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 Алгоритм Копперсмита-Винограда

Студент Криков А.В.

Группа ИУ7-53Б

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

Москва — 2021 г.

Оглавление

Введение

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

При достаточно большом количестве объектов возникает необходимость минимизировать временные затраты на произведение матриц.

Алгоритм Копперсмита — Винограда — алгоритм умножения квадратных матриц, предложенный в 1987 году Д. Копперсмитом и Ш. Виноградом [1]. В исходной версии асимптотическая сложность алгоритма составляла $O(n^{2,3755})$, где n — размер стороны матрицы. Алгоритм Копперсмита — Винограда, с учетом серии улучшений и доработок в последующие годы, обладает лучшей асимптотикой среди известных алгоритмов умножения матриц [2].

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

Для достижения поставленной цели необходимо :

1. изучить и реализовать 2 алгоритма перемножения матриц: стандартный и Копперсмита-Винограда;

2. сравнить трудоёмкость алгоритмов на основе теоретических расчетов;

3. сравнить алгоритмы на основе экспериментальных данных;

4. подготовить отчет по лабораторной работе.

1 Аналитическая часть

1.1 Описание алгоритмов

1.1.1 Стандартный алгоритм

Пусть даны две прямоугольные матрицы

$$A_{lm} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{l1} & a_{l2} & \dots & a_{lm} \end{pmatrix}, \quad B_{mn} = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{pmatrix}, \tag{1.1}$$

тогда матрица C

$$C_{ln} = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{l1} & c_{l2} & \dots & c_{ln} \end{pmatrix}, \tag{1.2}$$

где

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

будет называться произведением матриц A и B. Стандартный алгоритм реализует данную формулу.

1.1.2 Алгоритм Копперсмита — Винограда

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 (1.4):

$$V \cdot W = (v_1+w_2)(v_2+w_1)+(v_3+w_4)(v_4+w_3)-v_1v_2-v_3v_4-w_1w_2-w_3w_4. \quad (1.4)$$

Несмотря на то, что второе выражение требует вычисления большего количества операций, чем стандартный алгоритм: вместо четырех умножений - шесть, а вместо трех сложений - десять, 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, что позволит для каждого элемента выполнять лишь два умножения и пять сложений, складывая затем только лишь с 2 предварительно посчитанными суммами соседних элементов текущих строк и столбцов. Из-за того, что операция сложения быстрее операции умножения в ЭВМ, на практике алгоритм должен работать быстрее стандартного [3].

