



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"

Студент \_\_\_\_\_ Сукочева Алис

Группа \_\_\_\_\_ ИУ7-53Б

Название предприятия \_\_\_\_\_ МГТУ им. Н. Э. Баумана, каф. ИУ7

Тема \_\_\_\_\_ Алгоритмы умножения матриц

Студент: \_\_\_\_\_ Сукочева А.  
подпись, дата \_\_\_\_\_ Фамилия, И.О.

Преподаватель: \_\_\_\_\_ Волкова Л.Л.  
подпись, дата \_\_\_\_\_ Фамилия, И. О.

Преподаватель: \_\_\_\_\_ Строганов Ю.В.  
подпись, дата \_\_\_\_\_ Фамилия, И. О.

## Содержание

Введение . . . . .	3
1 Аналитический раздел . . . . .	4
1.1 Некоторые теоретические сведения . . . . .	4
1.2 Стандартный алгоритм умножения матриц . . . . .	4
1.3 Умножение матриц по Винограду . . . . .	5
1.4 Вывод . . . . .	5
2 Конструкторский раздел . . . . .	6
2.1 Вывод . . . . .	6
3 Технологический раздел . . . . .	7
3.1 Выбор ЯП . . . . .	7
3.2 Требования к программному обеспечению . . . . .	7
3.3 Сведения о модулях программы . . . . .	7
3.4 Тестирование . . . . .	9
3.5 Вывод . . . . .	9
4 Экспериментальная часть . . . . .	11
4.1 Временные характеристики . . . . .	11
4.2 Сравнительный анализ алгоритмов . . . . .	12
4.3 Вывод . . . . .	12
Заключение . . . . .	13
Список использованных источников . . . . .	14

## Введение

В данной лабораторной работе будут рассмотрены алгоритмы умножения матриц.

Матрицы  $A$  и  $B$  могут быть перемножены, если число столбцов матрицы  $A$  равно числу строк  $B$ .

Умножение матриц активно используется в компьютерной графике. В частности для того, чтобы передвинуть персонажа с координатами  $x, y, z$  на некоторое смещение  $dx, dy, dz$ . В этом случае нужно умножить координаты персонажа на матрицу перемещения. Аналогичная ситуация, если нужно повернуть персонажа. В этом случае матрица перемещения заменяется на матрицу вращения и производится та же операция умножения матриц.

Целью данной работы является изучение, программная реализация, а также сравнение алгоритмов умножения матриц.

В рамках выполнения работы необходимо решить следующие задачи.

- а) Изучить и реализовать на выбранном ЯП стандартный алгоритм умножения матриц.
- б) Изучить и реализовать алгоритм Винограда умножения матриц.
- в) Оптимизировать алгоритм Винограда умножения матриц.
- г) Сравнить временные характеристики вышеизложенных алгоритмов.
- д) Оценить алгоритмы.

# 1 Аналитический раздел

## 1.1 Некоторые теоретические сведения

Для начала нужно ввести собственно понятие матрицы.

*Матрица* – объект, записываемый в виде прямоугольной таблицы элементов, которая представляет собой совокупность строк и столбцов, на пересечении которых находятся её элементы (формула 1.1).

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \quad (1.1)$$

*Произведение матриц*  $AB$  состоит из всех возможных комбинаций скалярных произведений вектор-строк матрицы  $A$  и вектор-столбцов матрицы  $B$  (рис. 1.1).

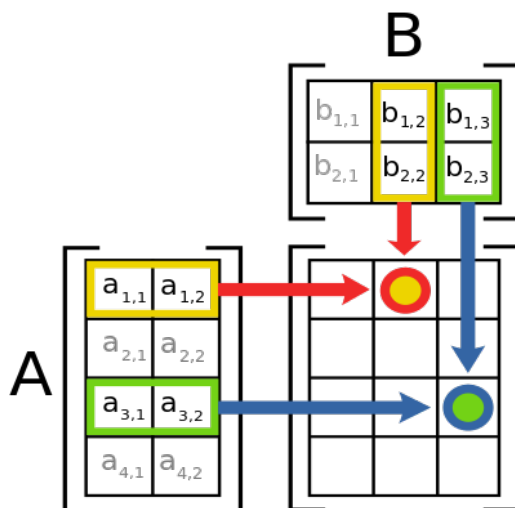


Рисунок 1.1 — Произведение матриц

Операция умножения двух матриц выполнима только в том случае, если число столбцов в первой матрице равно числу строк во второй.

