + a[i][k] * b[k][j]
  return c
```

Листинг 3.2 – Алгоритм умножения матриц Винограда

```
def vinograd_usual_mult(a, b, c, m, n, q):
  mulh = [0 for i in range(m)]
  for i in range(m):
    for k in range(n//2):
      mulh[i] = mulh[i] + a[i][k * 2] * a[i][k * 2 + 1]

  mulv = [0 for i in range(q)]
  for i in range(q):
    for k in range(n//2):
      mulv[i] = mulv[i] + b[k * 2][i] * b[k * 2 + 1][i]

  for i in range(m):
    for j in range(q):
      c[i][j] = -mulh[i] - mulv[j]
      for k in range(n//2):
        c[i][j] = c[i][j] + (a[i][k * 2] + b[2 * k + 1][j]) * (a[i
          ][2 * k + 1] + b[2 * k][j])

  if n % 2:
    for i in range(m):
      for j in range(q):
        c[i][j] = c[i][j] + a[i][n - 1] * b[n - 1][j]

  return c
```

Листинг 3.3 – Оптимизированный алгоритм умножения матриц Винограда

```
def vinograd_optimized_mult(a, b, c, m, n, q):
    n -= 1
    mulh = [0 for i in range(m)]
    for i in range(m):
        for k in range(0, n, 2):
            mulh[i] -= a[i][k] * a[i][k + 1]

    mulv = [0 for i in range(q)]
    for i in range(q):
        for k in range(0, n, 2):
            mulv[i] += b[k][i] * b[k + 1][i]

    for i in range(m):
        for j in range(q):
            c[i][j] = mulh[i] - mulv[j]
            for k in range(0, n, 2):
                c[i][j] += ((a[i][k] + b[k + 1][j]) * (a[i][k + 1] + b[k][j]))

    if not (n % 2):
        for i in range(m):
            for j in range(q):
                c[i][j] += a[i][n] * b[n][j]

    return c
```

3.4 Тестирование

В таблице 3.1 приведены функциональные тесты для алгоритмов умножения матриц: стандартного, Винограда и оптимизированного Винограда. Все тесты пройдены успешно каждым алгоритмом.

Матрица A	Матрица B	Ожидаемый результат C
$\begin{pmatrix} * \end{pmatrix}$	$\begin{pmatrix} * \end{pmatrix}$	$\begin{pmatrix} * \end{pmatrix}$
$\begin{pmatrix} 2 \end{pmatrix}$	$\begin{pmatrix} 2 \end{pmatrix}$	$\begin{pmatrix} 4 \end{pmatrix}$
$\begin{pmatrix} 1 & 1 \\ 1 & -1 \\ 2 & 2 \end{pmatrix}$	$\begin{pmatrix} 0 & -1 & 1 & 2 \\ 0 & 1 & 1 & 3 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 2 & 5 \\ 0 & -2 & 0 & -1 \\ 0 & 0 & 4 & 10 \end{pmatrix}$
$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 2 & 3 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 2 & 3 & 4 \\ 2 & 3 & 4 \\ 2 & 3 & 4 \end{pmatrix}$
$\begin{pmatrix} 1 & 1 \\ 2 & 2 \end{pmatrix}$	$\begin{pmatrix} 1 \end{pmatrix}$	Не могут быть перемножены

Таблица 3.1 – Тестирование функций

Вывод

Был производен выбор средств реализации, приведены требования к ПО, реализованы и протестированы алгоритмы умножения матриц.

4 Исследовательская часть

4.1 Пример работы

На рисунке 4.1 приведен пример работы программы.

```
Введите размерность первой матрицы.
m: 2
n: 2
Введите первую матрицу построчно, через пробелы
1 2
3 4

Введите размерность второй матрицы.
n: 2
q: 3
Введите вторую матрицу построчно, через пробелы
1 1 1
2 2 2