Вывод

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

2 Конструкторская часть

2.1 Разработка алгоритмов

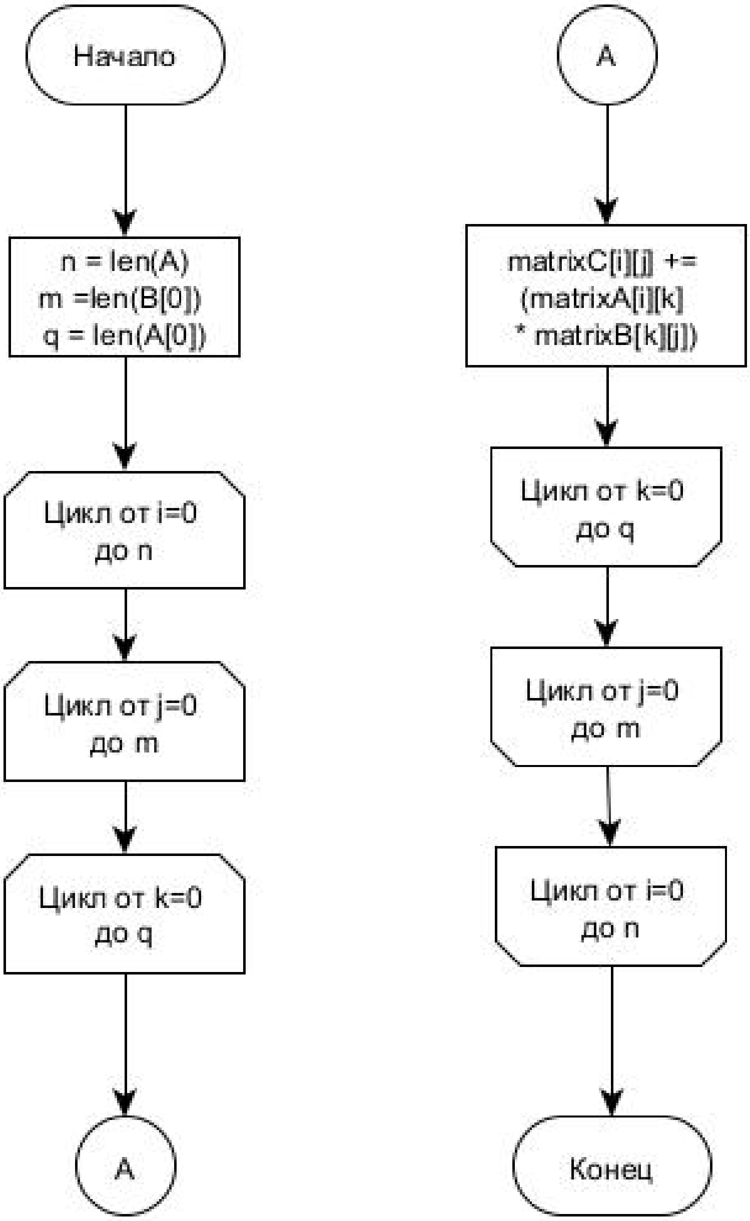

Рис. 2.1: Стандартный алгоритм заполнения матриц

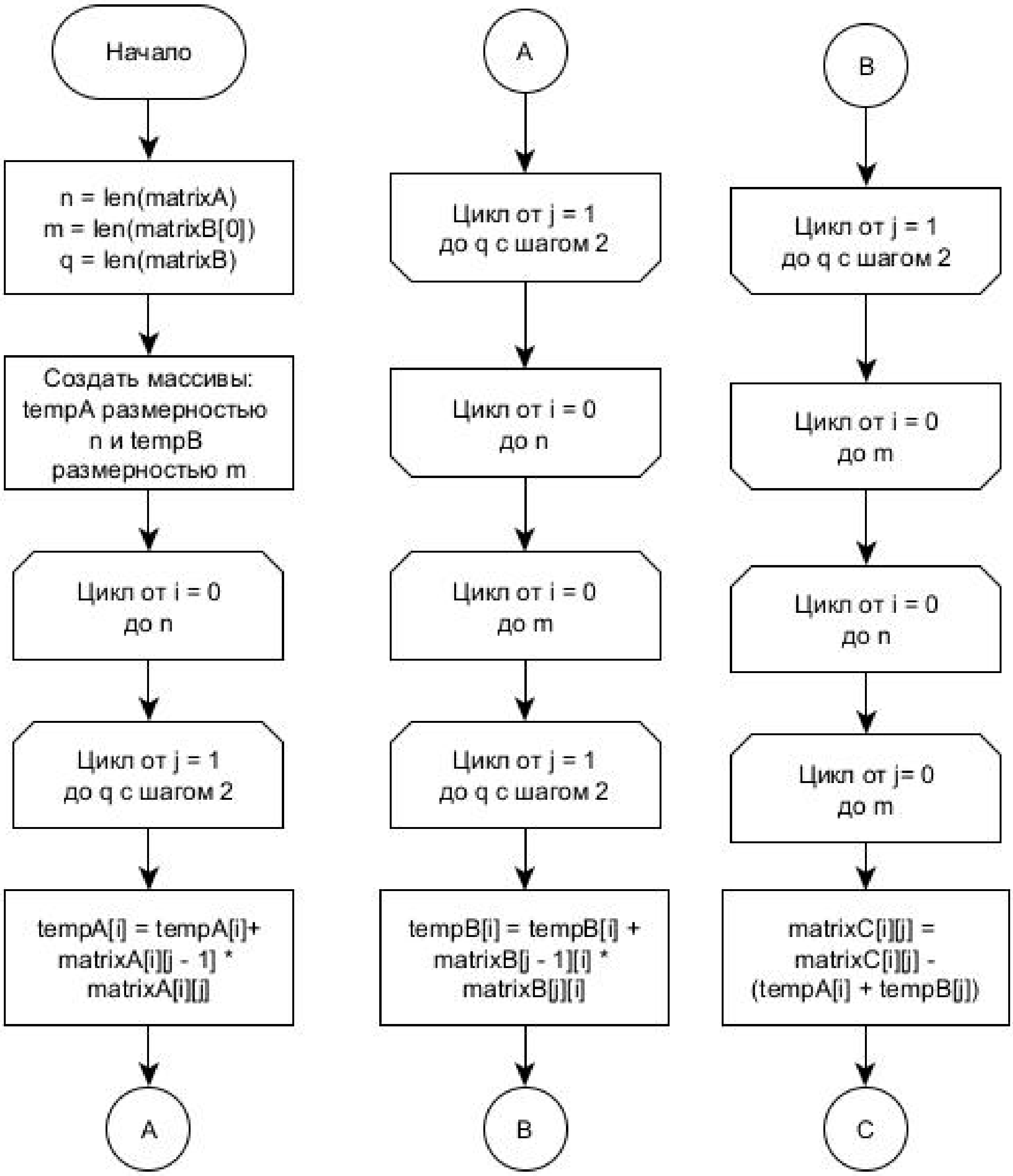