## 1.2 Стандартный алгоритм умножения матриц

Допустим имеется матрицы  $A$  (формула 1.1) и  $B$  (формула 1.2)

$$B = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{pmatrix} \quad (1.2)$$

Матрица  $C = AB$  будет размерностью  $l \times n$ , где матрица  $A$  размерностью  $l \times m$ , а матрица  $B$   $m \times n$ . Тогда каждый элемент матрицы  $C$  выражается формулой (1.3)

$$c_{ij} = \sum_{r=1}^m a_{ir} b_{ri} \quad (i = 1, 2, \dots, l; j = 1, 2, \dots, n) \quad (1.3)$$

### 1.3 Умножение матриц по Винограду

Каждый элемент в матрице  $C$ , которая является результатом умножения двух матриц, представляет собой скалярное произведение соответствующих строки и столбца исходных матриц. В алгоритме умножения матриц по Винограду предложено сделать предварительную обработку, позволяющую часть работы выполнить заранее.

Рассмотрим два вектора  $V$  (формула 1.4) и  $W$  (формула 1.5).

$$V = (v_1, v_2, v_3, v_4) \quad (1.4)$$

$$W = (w_1, w_2, w_3, w_4) \quad (1.5)$$

Их скалярное произведение равно 1.6.

$$V * W = v_1 w_1 + v_2 w_2 + v_3 w_3 + v_4 w_4 \quad (1.6)$$

Равенство 1.6 можно записать в виде 1.7.

$$V * W = (v_1 + w_2)(v_2 + w_1) + (v_3 + w_4)(v_4 + w_3) - v_1 v_2 - v_3 v_4 - w_1 w_2 - w_3 w_4 \quad (1.7)$$

Выражение в правой части равенства 1.7 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.

### 1.4 Вывод

Были рассмотрены основополагающие материалами, которые в дальнейшем потребуются при реализации алгоритмов умножения матриц.

## 2 Конструкторский раздел

В данном разделе будут рассмотрены схемы алгоритмов умножения матриц. На рис. 2.1 представлена схема стандартного алгоритма умножения матриц.

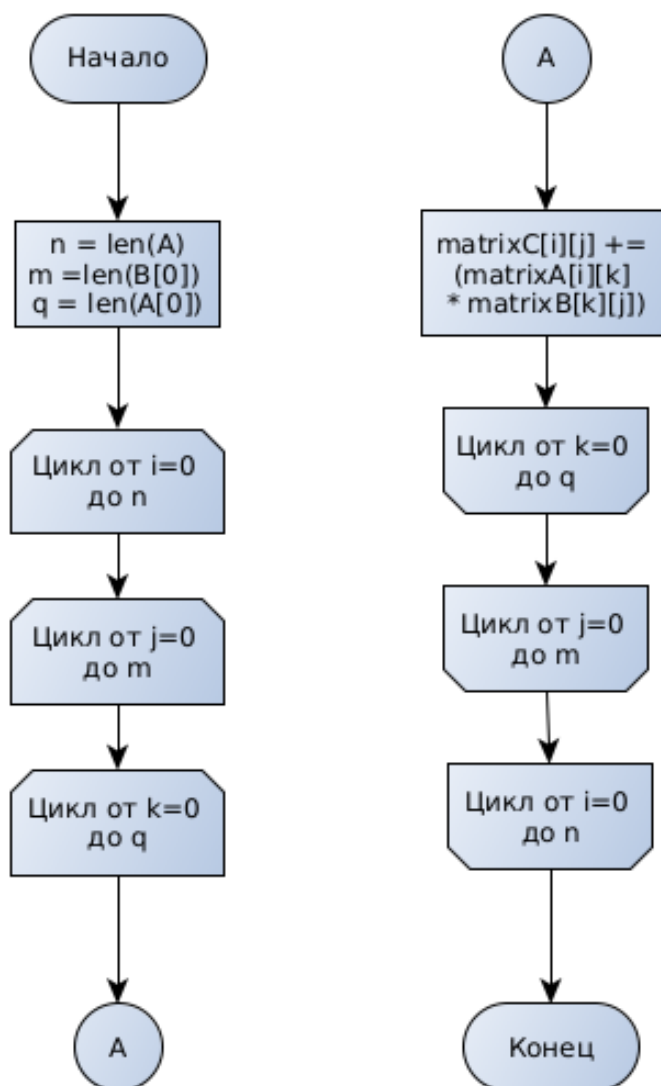


Рисунок 2.1 — Схема стандартного алгоритма умножения матриц

### 2.1 Вывод

В данном разделе были рассмотрены схемы (рис. 2.1 - 2.1) алгоритмов умножения матриц.