Стандартный алгоритм
Ответ:
[5, 5, 5]
[11, 11, 11]
Время: 0.0

Алгоритм Винограда
Ответ:
[5, 5, 5]
[11, 11, 11]
Время: 0.0

Оптимизированный алгоритм Винограда
Ответ:
[5, 5, 5]
[11, 11, 11]
Время: 0.0
```

Рисунок 4.1 – Пример работы программы

4.2 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование:

- операционная система: Windows 10;
- оперативная память: 16 Гб;
- процессор: Intel® Core™ i5-8259U.

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

4.3 Время выполнения реализаций алгоритмов

Все реализации алгоритмов сравнивались на случайно сгенерированных квадратных матрицах размерностями n*n, где n изменялось от 100 до 1000 с шагом 100 (в качестве лучшего случая, четная общеая размерность) и от 101 до 1001 с шагом 100 (в качестве худшего случая). Так как замеры времени имеют некоторую погрешность, они для каждой размерности и каждой реализации алгоритма производились 10 раз, а затем вычислялось среднее время работы реализации с матрицами.

На рисунке 4.2 приведены результаты сравнения времени работы всех реализаций на лучшем случае (четная общая размерность), а на рисунке 4.3 - на худшем (нечетная). Как видно на графиках, теоретические расчеты подтверждаются: все алгоритмы кубически зависят от размерностей матриц, при этом алгоритм Винограда работает дольше всех, а оптимизированный алгоритм Винограда - меньше всех. При нечетной общей размерности алгоритмы Винограда работают дольше по сравнению с тем же алгоритмом при четной общей размерности, однако все равно требуют меньше процессорного времени, чем станадртный алгоритм.

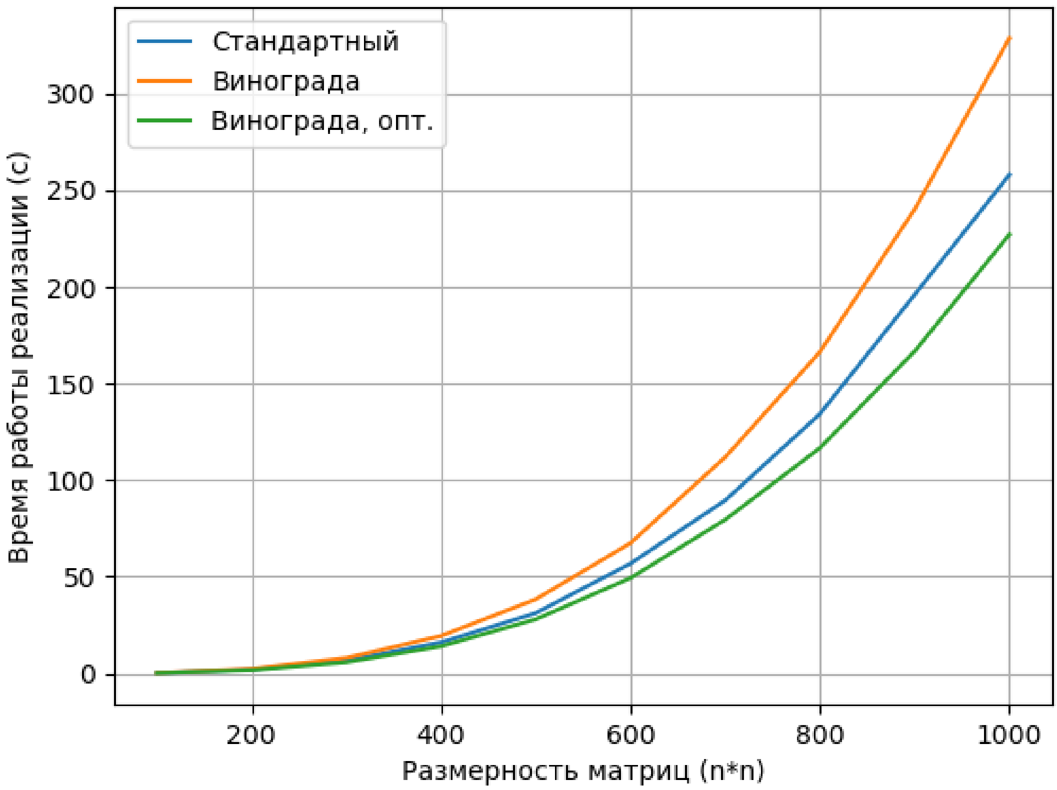

Рисунок 4.2 – Сравнение времени работы реализаций алгоритмов (лучший случай)

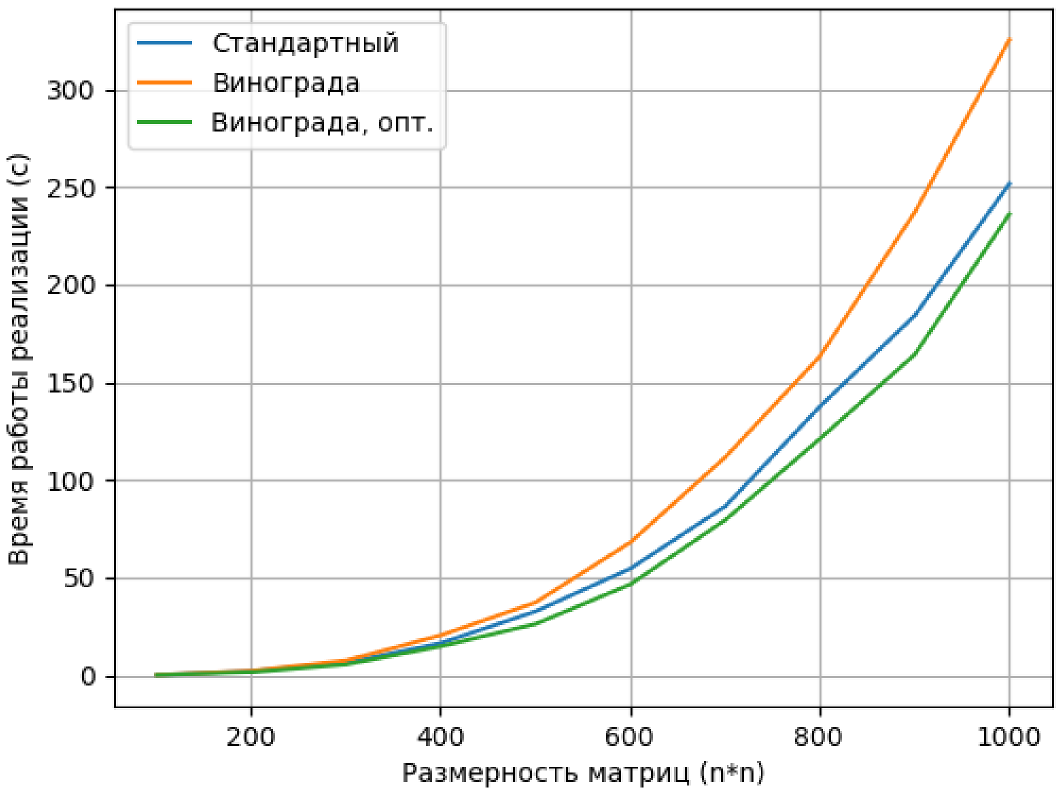

