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.

Москва — 2020 г.

Содержание

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).

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

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

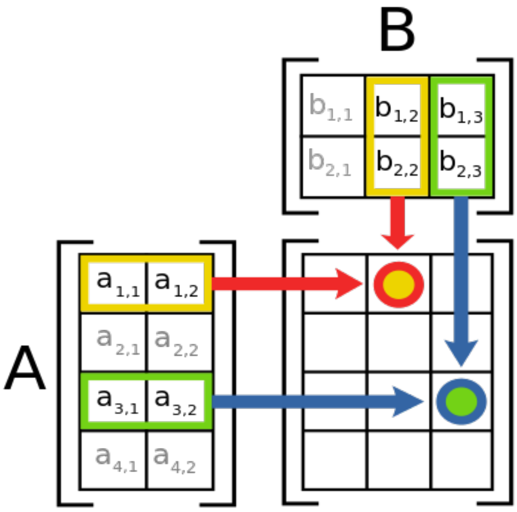

Рисунок 1.1 — Произведение матриц

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)

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

Матрица С = АВ будет размерностью $l \times n$, где матрица А размерностью $l \times m$, а матрица В $m \times n$ Тогда каждый элемент матрицы С выражается формулой (1.3)

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

1.3 Умножение матриц по Винограду

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

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

$$V = (v_1, v_2, v_3, v_4) \tag{1.4}$$

$$W = (w_1, w_2, w_3, w_4) \tag{1.5}$$

Их скалярное произведение равно 1.6.

$$V * W = v_1 w_1 + v_2 w_2 + v_3 w_3 + v_4 w_4 \tag{1.6}$$

Равенство 1.6 можно записать в виде 1.7.

$$\begin{aligned} V * W = (v_1 + w_2)(v_2 + w_1) + (v_3 + w_4)(v_4 + w_3) - \\ v_1 v_2 - v_3 v_4 - w_1 w_2 - w_3 w_4 \end{aligned} \tag{1.7}$$

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.

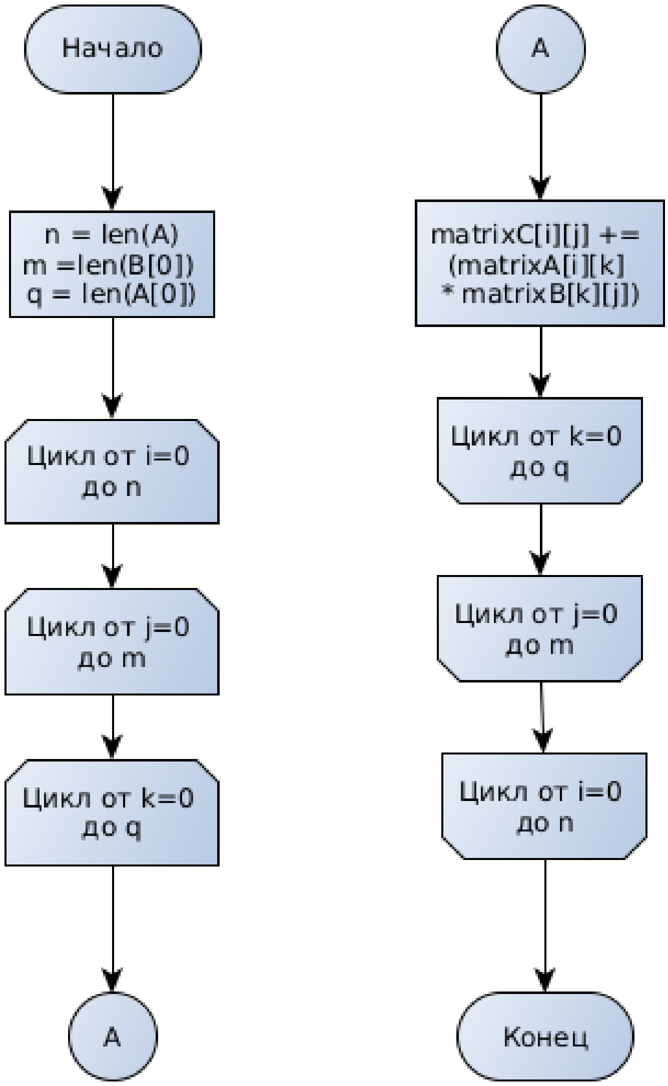

Рисунок 2.1 — Схема стандартного алгоритма умножения матриц

На рис. 2.2 - 2.3 представлена схема алгоритма Винограда умножения матриц
.

