



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.

# Оглавление

<b>Введение</b>	<b>3</b>
<b>1 Аналитическая часть</b>	<b>5</b>
1.1 Стандартный алгоритм умножения матриц . . . . .	5
1.2 Алгоритм Винограда умножения матриц . . . . .	5
<b>2 Конструкторская часть</b>	<b>7</b>
2.1 Стандартный алгоритм умножения матриц . . . . .	7
2.2 Алгоритм Винограда умножения матриц . . . . .	9
2.3 Оптимизированный алгоритм Винограда умножения матриц	11
2.4 Оценка трудоемкости алгоритмов умножения матриц . . . .	13
2.4.1 Стандартный алгоритм умножения матриц . . . . .	14
2.4.2 Алгоритм Винограда умножения матриц . . . . .	14
2.4.3 Оптимизированный алгоритм Винограда умножения матриц . . . . .	15
<b>3 Технологическая часть</b>	<b>17</b>
3.1 Требования к программному обеспечению . . . . .	17
3.2 Выбор средств реализации . . . . .	17
3.3 Реализация алгоритмов . . . . .	18
3.4 Тестирование . . . . .	20
<b>4 Исследовательская часть</b>	<b>23</b>
4.1 Пример работы программного обеспечения . . . . .	23
4.2 Технические характеристики . . . . .	25
4.3 Время выполнения реализаций алгоритмов . . . . .	25
4.4 Оценка затрат алгоритмов по памяти . . . . .	27
<b>Заключение</b>	<b>29</b>
<b>Список использованных источников</b>	<b>30</b>

# 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;
- 4) выполнение замеров процессорного времени работы реализаций алгоритмов умножения матриц;

- 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: стандартного, Винограда.

## 1.1 Стандартный алгоритм умножения матриц

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

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

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:

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

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 размерности матриц  $M$  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 и вычисления трудоемкости данных алгоритмов.

### 2.1 Стандартный алгоритм умножения матриц

На рисунке 2.1 приведена схема стандартного алгоритма умножения матриц.