Рисунок 4.3 – Сравнение времени работы реализаций алгоритмов (худший случай)

Вывод

Были подтверждены теоретические расчеты: алгоритм Винограда работает дольше стандартного, а оптимизированный алгоритм Винограда - меньше.

Таким образом, несмотря на сложность алгоритма Винограда по сравнению со стандартным, меньшая доля умножений в нем при применении оптимизаций позволяет получить меньшую трудоемкость

Заключение

В результате выполнения лабораторной работы при исследовании алгоритмов умножения матриц были изучены способы оптимизации алгоритмов.

В ходе выполнения лабораторной работы были выполнены следующие задачи:

1) изучены алгоритмы умножения матриц;
2) разработаны и реализованы 3 алгоритма умножения матриц: стандартный, Винограда и Винограда с оптимизациями;
3) дана теоретическая оценка трудоемкости реализаций алгоритмов;
4) проведен сравнительный анализ процессорного времени выполнения реализаций алгоритмов.

Литература

[1] Матрица, ее история и применение [Электронный ресурс]. Режим доступа: https://urok.1sept.ru/articles/637896 (дата обращения: 20.09.2021).

[2] Перемножение матриц [Электронный ресурс]. Режим доступа: https://dic.academic.ru/dic.nsf/ (дата обращения: 20.09.2021).

[3] Умножение матриц по Винограду [Электронный ресурс]. Режим доступа: http://algolib.narod.ru/Math/Matrix.html (дата обращения: 20.09.2021).

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

[5] time — Time access and conversions [Электронный ресурс]. Режим доступа: https://docs.python.org/3/library/time.html (дата обращения: 05.09.2021).

[6] Python и Pycharm [Электронный ресурс]. Режим доступа: https://py-charm.blogspot.com/2017/09/pycharm.html (дата обращения: 05.09.2021).