## 3 Технологический раздел

### 3.1 Выбор ЯП

В данной лабораторной работе использовался язык программирования - python [1]. Данный язык простой и понятный, также я знакома с ним. Поэтому данный язык был выбран. В качестве среды разработки я использовала Visual Studio Code [2], т.к. считаю его достаточно удобным и легким. Visual Studio Code подходит не только для Windows [3], но и для Linux [4], это еще одна причина, по которой я выбрала VS code, т.к. у меня установлена ОС Ubuntu 18.04.4 [5].

### 3.2 Требования к программному обеспечению

Входными данными являются две матрицы A и B. Количество столбцов матрицы A должно быть равно количеству строк матрицы B.

На выходе получается результат умножения, введенных пользователем, матриц.

### 3.3 Сведения о модулях программы

Данная программа разбита на модули.

- main.py - файл, содержащий точку входа в программу. В нем происходит общение с пользователем и вызов алгоритмов.
- matrix.py - файл, содержащий класс matrix.
- matrix\_multiplication.py - файл, содержащий реализации алгоритмов умножения матриц.

На листингах 3.1-3.4 представлен код программы.

Листинг 3.1 — Главная функция main

```
1 def main():
2     output("МАТРИЦА A", BLUE)
3     n = int(input(GREEN + "Введите кол-во строк: "))
4     m = int(input(GREEN + "Введите кол-во столбцов: "))
5     output("Введите матрицу:", GREEN)
6     matrixA = Matrix(n, m, [[int(j) for j in input(GREEN).split()]
7                               for i in range(n)])
8
9     output("МАТРИЦА B", BLUE)
10    k = int(input(GREEN + "Введите кол-во строк: "))
11    l = int(input(GREEN + "Введите кол-во столбцов: "))
12    output("Введите матрицу:", GREEN)
13    matrixB = Matrix(k, l, [[int(j) for j in input(GREEN).split()]
14                              for i in range(k)])
15
```

```

16     if m != k:
17         output("Некорректные размеры матриц!", RED)
18         return
19
20     matrixC = multiplication(matrixA, matrixB)
21     matrixC.output()

```

Листинг 3.2 — Класс matrix

```

1  class Matrix:
2      n, m = 0, 0
3      matrix = list()
4
5      def __init__(self, n, m, list_of_lists=None):
6          self.n, self.m = n, m
7          if list_of_lists:
8              self.matrix = deepcopy(list_of_lists)
9          else:
10             self.matrix = np.full((n, m), 0)
11
12     def output(self):
13         print(TURQUOISE, end='')
14         for i in range(self.n):
15             for j in range(self.m):
16                 print(self.matrix[i][j], end=' ')
17             print()
18
19     def fill(self, list_of_lists):
20         self.matrix = deepcopy(list_of_lists)
21
22     def size(self):
23         return (self.n, self.m)
24
25     def __getitem__(self, index):
26         return self.matrix[index]

```

Листинг 3.3 — Стандартный алгоритм умножения матриц

```

1  def multiplication(matrixA, matrixB):
2      n = matrixA.n
3      m = matrixB.m
4      q = matrixB.n
5
6      matrixC = Matrix(n, m)
7
8      for i in range(n):
9          for j in range(m):
10             for k in range(q):

```



```

11         matrixC[i][j] += (matrixA[i][k] * matrixB[k][j])
12
13     return matrixC

```

### Листинг 3.4 — Алгоритм Винограда

```

1  def WinogradMult(matrixA, matrixB):
2      n = matrixA.n
3      m = matrixB.m
4      q = matrixB.n
5
6      matrixC = Matrix(n, m)
7
8      tempA = np.full(n, 0)
9      for i in range(n):
10         for j in range(1, q, 2):
11             tempA[i] += matrixA[i][j - 1] * matrixA[i][j]
12
13     tempB = np.full(m, 0)
14     for i in range(m):
15         for j in range(1, q, 2):
16             tempB[i] += matrixB[j - 1][i] * matrixB[j][i]
17
18     for i in range(n):
19         for j in range(m):
20             matrixC[i][j] -= (tempA[i] + tempB[j])
21             for k in range(1, q, 2):
22                 matrixC[i][j] += (matrixA[i][k-1] + matrixB[k][j]) * \
23                     (matrixA[i][k] + matrixB[k-1][j])
24     if q % 2:
25         for i in range(n):
26             for j in range(m):
27                 matrixC[i][j] += matrixA[i][q-1] * matrixB[q-1][j]
28
29     return matrixC

```

## 3.4 Тестирование

В данном разделе будет приведена таблица с тестами (таблица 3.1).