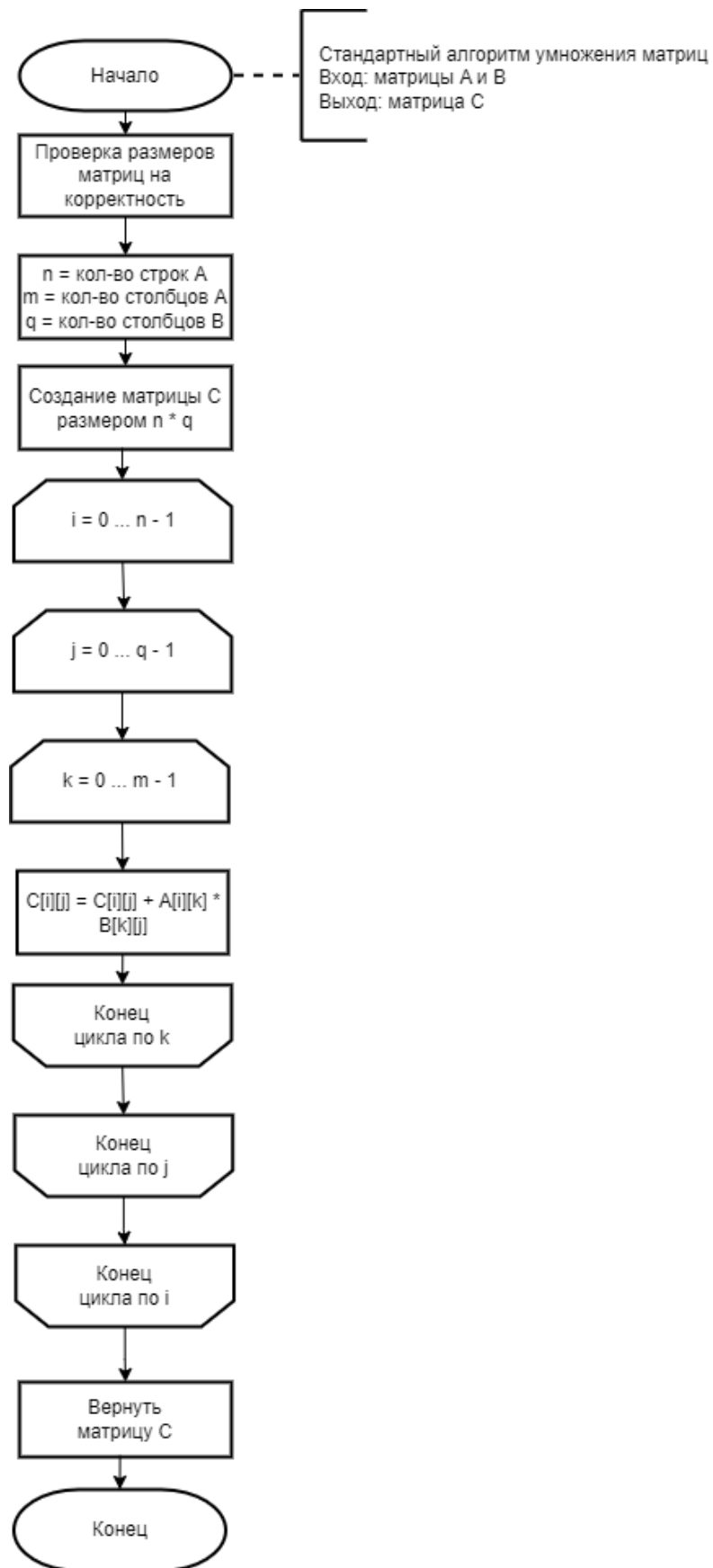


Рисунок 2.1 – Схема стандартного алгоритма умножения матриц



## 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 (длина перемножаемых векторов), а лучшим — с чётной, так как последний цикл в этом случае не задействуется.

