МГТУ им. Баумана

Лабораторная работа №2

По курсу: "Анализ алгоритмов"

Исследование алгоритмов умножения матриц

Студент: Аникин И. А., ИУ7-51Б

Преподаватель: Волкова Л.Л.

Москва, 2020

Содержание

Введение

В современном мире очень удобно представлять информацию в виде таблиц. Существует специальный математический объект, который называется матрица.

Матрица - это прямоугольная таблица чисел. Для матриц определены операции сложения и умножения. Операция сложения определяется тривиально и применима только к матрицам одинакового размера. Умножение матриц - операция не имеющая аналогов для чисел. Умножение матриц не является коммутативным (AB != BA), можно перемножать матрицы не совпадающего размера.

1. Аналитическая часть

Рассмотрим цель и задачи данной лабораторной работы.

Цель лабораторной работы - разработать и провести сравнительный анализ алгоритмов умножения матриц.
Для достижения данной цели можно выделить следующие задачи:

- разработать классический алгоритм, алгоритм Винограда и улучшенный алгоритм Винограда;
- реализовать разработанные алгоритмы;
- провести тестирование разработанных алгоритмов по методу черного ящика;
- провести замеры времени работы реализаций алгоритмов на наборе данных;
- провести теоретическую оценку трудоемкости алгоритмов в худшем и лучшем случаях;
- провести сравнительный анализ временной эффективности разработанных алгоритмов.

Произведение матриц - это математическая операция, которая определяется следующим образом:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1q} \\ b_{21} & b_{22} & \cdots & b_{2q} \\ \cdots & \cdots & \cdots & \cdots \\ b_{n1} & b_{n2} & \cdots & b_{nq} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1q} \\ c_{21} & c_{22} & \cdots & c_{2q} \\ \cdots & \cdots & \cdots & \cdots \\ c_{m1} & c_{m2} & \cdots & c_{mq} \end{pmatrix},$$

где $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} (i = 1, 2, \ldots, m; j = 1, 2, \ldots, q)$ [1].

Эта операция применима только к прямоугольным матрицам, у которых количество строк первой матрицы равно количеству столбцов второй.

Алгоритм Винограда - улучшенный алгоритм умножения матриц, который уменьшает свою вычислительную сложность засчет оптимизации вычисления произведения строки и столбца.
Пусть $u = (u_1, u_2, u_3, u_4)$ строка матрицы A, а $v = (v_1, v_2, v_3, v_4)$ столбец матрицы B.
В данном алгоритме вместо классического умножения $u_1v_1 + u_2v_2 + u_3v_3 + u_4v_4$ произведение вычисляется следующим образом:
$(u_1 + v_2)(u_2 + v_1) + (u_3 + v_4)(u_4 + v_3) - u_1u_2 - u_3u_4 - v_1v_2 - v_3v_4$

Если количество столбцов нечетно, то это является худшим случаем и к произведению добавляется множитель $+u_5v_5$.

Такой способ содержит меньшее количество умножений, а значит является более быстрым при вычислении на ЭВМ.

2. Конструкторская часть

2.1. Требования к ПО

Разработанное должно обладать следующей функциональностью:

- считывание двух матриц;
- выбор метода умножения;
- вывод результата умножения;
- осуществление замеров процессорного времени работы реализаций алгоритмов.

2.2. Разработка алгоритмов

В ходе данной лабораторной работы будет реализовано 3 различных алгоритма умножения матриц:

- классический алгоритм;
- алгоритм Винограда;
- улучшенный алгоритм Винограда.

Улучшенный алгоритм Винограда использует следующие оптимизации:

- замена $c[i][j] = c[i][j]+$ на $c[i][j]+ =$;
- замена $2k$ на $k+ = 2$ в цикле;
- вычисление отрицательных чисел в массиве multA и multB.

На рисунке 3 в приложении 1 представлена схема классического алгоритма умножения.

На рисунке 4 в приложении 2представлена схема алгоритма Винограда.

На рисунке 5 в приложении 3 представлена схема улучшенного алгоритма Винограда.

2.3. Определение трудоемкости алгортимов

Определим трудоемкость каждого алгоритма. Будем использовать следующую модель вычислений:

- все базовые операции (+, -, *, /, [], =, += , -=, >, <, ==, !=) стоят 1;
- проверка условия стоит 1;
- цикл стоит $f_{initialization} + f_{condition} + n(f_{condition} + f_{increment} + f_{body})$, где n - количество итераций.