Рис. 2.2: Схема алгоритма Копперсмита — Винограда

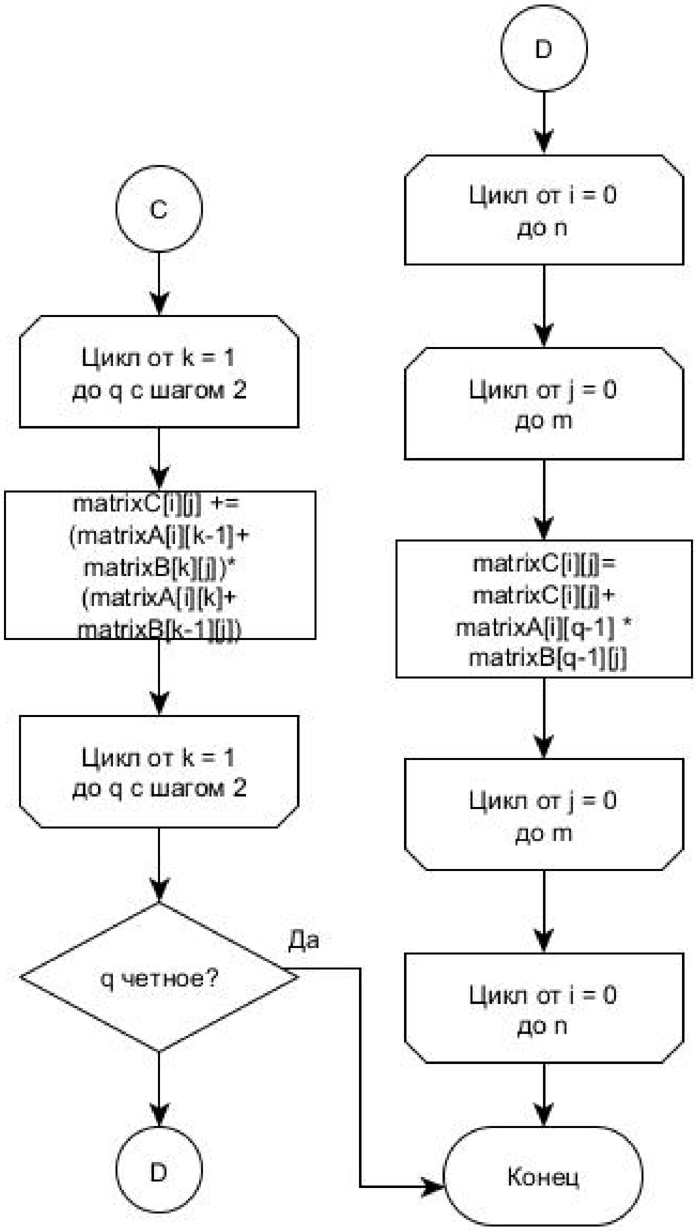

Рис. 2.3: Схема алгоритма Копперсмита — Винограда

2.2 Модель вычислений

Для последующего вычисления трудоемкости необходимо ввести модель вычислений:

1. операции из списка (2.1) имеют трудоемкость 1;

$$+, -, /, \%, ==, != , <, >, <=, >=, [], ++, -- \tag{2.1}$$

2. трудоемкость оператора выбора `if условие then A else B` рассчитывается, как (2.2);

$$f_{if} = f_{\text{условия}} + \begin{cases} f_A, & \text{если условие выполняется,} \\ f_B, & \text{иначе.} \end{cases} \tag{2.2}$$

3. трудоемкость цикла рассчитывается, как (2.3);

$$f_{for} = f_{\text{инициализации}} + f_{\text{сравнения}} + N(f_{\text{тела}} + f_{\text{инкремента}} + f_{\text{сравнения}}) \tag{2.3}$$

4. трудоемкость вызова функции равна 0.

2.3 Трудоемкость алгоритмов

2.3.1 Стандартный алгоритм умножения матриц

Во всех последующих алгоритмах не будем учитывать инициализацию матрицу, в которую записывается результат, потому что данное действие есть во всех алгоритмах и при этом не является самым трудоёмким.

Трудоёмкость стандартного алгоритма умножения матриц состоит из:

- внешнего цикла по $i \in [1..n]$, трудоёмкость которого: $f = 2 + n \cdot (2 + f_{body})$;
- цикла по $j \in [1..m]$, трудоёмкость которого: $f = 2 + m \cdot (2 + f_{body})$;
- цикла по $k \in [1..q]$, трудоёмкость которого: $f = 2 + 10q$;

Учитывая, что трудоёмкость стандартного алгоритма равна трудоёмкости внешнего цикла, можно вычислить ее, подставив циклы тела (2.4):

$$f_{standard} = 2 + n \cdot (4 + m \cdot (4 + 10q)) = 2 + 4n + 4nm + 10nmq \approx 10nmq \tag{2.4}$$

2.3.2 Алгоритм Копперсмита — Винограда

Трудоёмкость алгоритма Копперсмита — Винограда состоит из:

1. создания и инициализации массивов tempA и tempB, трудоёмкость которого (2.5):
$$f_{init} = n + m; \tag{2.5}$$

2. заполнения массива tempA, трудоёмкость которого (2.6):
$$f_{tempA} = 3 + \frac{q}{2} \cdot (5 + 12n); \tag{2.6}$$

3. заполнения массива tempB, трудоёмкость которого (2.7):
$$f_{tempB} = 3 + \frac{q}{2} \cdot (5 + 12m); \tag{2.7}$$

4. цикла заполнения для чётных размеров, трудоёмкость которого (2.8):
$$f_{cycle} = 2 + n \cdot (4 + m \cdot (11 + \frac{25}{2} \cdot q)); \tag{2.8}$$

5. цикла, для дополнения умножения суммой последних нечётных строки и столбца, если общий размер нечётный, трудоемкость которого (2.9):
$$f_{last} = \begin{cases} 2, & \text{чётная,} \\ 4 + n \cdot (4 + 14m), & \text{иначе.} \end{cases} \tag{2.9}$$

Итого, для худшего случая (нечётный общий размер матриц) имеем (2.10):

$$f = n + m + 12 + 8n + 5q + 6nq + 6mq + 25nm + \frac{25}{2}nmq \approx 12.5 \cdot nmq \tag{2.10}$$

Для лучшего случая (чётный общий размер матриц) имеем (2.11):

$$f = n + m + 10 + 4n + 5q + 6nq + 6mq + 11nm + \frac{25}{2}nmq \approx 12.5 \cdot nmq \tag{2.11}$$

Вывод

На основе теоретических данных, полученных из аналитического раздела, были построены схемы обоих алгоритмов умножения матриц. Оценены их трудоёмкости в лучшем и худшем случаях.

3 Технологическая часть

В данном разделе приведены средства реализации и листинг кода.

3.1 Требования к ПО

К программе предъявляется ряд требований:

- на вход подаются размеры 2 матриц, а также их элементы;
- на выходе — матрица, которая является результатом умножения входных матриц.

3.2 Средства реализации

В качестве языка программирования был выбран ЯП Java. Данный выбор обусловлен тем, что данный язык легок в использовании, быстр и поддерживает кроссплатформенность.