Все тесты пройдены.

## 3.5 Вывод

В данном разделе был разобран выбор языка программирования, а также среды разработки. Разобраны требования к программному обеспечению. Протестированная программа.

Таблица 3.1 — Таблица тестов

Матрица А	Матрица В	Результат
2 2 1 0 0 1	2 2 1 0 0 1	Ответ верный
3 2 2 3 1 0 2 2	2 4 2 2 1 9 4 2 8 1	Ответ верный
2 1 2 1	12 12	Ответ верный
2 2 1 0 0 1	1 1 0	Сообщение о неверном вводе размерностей
0 0	0 0	

## 4 Экспериментальная часть

В данном разделе будет произведено сравнение вышеизложенных алгоритмов.

### 4.1 Временные характеристики

Для сравнения возьмем квадратные матрицы размерностью  $[10, 20, 30, \dots, 100]$ . Так как подсчет умножения матриц считается короткой задачей, воспользуемся усреднением массового эксперимента. Для этого сложим результат работы алгоритма  $n$  раз ( $n \geq 10$ ), после чего поделим на  $n$ . Тем самым получим достаточно точные характеристики времени. Сравнение произведем при  $n = 50$ . Результат можно увидеть на рис. 4.1.

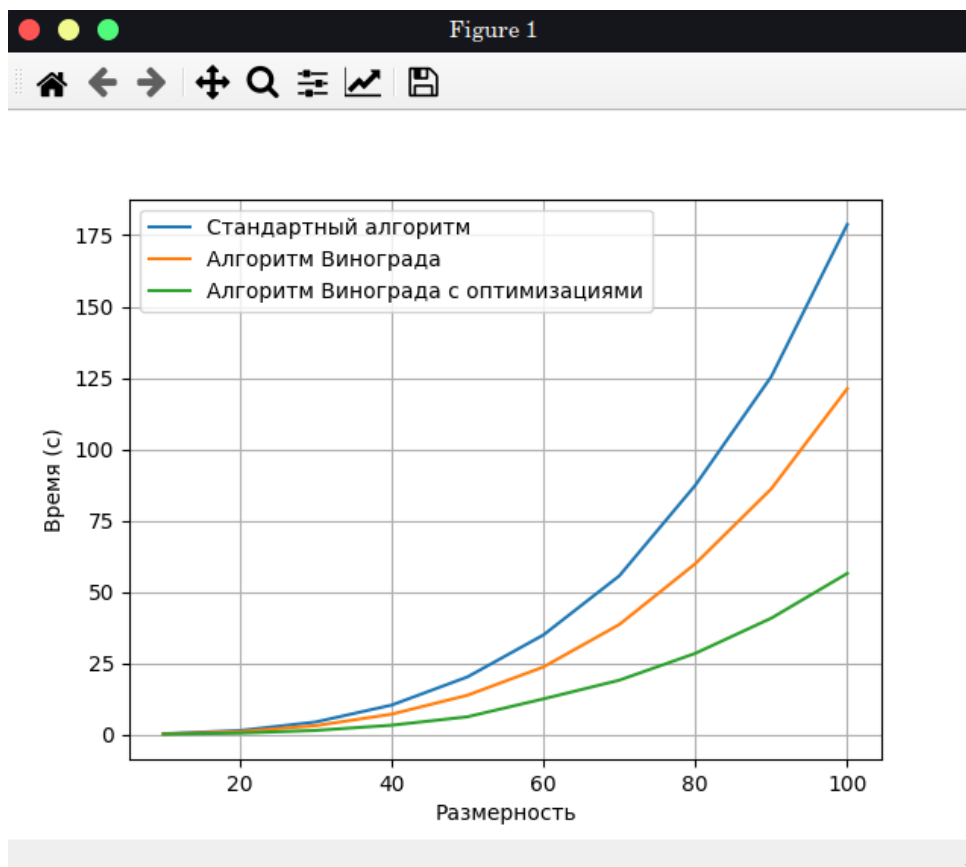


Рисунок 4.1 — Временные характеристики на четных размерах матриц

На рис. 4.2 показана работа алгоритмов с матрицами, размерностью  $[11, 21, 31, \dots, 91]$ .