2.3.1. Классический алгоритм

Рассмотрим классический алгоритм.
$f_{classic} = 3 + 2 + m(2 + f_1)$, где f_1 - трудоемкость тела цикла i,
$f_1 = 2 + q(2 + f_2)$, где f_2 - трудоемкость тела цикла j,
$f_2 = 2 + n(2 + f_3)$, где f_3 - трудоемкость тела цикла k,
$f_3 = 8$
$f_{classic} = 10mnq + 4mq + 4m + 5$

2.3.2. Алгоритм Винограда

Рассмотрим алгоритм Винограда.
$f_{Winograd} = 3+2+m(2+f_1)+2+q(2+f_3)+2+m(2+f_5)$, где f_1 - трудоемкость тела цикла i,
$f_1 = 2 + \frac{n}{2}(4 + f_2)$, где f_2 - трудоемкость тела цикла k,
$f_2 = 9$
$f_3 = 2 + \frac{n}{2}(4 + f_4)$, где f_4 - трудоемкость тела цикла k,
$f_4 = 9$
$f_5 = 2 + q(2 + f_6)$
$f_6 = 7 + 2 + \frac{n}{2}(3 + f_7) + f_8$, где $f_8 -$
$f_7 = 23$
$f_8^{\vee} = 2$
$f_8^{\wedge} = 2 + 13$
$f_{Winograd}^{\vee} = 13mnq + 13mq + 6.5nq + 6.5mn + 8m + 4q + 9$
$f_{Winograd}^{\wedge} = 13mnq + 26mq + 6.5nq + 6.5mn + 8m + 4q + 9$

2.3.3. Улучшенный алгоритм Винограда

Рассмотрим улучшенный алгоритм Винограда.
$f_{opt.Winograd} = 3+2+m(2+f_1)+2+q(2+f_3)+2+m(2+f_5)$, где f_1 - трудоемкость

тела цикла i,
$f_1 = 4 + \frac{n}{2}(4 + f_2)$, где f_2 - трудоемкость тела цикла k,
$f_2 = 8$
$f_3 = 4 + \frac{n}{2}(4 + f_4)$, где f_4 - трудоемкость тела цикла k,
$f_4 = 8$
$f_5 = 2 + q(2 + f_6)$
$f_6 = 6 + 4 + \frac{n}{2}(4 + f_7) + f_8$, где $f_8 -$
$f_7 = 16$
$f_8^{\vee} = 2$
$f_8^{\wedge} = 2 + 10$
$f_{opt.Winograd}^{\vee} = 10mnq + 14mq + 6nq + 6mn + 10m + 4q + 9$
$f_{opt.Winograd}^{\wedge} = 10mnq + 24mq + 6nq + 6mn + 10m + 4q + 9$

3. Технологическая часть

В ходе выполнения лабораторной работы были реализованы следующие модули:

- модуль matrices.py, в котором реализованы алгоритмы умножения матриц;
- модуль test.py, предназначенный для тестирования реализованных алгоритмов;
- модуль analysis.py, предназначенный для замеров времени выполнения алгоритмов на различных входных данных и построения графиков;
- модуль main.py, предоставляющий пользователю возможности умножения матриц выбранным методом.

В качестве языка разработки был выбран язык python, т. к. он имеет ряд преимуществ для достижения цели лабораторной работы:

- высокая скорость разработки программ;
- простой синтаксис языка;
- наличие большого количества библиотек, что позволяет легко анализировать полученные результаты и проводить тестирование.

3.1. Листинги алгоритмов

На листинге 1 приведен алгоритм классического умножения.

Листинг 1: Алгоритм классического умножения

```
def classic_multiply(a, b, c):
    m = len(a)
    n = len(b)
    q = len(b[0])

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

На листинге 2 приведен алгоритм Винограда.

Листинг 2: Алгоритм Винограда

```
def winograd(a, b, c):
    m = len(a)
    n = len(b)
    q = len(b[0])

    mult_a = [0] * m
    for i in range(m):
        mult_a.append(0)
        for k in range(n//2):
            mult_a[i] += a[i][2*k] * a[i][2*k + 1]

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

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

На листинге 3 приведен алгоритм сортировки вставками.