3.3 Листинг кода

Листинг 3.1: Стандартный алгоритм умножения матриц

```
public static Double [][] standartMult(Double[][] matrixA, Double[][] matrixB) {
    int n = matrixA.length;
    int m = matrixB[0].length;
    int q = matrixB.length;

    Double[][] matrixC = new Double[n][m];
    for (int i = 0; i < n; i++) {
        Arrays.fill(matrixC[i], 0.0);
    }

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            for (int k = 0; k < q; k++) {
                matrixC[i][j] += (matrixA[i][k] * matrixB[k][j]);
            }
```

```
        }
    }

    return matrixC;
}
```

Листинг 3.2: Алгоритм Копперсмита — Винограда

```
public static Double [][] vinogradMult(Double[][] matrixA, Double[][] matrixB) {
    int n = matrixA.length;
    int m = matrixB[0].length;
    int q = matrixB.length;

    Double[][] matrixC = new Double[n][m];
    Arrays.fill(matrixC, 0);

    Double[] tempA = new Double[n];
    Arrays.fill(tempA, 0.0);
    for (int i = 0; i < n; i++) {
        for (int j = 1; j < q; j += 2) {
            tempA[i] += matrixA[i][j - 1] * matrixA[i][j];
        }
    }

    Double[] tempB = new Double[m];
    Arrays.fill(tempB, 0.0);
    for (int i = 0; i < m; i++) {
        for (int j = 1; j < q; j += 2) {
            tempB[i] += matrixB[j - 1][i] * matrixB[j][i];
        }
    }


    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            matrixC[i][j] -= (tempA[i] + tempB[j]);
            for (int k = 1; k < q; k += 2) {
                matrixC[i][j] += (matrixA[i][k - 1] + matrixB[k][j]) * (matrixA[i][k]
                    + matrixB[k - 1][j]);
            }
        }
    }

    if (q % 2 == 1) {
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                matrixC[i][j] += matrixA[i][q - 1] * matrixB[q - 1][j];
            }
        }
```

```
        }

        return matrixC;
    }
```

3.4 Тестирование функций

В таблице 3.1 приведены тесты для функций, реализующих стандартный алгоритм умножения матриц и алгоритм Винограда. Тесты пройдены успешно.

Матрица 1	Матрица 2	Ожидаемый результат
$\begin{pmatrix}1 & 1 & 1\\2 & 2 & 2\\3 & 3 & 3\end{pmatrix}$	$\begin{pmatrix}2 & 2 & 2\\3 & 3 & 3\\1 & 1 & 1\end{pmatrix}$	$\begin{pmatrix}6 & 6 & 6\\12 & 12 & 12\\18 & 18 & 18\end{pmatrix}$
$\begin{pmatrix}1 & 5 & 5\\1 & 5 & 5\end{pmatrix}$	$\begin{pmatrix}1 & 3\\1 & 3\\1 & 3\end{pmatrix}$	$\begin{pmatrix}11 & 33\\11 & 33\end{pmatrix}$
$\begin{pmatrix}2\end{pmatrix}$	$\begin{pmatrix}2\end{pmatrix}$	$\begin{pmatrix}4\end{pmatrix}$
$\begin{pmatrix}1 & -2 & 3\\1 & 2 & 3\\1 & 2 & 3\end{pmatrix}$	$\begin{pmatrix}-1 & 2 & 3\\1 & 2 & 3\\1 & 2 & 3\end{pmatrix}$	$\begin{pmatrix}0 & 4 & 6\\4 & 12 & 18\\4 & 12 & 18\end{pmatrix}$
$\begin{pmatrix}111 & 222\end{pmatrix}$	$\begin{pmatrix}4 & 0\end{pmatrix}$	Не могут быть перемножены

Таблица 3.1: Тестирование функций

Вывод

Правильный выбор инструментов разработки позволил эффективно реализовать алгоритмы, настроить модульное тестирование и выполнить исследовательский раздел лабораторной работы.

4 Исследовательская часть

В данном разделе будет произведено сравнение вышеизложенных алгоритмов.

4.1 Технические характеристики

- Операционная система: Windows 10.
- Память: 16 GiB.
- Процессор: Intel(R) Core(TM) i7-4700HQ CPU @ 2.40GHz.