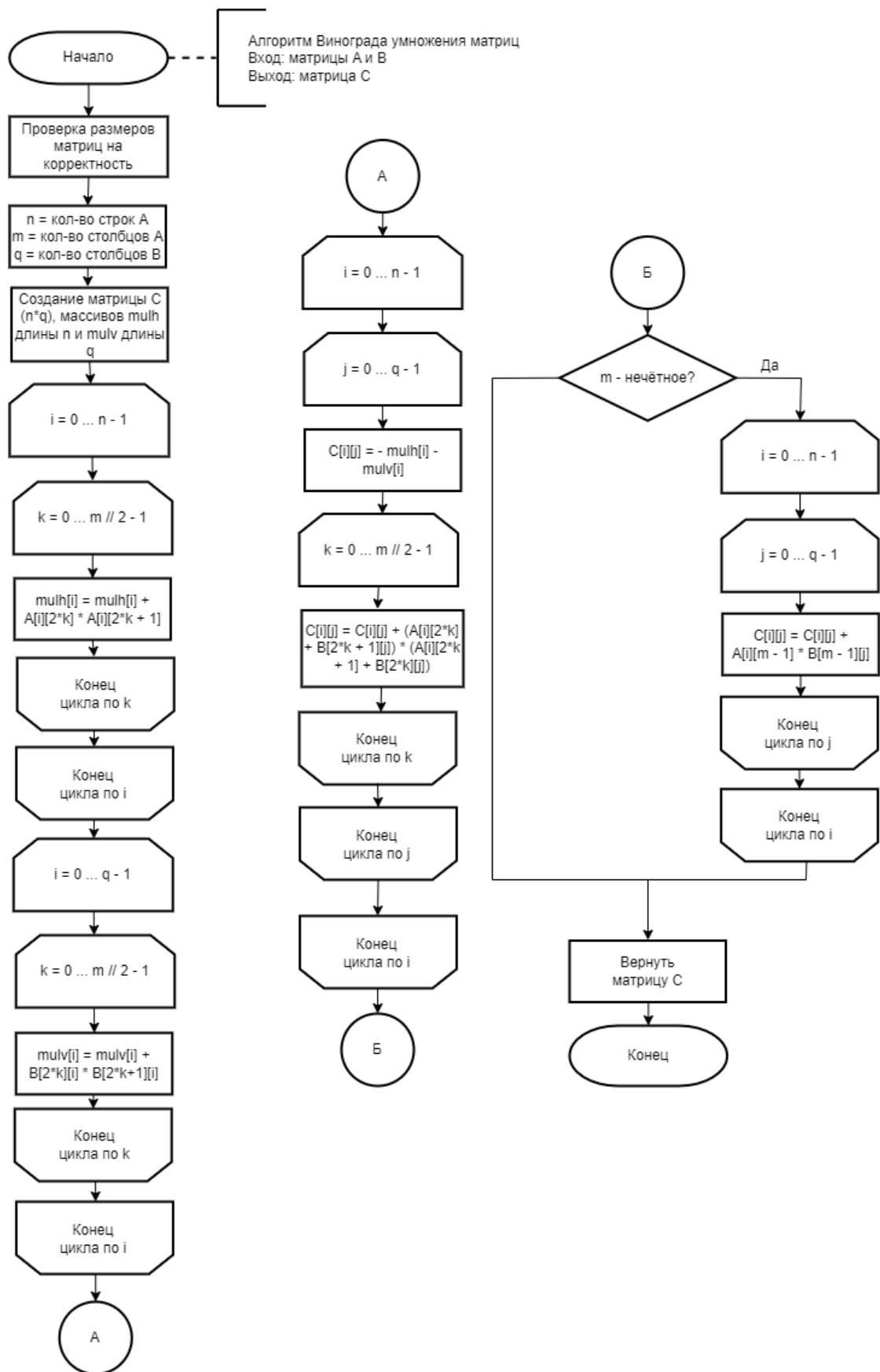


Рисунок 2.2 – Схема алгоритма Винограда умножения матриц

## 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 (длина перемножаемых векторов) входных матриц проверить до третьего цикла и установить соответствующий флаг, если флаг содержит значение истина, то выполняем дополнительные операции в третьем цикле (предвычисляются некоторые слагаемые для алгоритма).

На рисунке 2.3 приведена схема алгоритма Винограда умножения матриц с учетом перечисленных выше оптимизаций.

