**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное автономное 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»

# Лабораторная работа № 1 по дисциплине «Анализ алгоритмов»

**Тема** Алгоритмы умножения матриц

**Студент** Доколин Г. А.

**Группа** ИУ7-52Б

**Преподаватели** Волкова Л. Л.

Москва, 2025

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

В рамках данной лабораторной работы анализируются алгоритмы умножения матриц: стандартный и метод Винограда, который рассматривается в базовой и оптимизированной модификациях. Целью исследования является сравнительный анализ производительности и особенностей реализации указанных алгоритмов. Для достижения цели были поставлены следующие задачи:

1) Изложить математические принципы, лежащие в основе классического метода умножения матриц [**?**, с. 8] и алгоритма Винограда [**?**].
2) Представить схемы, описывающие логику работы трёх рассматриваемых алгоритмов: стандартного, базового Винограда и его оптимизированной версии.
3) Сформулировать вычислительную модель и провести в её рамках теоретический анализ трудоёмкости трёх изучаемых алгоритмов умножения матриц.
4) Разработать программные реализации трёх алгоритмов умножения.
5) Провести эксперименты по измерению процессорного времени, затрачиваемого выполнением реализованных алгоритмов.
6) Выполнить сравнительную характеристику алгоритмов на основе данных экспериментов.

# 1 Аналитическая часть

В данном разделе излагаются математические принципы, лежащие в основе двух методов умножения матриц: классического и алгоритма Винограда.

## 1.1 Матрица

Матрица [**?**, с. 5] — прямоугольная таблица, образованная из элементов некоторого множества и состоящая из $m$ строк и $n$ столбцов. Матрица обозначается следующим образом:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \tag{1.1}$$

где $a_{ij}$ — элементы матрицы ($i = \overline{1,m}; j = \overline{1,n}$); $i$ — номер строки; $j$ — номер столбца; $m \times n$ — размер матрицы.

При работе с матрицами выделяют три основные операции:

1) сложение, данное действие выполняется для двух матриц одинакового размера;

2) умножение, данное действие выполняется для двух таких матриц, что у первой матрицы число столбцов должно быть равно числу строк второй матрицы;

3) транспонирование, данное действие выполняется для матриц любых размеров.

## 1.2 Стандартный алгоритм

Пусть даны две матрицы. Матрица $A$, размерностью $m \times n$ и матрица $B$, размерностью $n \times k$:

$$A_{m\times n} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, B_{n\times k} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1k} \\ b_{21} & b_{22} & \cdots & b_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ b_{n1} & b_{n2} & \cdots & b_{nk} \end{pmatrix}, \tag{1.2}$$

тогда матрица $C$ будет иметь размер $m \times k$:

$$C_{m\times k} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1k} \\ c_{21} & c_{22} & \cdots & c_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ c_{m1} & c_{m2} & \cdots & c_{mk} \end{pmatrix}, \tag{1.3}$$

где

$$c_{ij} = \sum_{t=1}^{n} a_{it} \cdot b_{tj}, \ i = \overline{1,m}; j = \overline{1,k} \tag{1.4}$$

## 1.3 Алгоритм Винограда

В формуле (1.4) можно заметить, что каждый элемент в результирующей матрице равен скалярному произведению соответствующих строк и столбцов исходных матриц. Пусть даны два вектора $\overrightarrow{V} = (v_1, v_2, \ldots, v_n)$ и $\overrightarrow{U} = (u_1, u_2, \ldots, u_n)$, тогда их скалярное произведение будет записано в виде:

$$\overrightarrow{V} \cdot \overrightarrow{U} = v_1 \cdot u_1 + v_2 \cdot u_2 + \cdots + v_n \cdot u_n \tag{1.5}$$

Это равенство (1.5) можно представить в виде выражения для чётных $n$ (1.6) и в виде выражения для нечётных $n$ (1.7) :

$$\begin{gathered}\overrightarrow{V} \cdot \overrightarrow{U} = (v_1 + u_2) \cdot (v_2 + u_1) + \cdots (v_{n-1} + u_n) \cdot (v_n + u_{n-1}) - \\ - v_1 \cdot v_2 - u_1 \cdot u_2 - \cdots - v_{n-1} \cdot v_n - u_{n-1} \cdot u_n\end{gathered} \tag{1.6}$$

$$\begin{gathered}\overrightarrow{V} \cdot \overrightarrow{U} = (v_1 + u_2) \cdot (v_2 + u_1) + \cdots (v_{n-2} + u_{n-1}) \cdot (v_{n-1} + u_{n-2}) - \\ - v_1 \cdot v_2 - u_1 \cdot u_2 - \cdots - v_{n-2} \cdot v_{n-1} - u_{n-2} \cdot u_{n-1} + v_n \cdot u_n\end{gathered} \tag{1.7}$$

Произведения вида $q_i \cdot q_{i+1}$ можно вычислить заранее для каждой из двух матриц: для первой матрицы посчитать произведения элементов для каждой строки, для второй — для каждого столбца.