2.1 Вывод

В данном разделе были рассмотрены схемы (рис. 2.1 - 2.3) алгоритмов умножения матриц.

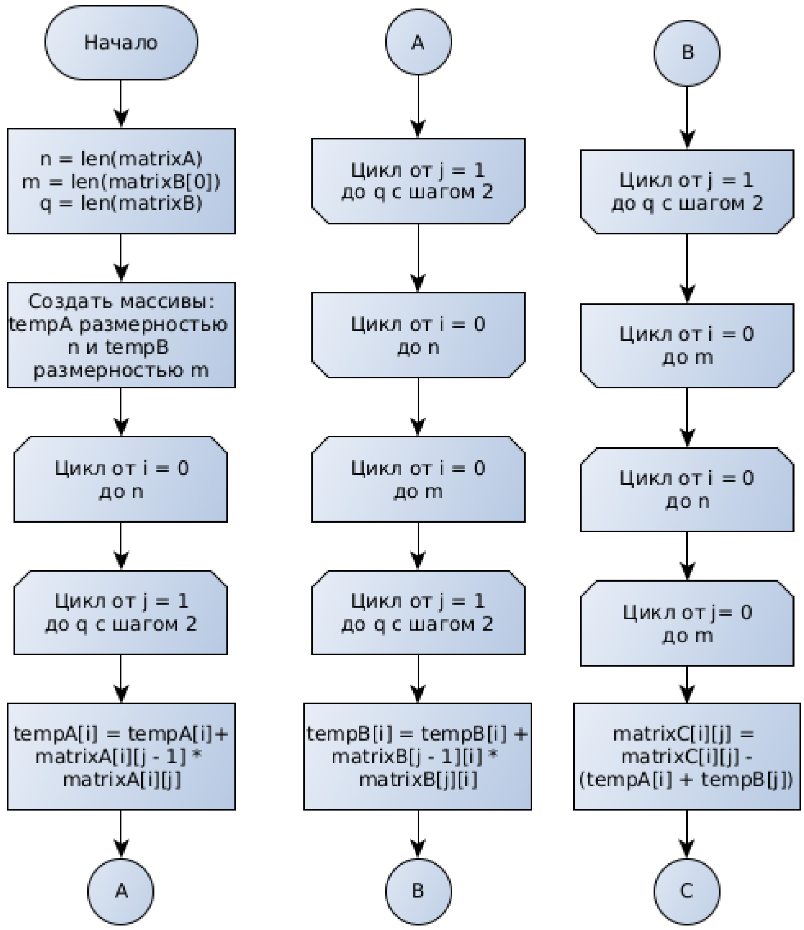

Рисунок 2.2 — Схема алгоритма Винограда умножения матриц

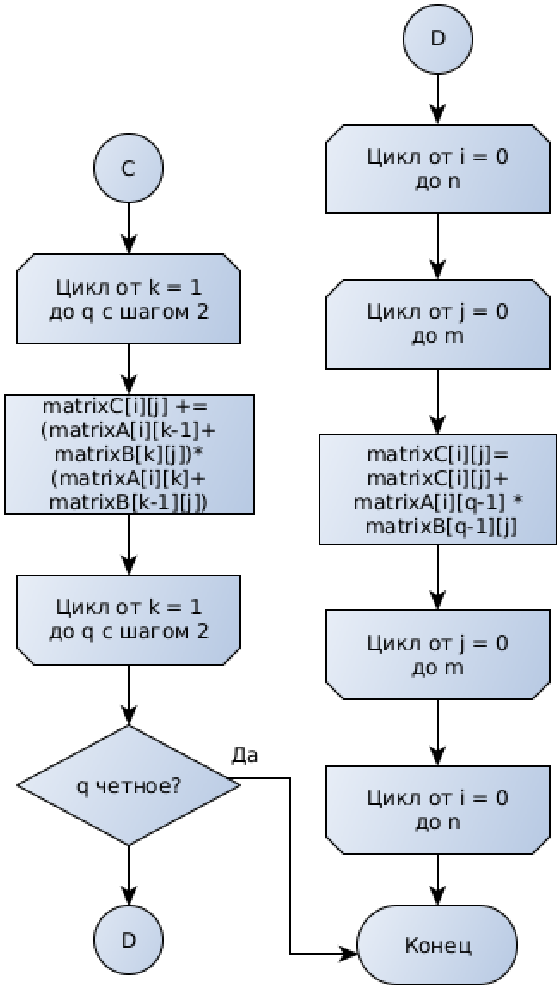

Рисунок 2.3 — Схема алгоритма Винограда умножения матриц

3 Технологический раздел

3.1 Выбор ЯП

В данной лабораторной работе использовался язык программирования - python [1]. Я знакома с данным языком. 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 долджно 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

```
def main():
    output("МАТРИЦА A", BLUE)
    n = int(input(GREEN + "Введите кол—во строк: "))
    m = int(input(GREEN + "Введите кол—во столбцов: "))
    output("Введите матрицу:", GREEN)
    matrixA = Matrix(n, m, [[int(j) for j in input(GREEN).split()]
                            for i in range(n)])

    output("МАТРИЦА B", BLUE)
    k = int(input(GREEN + "Введите кол—во строк: "))
    l = int(input(GREEN + "Введите кол—во столбцов: "))
    output("Введите матрицу:", GREEN)
    matrixB = Matrix(k, l, [[int(j) for j in input(GREEN).split()]
                            for i in range(k)])

```

```
    if m != k:
        output("Некорректные размеры матриц!", RED)
        return

    matrixC = multiplication(matrixA, matrixB)
    matrixC.output()
```

Листинг 3.2 — Класс matrix

```
class Matrix:
    n, m = 0, 0
    matrix = list()

    def __init__(self, n, m, list_of_lists=None):
        self.n, self.m = n, m
        if list_of_lists:
            self.matrix = deepcopy(list_of_lists)
        else:
            self.matrix = np.full((n, m), 0)

    def output(self):
        print(TURQUOISE, end='')
        for i in range(self.n):
            for j in range(self.m):
                print(self.matrix[i][j], end=' ')
            print()

    def fill(self, list_of_lists):
        self.matrix = deepcopy(list_of_lists)

    def size(self):
        return (self.n, self.m)

    def __getitem__(self, index):
        return self.matrix[index]
```

Листинг 3.3 — Стандарный алгоритм умножения матриц

```
    def multiplication(matrixA, matrixB):
        n = matrixA.n
        m = matrixB.m
        q = matrixB.n

        matrixC = Matrix(n, m)

        for i in range(n):
            for j in range(m):
                for k in range(q):
```

```
                matrixC[i][j] += (matrixA[i][k] * matrixB[k][j])

    return matrixC
```