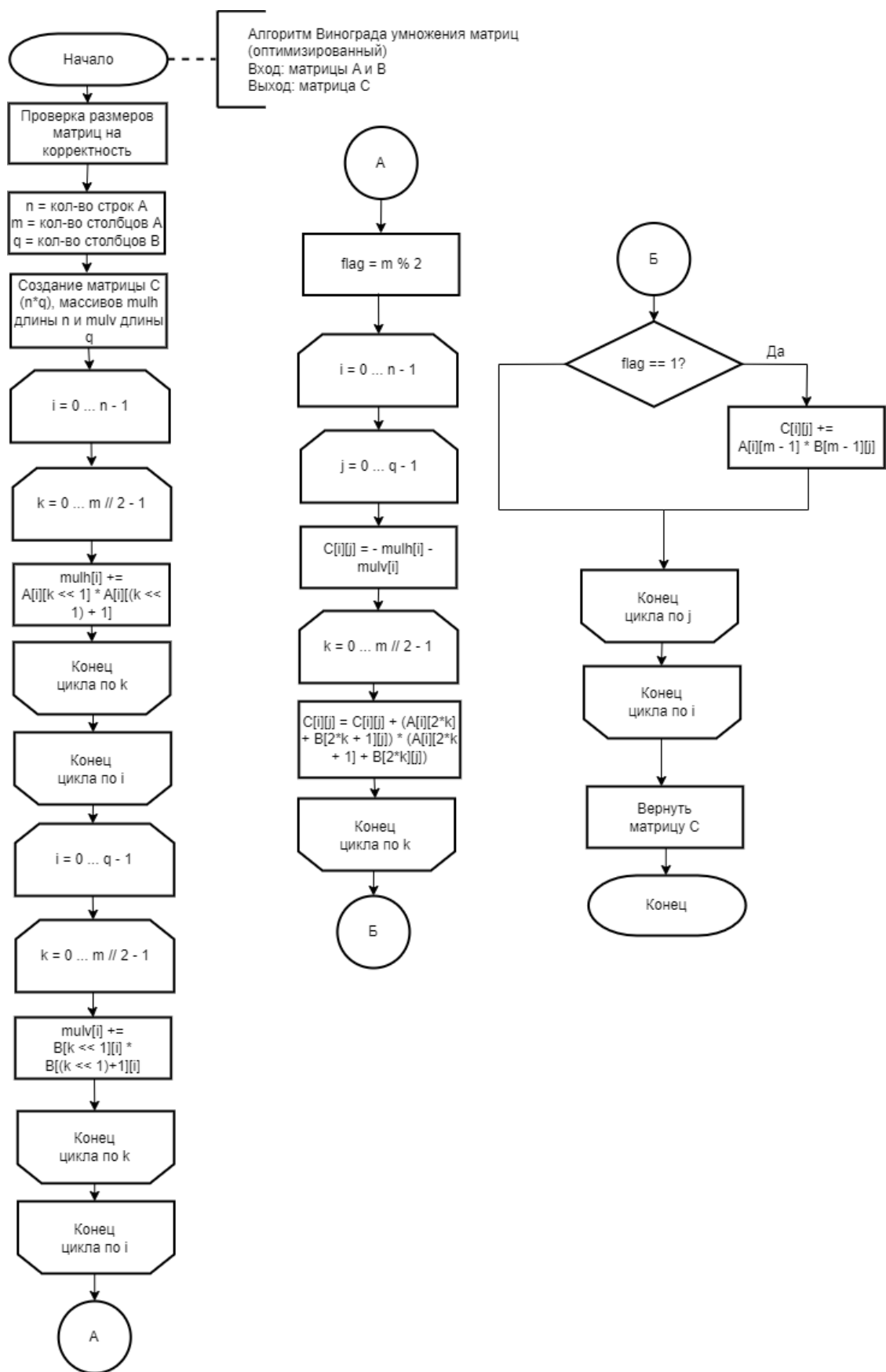


Рисунок 2.3 – Оптимизированный алгоритм Винограда умножения матриц

## 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:  $F_{\text{цикла}} = F_{\text{иниц.}} + F_{\text{сравн.}} + M * (F_{\text{тела}} + F_{\text{инк.}} + F_{\text{сравн.}})$ , где  $F_{\text{иниц.}}$ ,  $F_{\text{сравн.}}$ ,  $F_{\text{тела}}$ ,  $F_{\text{инк.}}$  — трудоемкости инициализации, проверки условия цикла, тела цикла и инкрементирования соответственно, а  $M$  — количество итераций.
- Трудоемкость условного оператора:

$$F_{if} = F_{cmp} + \begin{cases} \min(f1, f2) - \text{лучший случай} \\ \max(f1, f2) - \text{худший случай} \end{cases}, \quad (2.1)$$

где  $F_{cmp}$ ,  $f1$ ,  $f2$  — трудоемкости проверки условия, первого блока и второго блока, соответственно.

- Трудоемкость инициализации массива/матрицы —  $N$ , где  $N$  — число элементов в массиве/матрице.

Обозначим во всех последующих вычислениях размерности входных матриц ( $A[N \times M]$ ,  $B[M \times Q]$ ) 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 следующих составляющих.

- Трудоёмкость внешнего цикла (по  $i \in [1..N]$ ):  $f_1 = 2 + N \cdot (2 + f_{body})$
- Трудоёмкость внутреннего цикла (по  $j \in [1..Q]$ ):  $f_2 = 2 + Q \cdot (2 + f_{body})$
- Трудоёмкость внутреннего цикла (по  $k \in [1..M]$ ):  $f_3 = 2 + 14M$

Суммарная трудоёмкость стандартного алгоритма умножения матриц:

$$f_s = 2 + N \cdot (4 + Q \cdot (4 + 14M)) \approx 14NMQ \quad (2.2)$$

### 2.4.2 Алгоритм Винограда умножения матриц

Трудоёмкость алгоритма Винограда умножения матриц состоит из следующих составляющих.