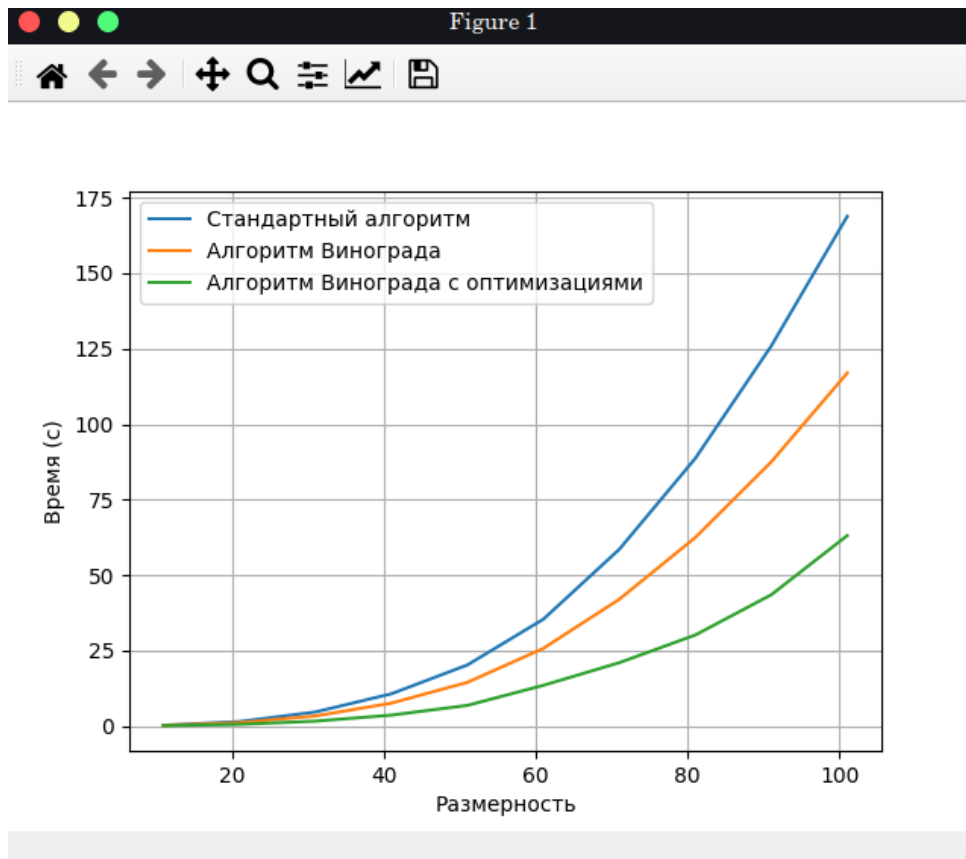


Рисунок 4.2 — Временные характеристики на нечетных размерах матриц

## 4.2 Сравнительный анализ алгоритмов

### 4.3 Вывод

В данном разделе было произведено сравнение количества затраченного времени вышеизложенных алгоритмов. Самым быстрым оказался модифицированный алгоритм Винограда.

## Заключение

Алгоритмы умножения матриц являются очень важными алгоритмами В этой лабораторной работе были изучены алгоритмы ... (тут ссылки на все, что было сделано)

В рамках выполнения работы решены следующие задачи.

- а) Изучен и реализован на выбранном ЯП стандартный алгоритм умножения матриц.
- б) Изучен и реализован алгоритм Винограда умножения матриц.
- в) Оптимизирован алгоритм Винограда умножения матриц.
- г) Произведены сравнения временные характеристик вышеизложенных алгоритмов.
- д) Оценены алгоритмы.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Майкл, Доусон*. Python Programming for the Absolute Beginner, 3rd Edition / Доусон Майкл. — Прогресс книга, 2019. — Р. 416.
2. Visual Studio Code. — Microsoft, 2005. <https://code.visualstudio.com/>.
3. Windows. — Microsoft, 1985. <https://www.microsoft.com/ru-ru/windows>.
4. Linux. — 1991. <https://www.linux.org.ru/>.
5. Ubuntu 18.04. — 2018. <https://releases.ubuntu.com/18.04/>.