Листинг 3: Улучшенный алгоритм Винограда

```
def optimised_winograd(a, b, c):
    m = len(a)
    n = len(b)
    q = len(b[0])

    mult_a = [0] * m
    for i in range(m):
        for k in range(0, n - n%2, 2):
            mult_a[i] += -a[i][k] * a[i][k + 1]

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

    for i in range(m):
        for j in range(q):
            c[i][j] = mult_a[i] + mult_b[j]
            for k in range(0, n - n%2, 2):
                c[i][j] += (a[i][k] + b[k+1][j]) * (a[i][k+1] + b[k][
                    j])
            if n % 2:
                c[i][j] += a[i][-1] * b[-1][j]
    return c
```

3.2. Тестирование

Тестирование разработанных алгоритмов проводилось на наборе тестов, который представлен в таблице 1.

Таблица 1: Набор тестов

Матрица A	Матрица B	Ожидаемые выходные данные
$\begin{matrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{matrix}$	$\begin{matrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{matrix}$	$\begin{matrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{matrix}$
$\begin{matrix}1 & 2 & 3\\ 3 & 2 & 1\\ 2 & 3 & 1\end{matrix}$	$\begin{matrix}1 & 1 & 1\\ 2 & 2 & 2\\ 3 & 3 & 3\end{matrix}$	$\begin{matrix}14 & 14 & 14\\ 10 & 10 & 10\\ 11 & 11 & 11\end{matrix}$
$\begin{matrix}3 & 7 & -2\\ -11 & 7 & 3\end{matrix}$	$\begin{matrix}1 & 0 & 11\\ -1 & 7 & 17\\ 2 & 6 & -14\end{matrix}$	$\begin{matrix}-8 & 37 & 180\\ -12 & 67 & -44\end{matrix}$
$\begin{matrix}3 & 7 & -2 & 0\\ -11 & 7 & 3 & 1\end{matrix}$	$\begin{matrix}1 & 0 & 11\\ -1 & 7 & 17\\ 2 & 6 & -14\\ -3 & -4 & -22\end{matrix}$	$\begin{matrix}-8 & 37 & 180\\ -15 & 63 & -66\end{matrix}$

Все тесты пройдены успешно.

4. Исследовательская часть

4.1. Сравнение временных характеристик алгоритмов

Проведем исследование временных характеристик реализованных алгоритмов.
Ниже представлены графики зависимости времени работы алгоритмов на различных входных данных. На вход подаются две матрицы.
На рисунке 1 изображено время работы в лучшем случае, где количество столбцов первой матрицы и количество столбцов второй - четное.