- Трудоёмкость создания и инициализации массивов  $MulH$  и  $MulV$ :

$$f_{init} = N + Q \quad (2.3)$$

- Трудоёмкость заполнения массива  $MulH$ :

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

- Трудоёмкость заполнения массива  $MulV$ :

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

- Трудоёмкость цикла заполнения для чётных размеров:

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

- Трудоемкость цикла, для дополнения результата умножения суммой последних нечётных строки и столбца, если общая размерность (длина перемножаемых векторов) входных матриц нечётная:

$$f_{for4} = 3 + \begin{cases} 0, & \text{л.с.}, \\ 2 + M \cdot (4 + 16N), & \text{х.с.} \end{cases} \quad (2.7)$$

Трудоемкость в худшем случае, при нечётной размерности матриц  $M$ :

$$f_v = f_{MulH} + f_{MulV} + f_{for3} + f_{for4} \approx 14.5 \cdot NMQ \quad (2.8)$$

Трудоемкость в лучшем случае, при чётной размерности матриц  $M$ :

$$f_v = f_{MulH} + f_{MulV} + f_{for3} + f_{for4} \approx 14.5 \cdot NMQ \quad (2.9)$$

### 2.4.3 Оптимизированный алгоритм Винограда умножения матриц

Трудоёмкость оптимизированного алгоритма Винограда умножения матриц состоит из следующих составляющих.

- Трудоемкость создания и инициализации массивов  $MulH$  и  $MulV$ :

$$f_{init} = N + Q \quad (2.10)$$

- Трудоемкость заполнения массива  $MulH$ :

$$f_{MulH} = 2 + N \cdot \left(3 + \frac{M}{2} \cdot 12\right) \quad (2.11)$$

- Трудоемкость заполнения массива  $MulV$ :

$$f_{MulV} = 2 + Q \cdot \left(3 + \frac{M}{2} \cdot 12\right) \quad (2.12)$$

- Трудоемкость цикла заполнения для чётных размеров:

$$f_{for3} = 2 + N \cdot (4 + Q \cdot (12 + \frac{M}{2} \cdot 24)) \quad (2.13)$$

- Трудоемкость условия, для дополнения умножения суммой последних нечётных строки и столбца, если размерность входных матриц  $M$  — нечётная:

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

Трудоемкость в худшем случае, при нечётной размерности матриц  $M$ :

$$f_{ov} = f_{MulH} + f_{MulV} + f_{for3} + f_{if} \approx 12NMQ \quad (2.15)$$

Трудоемкость в лучшем случае, при чётной размерности матриц  $M$ :

$$f_{ov} = f_{MulH} + f_{MulV} + f_{for3} + f_{if} \approx 12NMQ \quad (2.16)$$

## Вывод

В данном разделе были разработаны схемы алгоритмов умножения матриц: стандартного, Винограда, оптимизированного 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 столбцов 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: стандартного, Винограда, оптимизированного Винограда.

### 3.2 Выбор средств реализации

В качестве языка программирования для реализации данной лабораторной работы был выбран язык программирования Python [4]. В данном языке программирования есть необходимая для замеров процессорного времени библиотека.

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 установки необходимых библиотек при необходимости.

### 3.3 Реализация алгоритмов

В листингах 3.1 – 3.3 представлены реализации различных алгоритмов умножения матриц: стандартного, Винограда, оптимизированного Винограда.

Листинг 3.1 – Реализация стандартного алгоритма умножения матриц

```
1 def standart_mult_matr(matr1, matr2):
2     if len(matr1[0]) != len(matr2):
3         print("Incorrect size of matrices!")
4         return -1
5
6     n = len(matr1)
7     m = len(matr1[0])
8     q = len(matr2[0])
9
10    res = [[0] * q for i in range(n)]
11
12    for i in range(n):
13        for j in range(q):
14            for k in range(m):
15                res[i][j] = res[i][j] + matr1[i][k] * matr2[k][j]
16
17    return res
```

Листинг 3.2 – Реализация алгоритма Винограда умножения матриц

```
1 def vinograd_mult_matr(matr1, matr2):
2     if len(matr1[0]) != len(matr2):
3         print("Incorrect size of matrices!")
4         return -1
5
6     n = len(matr1)
7     m = len(matr1[0])
8     q = len(matr2[0])
9
10    res = [[0] * q for i in range(n)]
11    # vector of rows matrix A
12    mulh = [0] * n
13    for i in range(n):
14        for k in range(m // 2):
```

```

15         mulh[i] = mulh[i] + matr1[i][2 * k] * matr1[i][2 * k +
16             1]
17     # vector of columns matrix B
18     mulv = [0] * q
19     for i in range(q):
20         for k in range(m // 2):
21             mulv[i] = mulv[i] + matr2[2 * k][i] * matr2[2 * k +
22                 1][i]
23     # filling matrix C
24     for i in range(n):
25         for j in range(q):
26             res[i][j] = - mulh[i] - mulv[j]
27             for k in range(m // 2):
28                 res[i][j] = res[i][j] + (matr1[i][2 * k] + matr2[2
29                     * k + 1][j]) * \
30                     (matr1[i][2 * k + 1] + matr2[2 * k][j
31                         ])
32     if m % 2 == 1:
33         for i in range(n):
34             for j in range(q):
35                 res[i][j] = res[i][j] + matr1[i][m - 1] * matr2[m
36                     - 1][j]
37     return res