Листинг 3.4 — Алгоритм Винограда

```
def WinogradMult(matrixA, matrixB):
    n = matrixA.n
    m = matrixB.m
    q = matrixB.n

    matrixC = Matrix(n, m)

    tempA = np.full(n, 0)
    for i in range(n):
        for j in range(1, q, 2):
            tempA[i] += matrixA[i][j - 1] * matrixA[i][j]

    tempB = np.full(m, 0)
    for i in range(m):
        for j in range(1, q, 2):
            tempB[i] += matrixB[j - 1][i] * matrixB[j][i]

    for i in range(n):
        for j in range(m):
            matrixC[i][j] -= (tempA[i] + tempB[j])
            for k in range(1, q, 2):
                matrixC[i][j] += (matrixA[i][k-1] + matrixB[k][j]) * \
                    (matrixA[i][k] + matrixB[k-1][j])
    if q % 2:
        for i in range(n):
            for j in range(m):
                matrixC[i][j] += matrixA[i][q-1] * matrixB[q-1][j]

    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 A	Матрица B	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,...,100]. Так как подсчет умножения матриц считается короткой задачей, воспользуемся усреднением массового эксперимента. Для этого сложим результат работы алгоритма n раз (n >= 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.

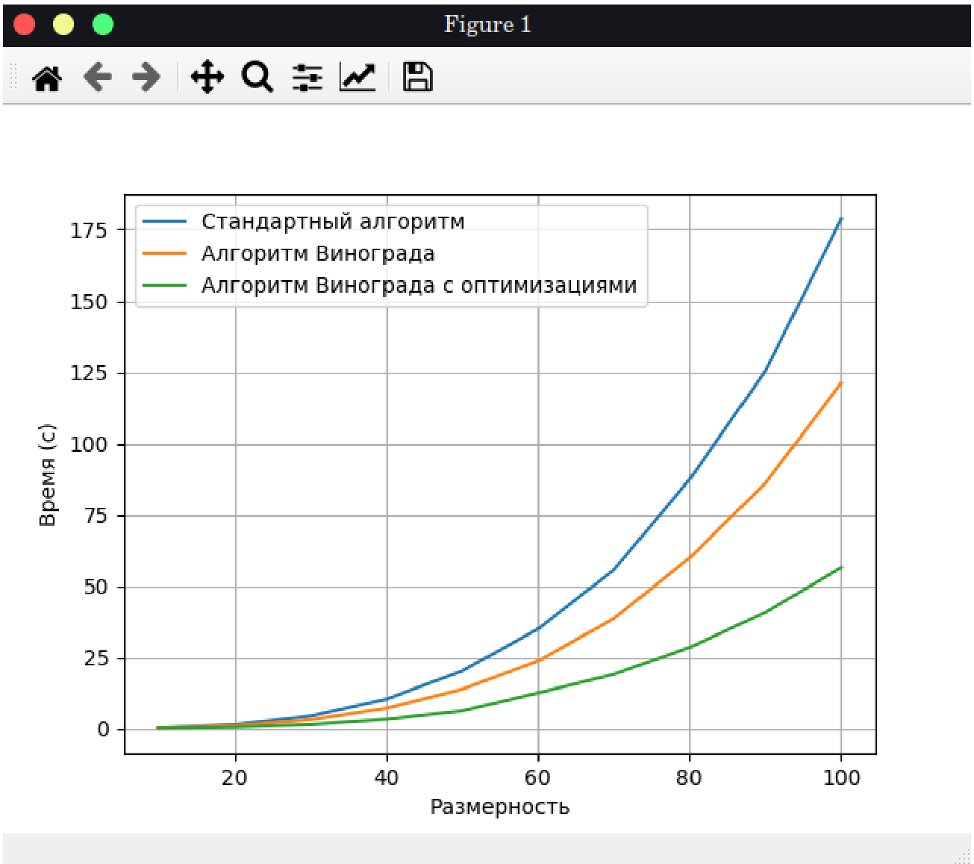

Рисунок 4.1 — Временные характеристики на четных размерах матриц

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

4.2 Сравнительный анализ алгоритмов

Введем модель вычислений трудоемкости алгоритма. Пусть трудоемкость 1 у следующих базовых операций: +, -, *, /, %, =, ==, !=, <, <=, >, >=, []. Трудоемкость

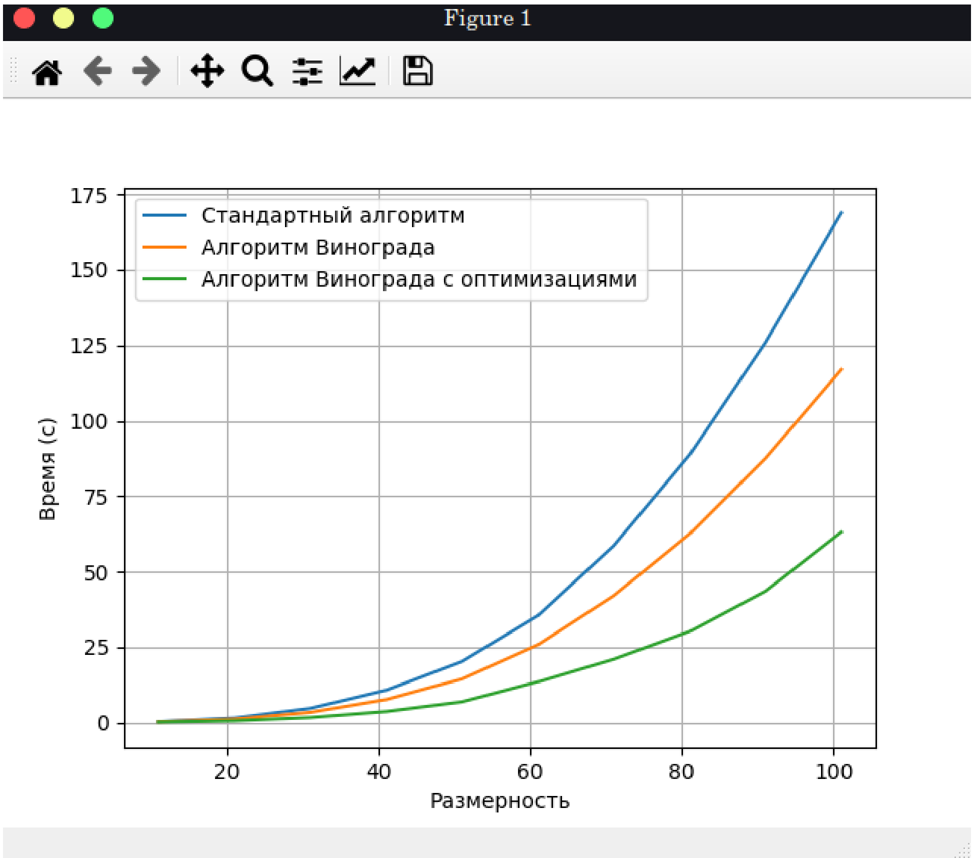