4.2 Временные характеристики

Для сравнения возьмем квадратные матрицы размерностью [10, 20, 30, ... 100]. Так как подсчет умножения матриц считается короткой считается задачей, возспользуемся 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 произведен при n = 50. Результат можно увидеть на рис 4.1.

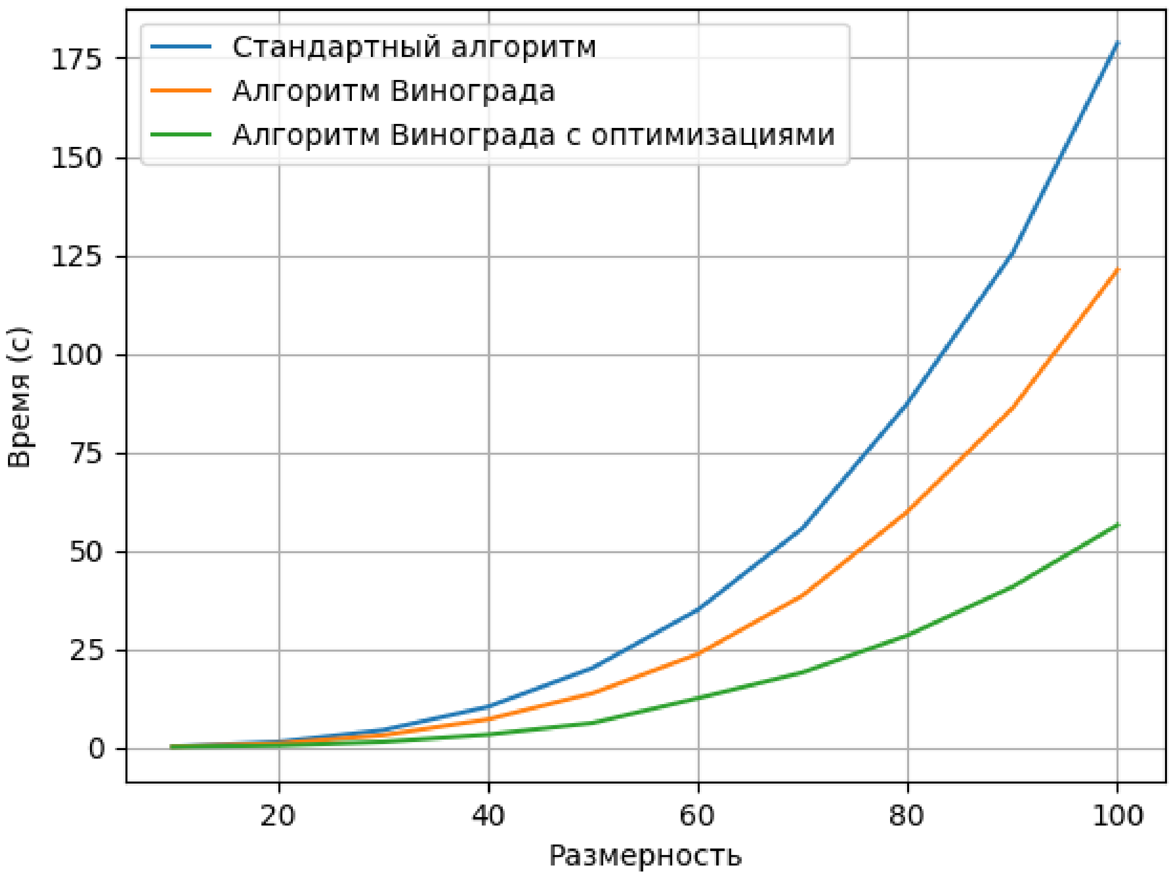

Рис. 4.1: Временные характеристики на четных размерах матриц

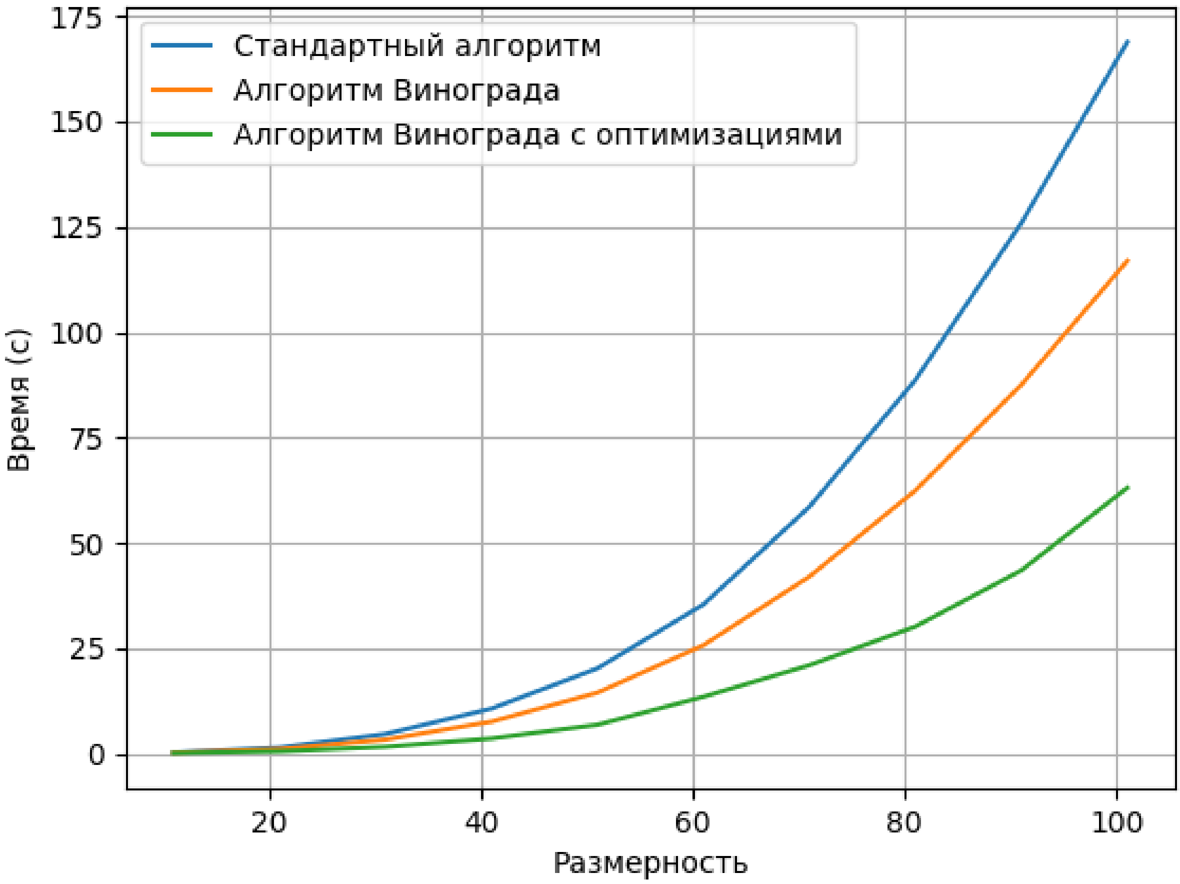

Рис. 4.2: Временные характеристики на нечетных размерах матриц

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

В рамках данной лабораторной работы:

1. были изучены и реализованы алгоритмы перемножения матриц: обычный, Копперсмита-Винограда;
2. был произведен анализ трудоёмкости алгоритмов на основе теоретических расчетов и выбранной модели вычислений;
3. был сделан сравнительный анализ алгоритмов на основе экспериментальных данных;
4. был подготовлен отчет по проделанной работе.

Литература

[1] Coppersmith D., Winograd S. Matrix multiplication via arithmetic progressions // Journal of Symbolic Computation. 1990. no. 9. P. 251–280.

[2] Group-theoretic Algorithms for Matrix Multiplication / H. Cohn, R. Kleinberg, B. Szegedy et al. // Proceedings of the 46th Annual Symposium on Foundations of Computer Science. 2005. October. P. 379–388.

[3] Погорелов Дмитрий Александрович Таразанов Артемий Михайлович Волкова Лилия Леонидовна. Оптимизация классического алгоритма Винограда для перемножения матриц // Журнал №1. 2019. Т. 49.