```

Листинг 3.3 – Реализация оптимизированного алгоритма Винограда умножения матриц

```

1 def vinograd_optimized_mult_matrix(matr1, matr2):
2     if len(matr1[0]) != len(matr2):
3         print("Incorrect size of matrices!")
4         return -1
5
6     n = len(matr1)
7     m = len(matr1[0])
8     q = len(matr2[0])
9
10    res = [[0] * q for i in range(n)]
11

```

```

12  # vector of rows matrix A
13  mulh = [0] * n
14  for i in range(n):
15      for k in range(m // 2):
16          mulh[i] += matr1[i][k << 1] * matr1[i][(k << 1) + 1]
17
18  # vector of columns matrix B
19  mulv = [0] * q
20  for i in range(q):
21      for k in range(m // 2):
22          mulv[i] += matr2[k << 1][i] * matr2[(k << 1) + 1][i]
23
24  flag = m % 2
25
26  # filling matrix C
27  for i in range(n):
28      for j in range(q):
29          res[i][j] = - mulh[i] - mulv[j]
30          for k in range(m // 2):
31              res[i][j] += (matr1[i][k << 1] + matr2[(k << 1) +
32                          1][j]) * \
33                          (matr1[i][(k << 1) + 1] + matr2[k <<
34                          1][j])
35          if flag == 1:
36              res[i][j] += matr1[i][m - 1] * matr2[m - 1][j]
37  return res

```

## 3.4 Тестирование

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

Таблица 3.1 – Тестирование реализаций алгоритмов умножения матриц

Матрица A	Матрица B	Ожидаемый результат
$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 3 \\ 6 \\ 9 \end{pmatrix}$
$\begin{pmatrix} 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$
$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix}$
(7)	(7)	(49)
$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 1 & 2 & 3 \end{pmatrix}$	$\begin{pmatrix} 6 & 7 & 9 \\ -1 & -3 & 4 \\ 3 & 7 & 9 \\ 17 & -5 & 6 \end{pmatrix}$	$\begin{pmatrix} 81 & 2 & 68 \\ 181 & 26 & 180 \\ 110 & 59 & 121 \end{pmatrix}$
$\begin{pmatrix} 1 \\ 2 \end{pmatrix}$	(0)	$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$
$\begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$	$\begin{pmatrix} 4 & 5 \\ 6 & 7 \end{pmatrix}$	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 —  $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 1 & 2 & 3 \end{bmatrix}$ , вторая матрица —  $\begin{bmatrix} 6 & 7 & 9 \\ -1 & -3 & 4 \\ 3 & 7 & 9 \\ 17 & -5 & 6 \end{bmatrix}$ . 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 выполнялись замеры процессорного времени представлены ниже.

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

Во время проведения замеров времени 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 умножения матриц (стандартный, Винограда, оптимизированный Винограда) 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 квадратных 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 по 10 раз для всех трех реализованных алгоритмов умножения матриц. В качестве результата бралось среднее время работы реализации алгоритма на каждой линейной размерности матрицы.

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

На графике видно, что реализация оптимизированного алгоритма Винограда является самой эффективной по времени среди реализованных ал-

горитмов, на втором месте по эффективности — реализация алгоритма Винограда без оптимизации, самая неэффективная по времени — реализация стандартного алгоритма умножения матриц, но она практически не уступает по времени реализации алгоритма Винограда без оптимизации.