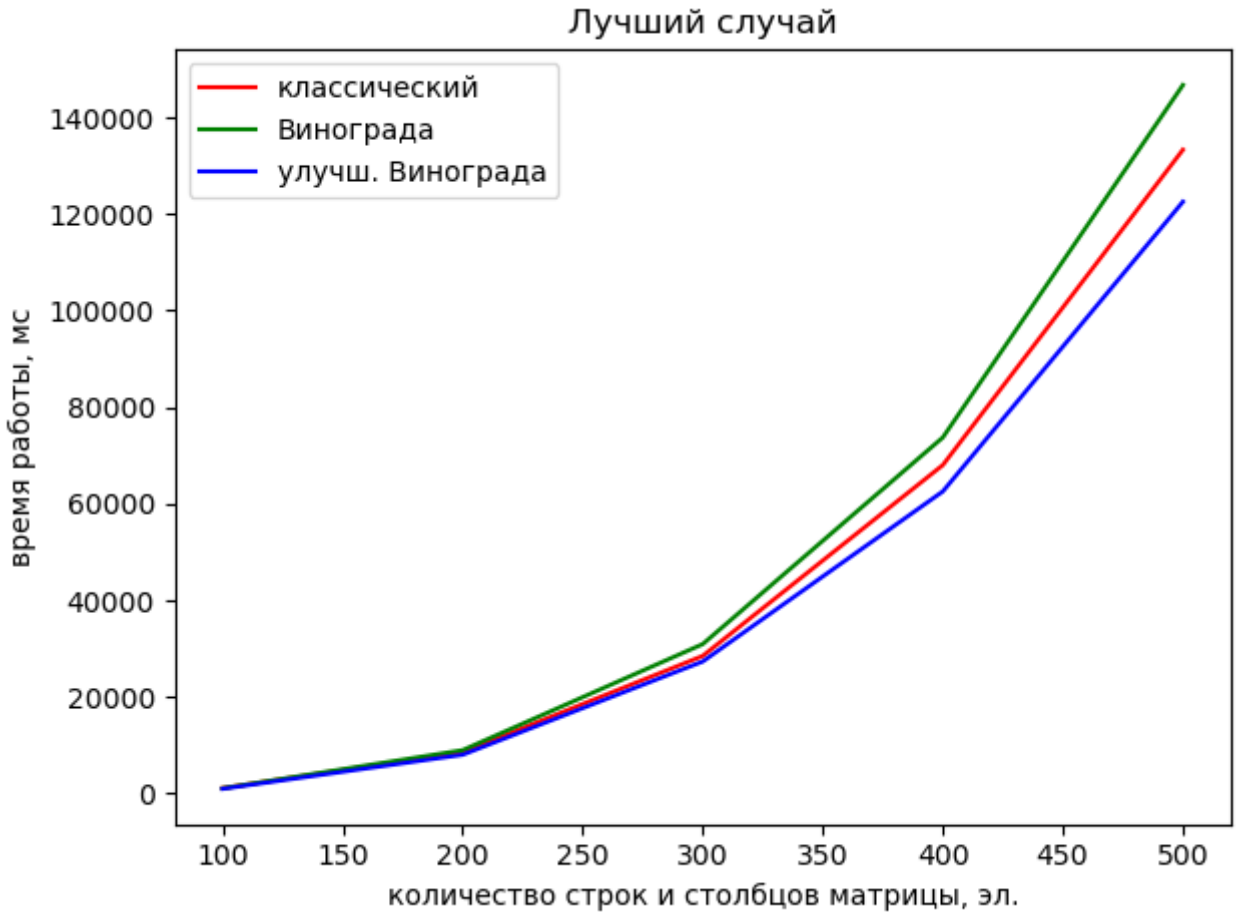

Рис. 1: Лучший случай для реализованных алгоритмов

На рисунке 2 изображено время работы в худшем случае, где количество столбцов первой матрицы и количество столбцов второй - нечетное.