Рисунок 4.2 — Временные характеристики на нечетных размерах матриц

цикла: fцикла = fиниц + fсравн + Nитер ∗ (fтела + fинкрем + fсравн).Трудоемкость условного перехода 1.

Стандартная реализация алгоритма не эффективна по времени, так как обладает трудоемкостью 5qmn + 4n + 4mn + 5. Оценка трудоемкости данного алгоритма составляет 5qmn. По памяти в стандартном алгоритме умножения матриц требуется m*n памяти под результат.

Теперь рассмотрим алгоритм Винограда умножения матриц. Реализация алгоритма Винограда обладает трудоемкостью формула 4.1. Оценка трудоемкости данного алгоритма составляет 3qmn. В алгоритме Винограда умножения матриц требуется дополнительно m+n памяти под результат.

$$3qmn + 7mn + 2m + 5q + 6n + 11 + \left[\begin{array}{c} 0 \text{ л.с.} \\ 5mn + 4n + 2 \text{ х.с.} \end{array}\right. \tag{4.1}$$

4.3 Вывод

В данном разделе было произведено сравнение количества затраченного времени вышеизложенных алгоритмов. Самым быстрым оказался модифицированный алгоритм Винограда. При этом в алгоритме Винограда умножения матриц требуется дополнительно m+n памяти под результат.

Заключение

Алгоритмы умножения матриц являются очень важными алгоритмами. Алгоритм Винограда умножения матриц выиграл по времени, но за это он платит дополнительной памятью. В этой лабораторной работе были рассмотрены формулы для умножения матриц (1.3 и 1.7) Также рассмотрены схемы алгоритмов умножения матриц (рис. 2.1 - 2.3). Выбран и обоснован ЯП. А также приведены листинги (3.1-3.4), на которых показаны алгоритмы умножения матриц.

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. — P. 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/`.