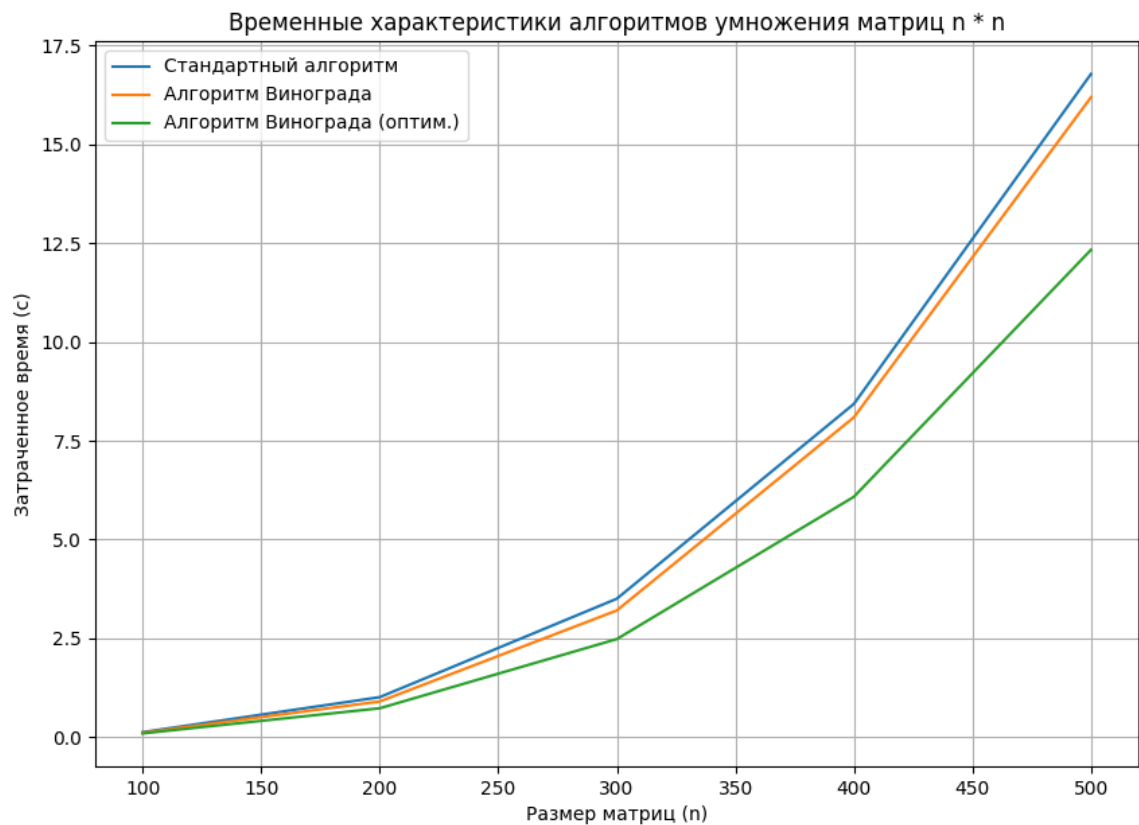


Рисунок 4.2 – Сравнение процессорного времени работы реализаций алгоритмов умножения матриц на квадратных матрицах четной размерности ( $n \times n$ )

На рисунке 4.3 представлено сравнение процессорного времени работы реализаций алгоритмов умножения матриц на матрицах нечетной размерности  $(n + 1) \times (n + 1)$ . На графике видно, что реализация оптимизированного алгоритма Винограда является самой эффективной по времени среди реализованных алгоритмов (несмотря на то, что это худший случай для данного алгоритма), на втором месте по эффективности — реализация алгоритма Винограда без оптимизации, самая неэффективная по времени — реализация стандартного алгоритма умножения матриц, но она практически не уступает по времени реализации алгоритма Винограда без оптимизации.