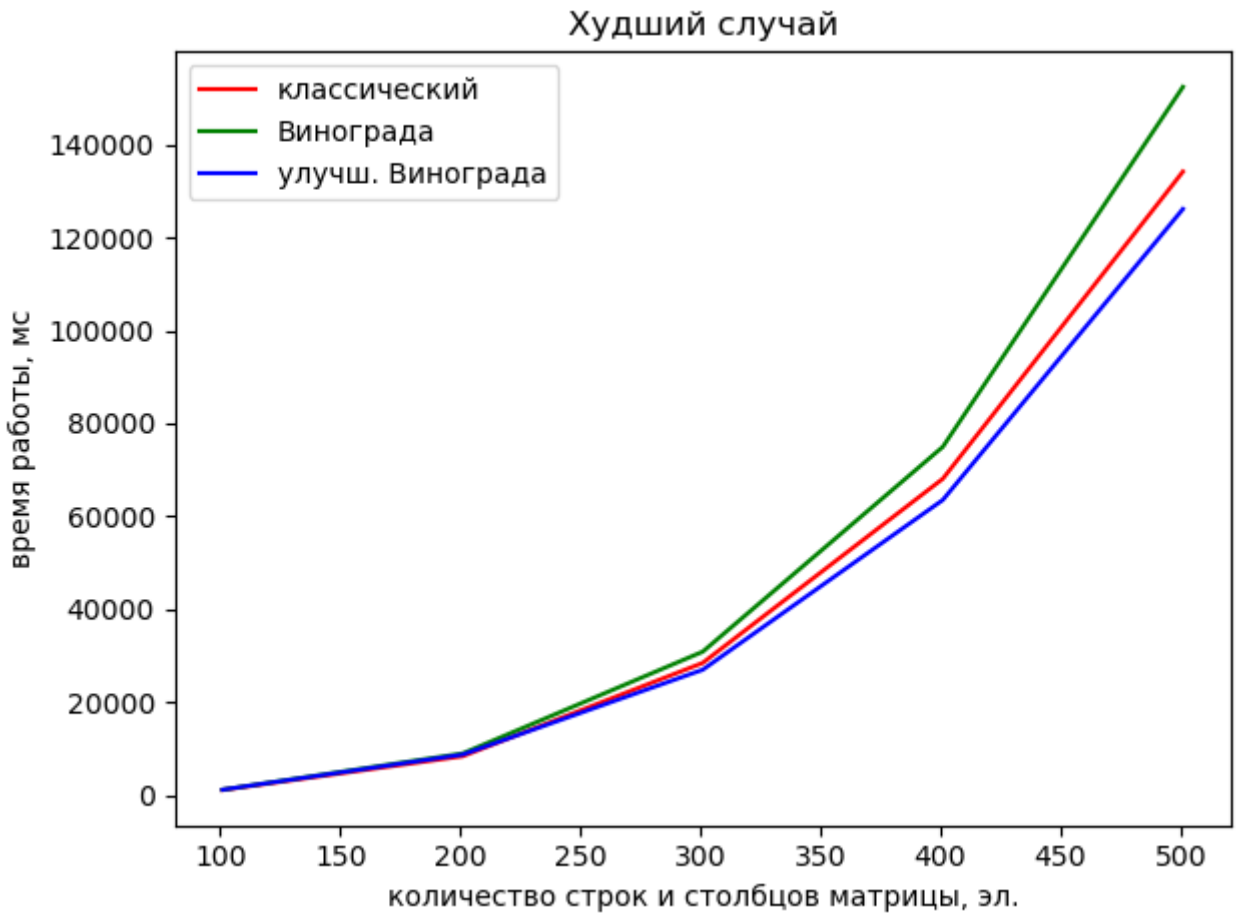

Рис. 2: Худший случай для реализованных алгоритмов

Исходя из данных графиков, к применению рекомендуется улучшенный алгоритм Винограда, т. к. он имеет наименьшее время выполнения. Этот алгоритм использует численные оптимизации, для уменьшения трудоемкости по сравнению с неоптимизированным алгоритмом Винограда. Классический алгоритм является более быстрым, чем неоптимизированный алгоритм Винограда, несмотря на большее количество умножений.

Полученные графики подтверждают теоретические оценки трудоемкости.

Заключение

В ходе выполнения лабораторной работы были выполнены следующие задачи:

- разработаны алгоритмы: классический, Винограда и улучш. Винограда;
- реализованы разработанные алгоритмы;
- проведено тестирование разработанных алгоритмов по методу черного ящика;
- проведены замеры времени работы реализаций алгоритмов на наборе данных;
- проведена теоретическую оценку трудоемкости алгоритмов;
- проведен сравнительный анализ временной эффективности разработанных алгоритмов.

Цель лабораторной работы достигнута, все задачи выполнены.

По итогам достигнутых задач можно сделать следующие выводы: самым быстрым является оптимизированный алгоритм Винограда, т. к. он является самым быстрым в лучшем и худшем случаях, потому что он совершает меньшее количество умножений и использует оптимизации для уменьшения вычислительной сложности.

Даны рекомендации по использованию исследованных алгоритмов.

Список литературы

[1] Корн Г., Корн Т. Алгебра матриц и матричное исчисление // Справочник по математике. — 4-е издание. — М: Наука, 1978. — С. 392.