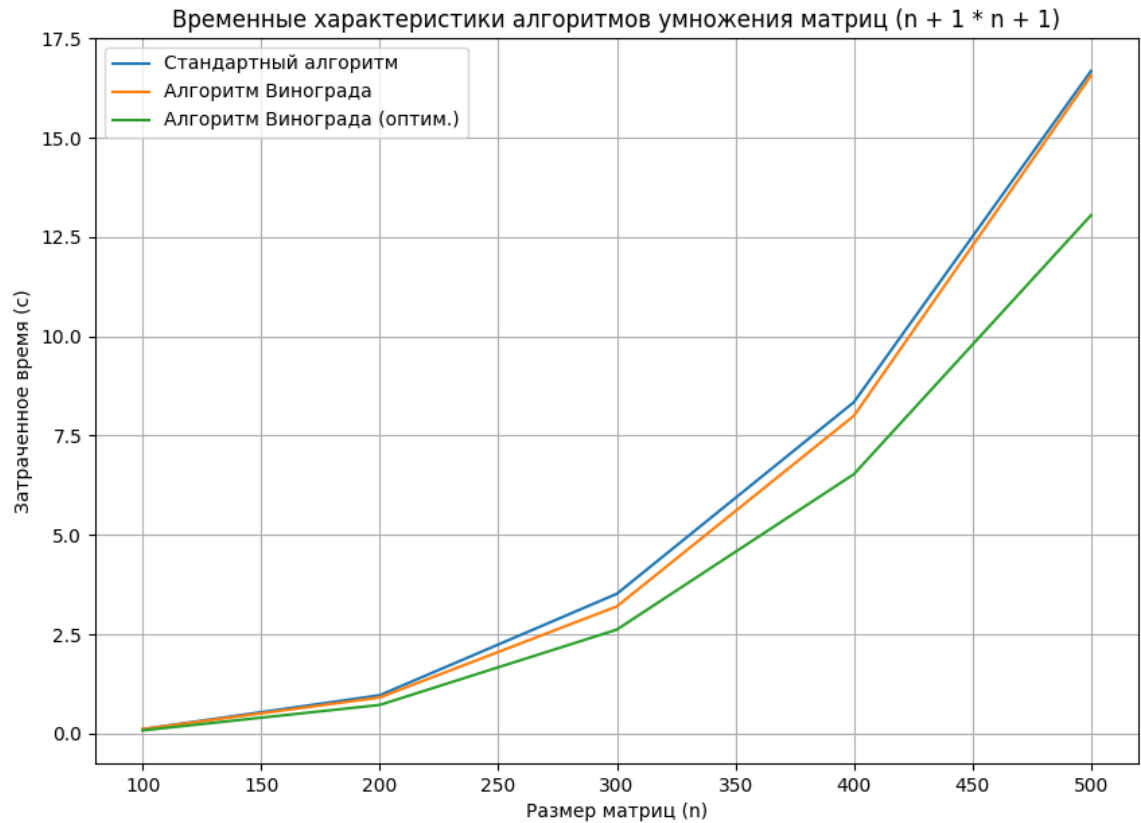


Рисунок 4.3 – Сравнение процессорного времени работы реализаций алгоритмов умножения матриц на квадратных матрицах нечетной размерности  $(n + 1) * (n + 1)$

## 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 —  $(N * M + M * Q) * \text{sizeof}(\text{int})$ ;
- результирующая матрица —  $(N * Q) * \text{sizeof}(\text{int})$ ;
- размерности матриц —  $3 * \text{sizeof}(\text{int})$ .

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

Алгоритм Винограда умножения матриц:

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

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

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

## Вывод

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

Самым эффективным 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: изучены способы оптимизации алгоритмов на примере алгоритмов умножения матриц — стандартный, Винограда.

В ходе выполнения данной работы были решены все задачи:

- 1) изучены два алгоритма умножения матриц — стандартный алгоритм, алгоритм Винограда;
- 2) разработаны три алгоритма умножения матриц — стандартный, Винограда и Винограда с оптимизациями;
- 3) реализованы алгоритмы умножения матриц;
- 4) выполнены замеры процессорного времени работы реализаций алгоритмов умножения матриц;
- 5) проведен сравнительный анализ трудоемкости реализаций разработанных алгоритмов умножения матриц на основе теоретических расчетов;
- 6) выполнена оценка и сравнительный анализ затрат алгоритмов умножения матриц по памяти;
- 7) 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.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Матрицы. Виды матриц [Электронный ресурс]. Режим доступа: [https://vuzlit.com/833194/matritsy\\_vidy\\_matrits](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](https://function-x.ru/operations_with_matrices.html) (дата обращения: 20.10.2022).
3. Умножение матриц по Винограду [Электронный ресурс]. Режим доступа: <http://algotlib.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).