Приложения

Приложение 1

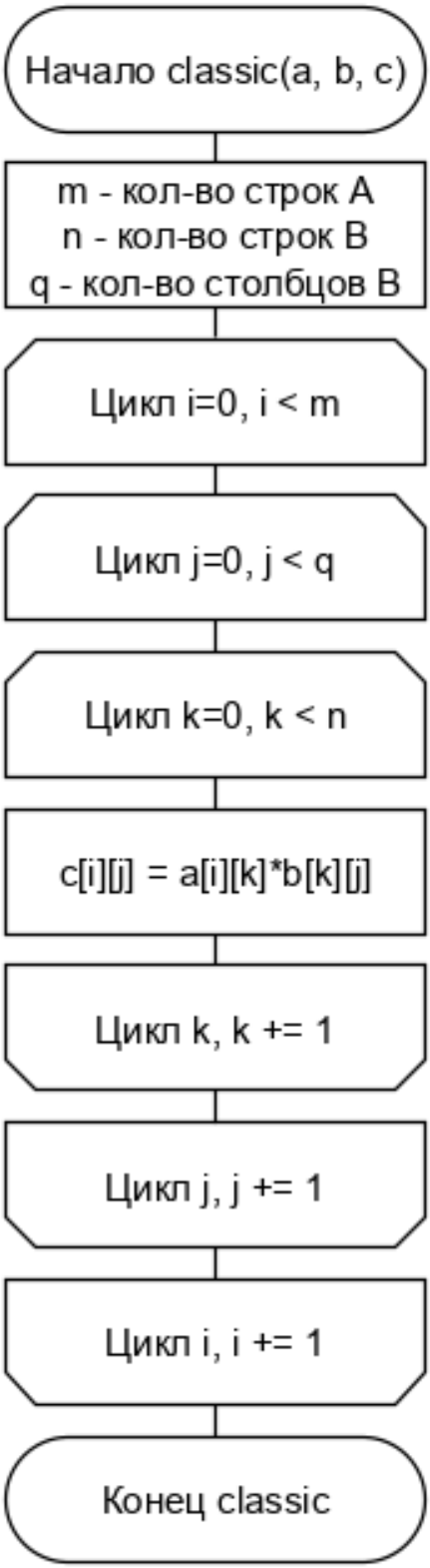

Рис. 3: Схема классического алгоритма умножения

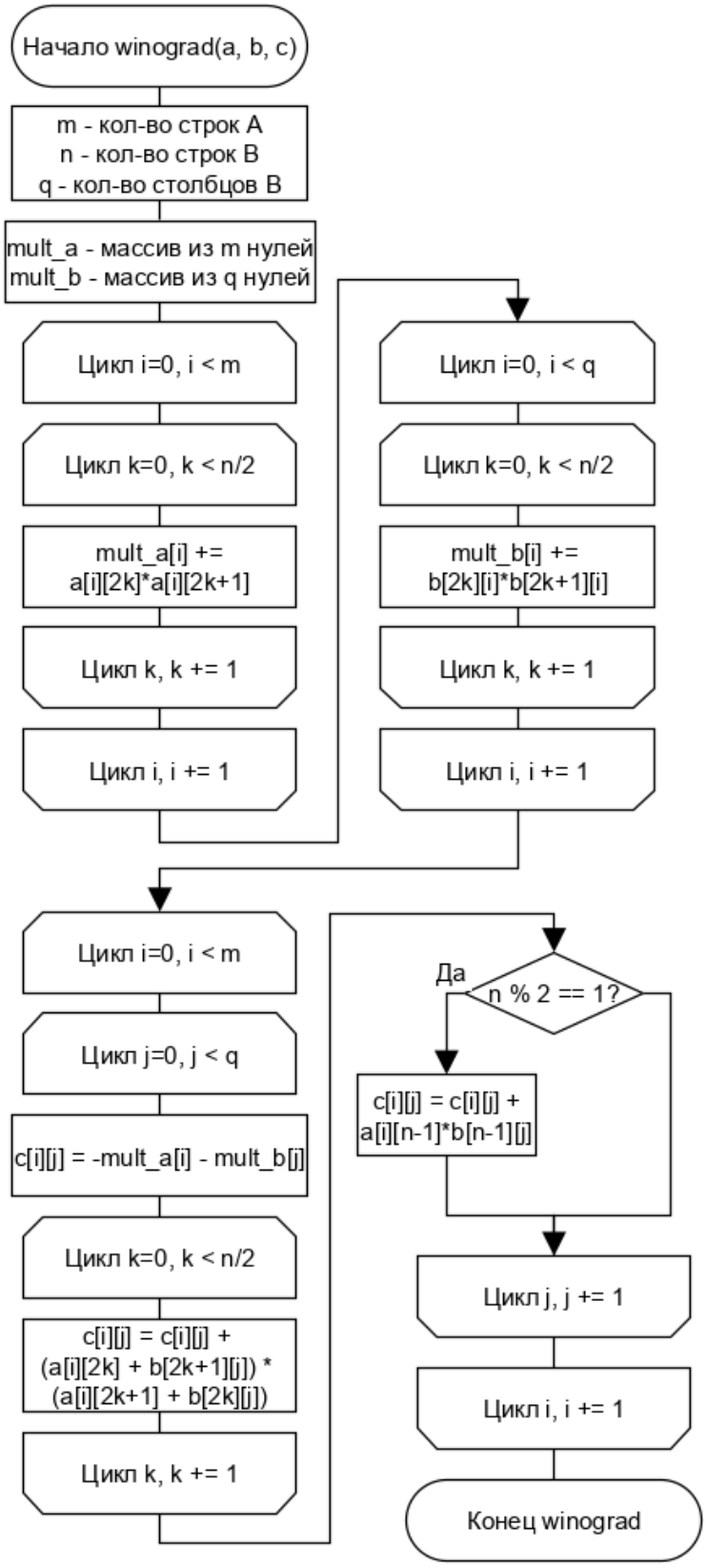

Рис. 4: Схема алгоритма Винограда

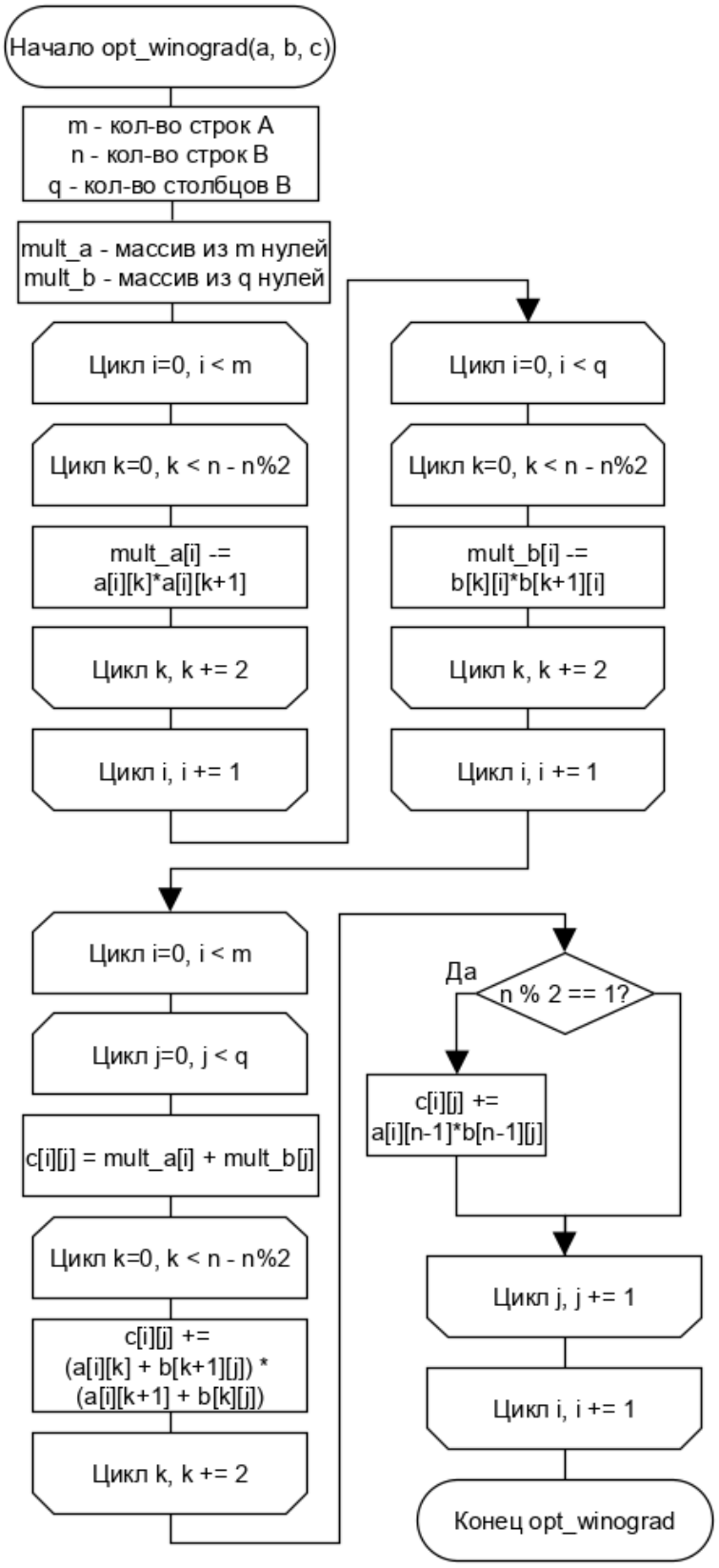

Рис. 5: Схема улучшенного алгоритма Винограда