## 1.4 Оптимизированный алгоритм Винограда

В алгоритме Винограда можно дополнительно сократить число арифметических операций за счёт вынесения части вычислений за пределы вложенных циклов. Оптимизация основана на двух ключевых приёмах:

1) **Предварительное вычисление постоянных выражений.** Значения, зависящие только от размеров матриц (например, $N-1$, $\lfloor N/2 \rfloor$), вычисляются один раз перед началом умножения. Это избавляет от повторных вычислений в теле циклов.
2) **Буферизация промежуточных результатов.** При заполнении массивов дополнительно формируются вспомогательные буферы:
   - — для каждой строки первой матрицы сохраняются суммы произведений пар элементов;
   - — для каждого столбца второй матрицы аналогично сохраняются накопленные значения.

Использование буферов позволяет уменьшить количество умножений и сложений при вычислении каждого элемента результирующей матрицы.

Таким образом, оптимизированный алгоритм Винограда, сохраняя идею скалярного произведения строк и столбцов, уменьшает общее количество выполняемых операций. Это делает его более эффективным при работе с матрицами больших размеров.

## Вывод

В аналитической части были рассмотрены математические основы алгоритма Винограда и стандартного алгоритма умножения матриц.

# 2 Конструкторская часть

В данной главе представлены требования к разрабатываемому программному обеспечению, приведены схемы алгоритмов и введена модель вычислений для проведения теоретической оценки трудоёмкости.

## 2.1 Требования к программе

Для разрабатываемой программы определены следующие задачи:

1) реализовать интерфейс выбора операций для пользователя;
2) обеспечить возможность работы программы в двух режимах: одиночное выполнение и массовое измерение времени выполнения;
3) в режиме одиночного запуска предусмотреть:
    - ввод размеров и элементов двух матриц;
    - проверку корректности введённых данных;
    - проверку совместимости матриц для операции умножения;
4) в режиме массового измерения времени фиксировать затраченное процессорное время и выводить результаты в табличном виде.

## 2.2 Разработка алгоритмов

На рисунке 2.1 представлена схема стандартного алгоритма перемножения матриц. Рисунки 2.2–2.3 демонстрируют структуру алгоритма Винограда.

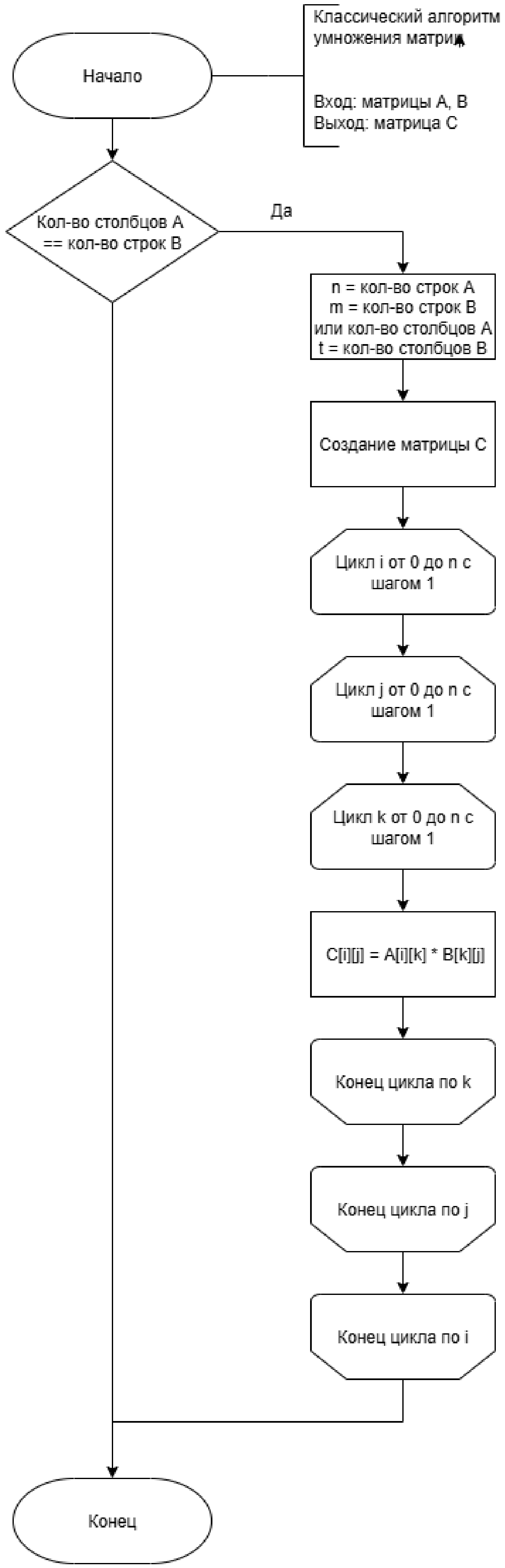


Рисунок 2.1 — Схема стандартного алгоритма умножения матриц

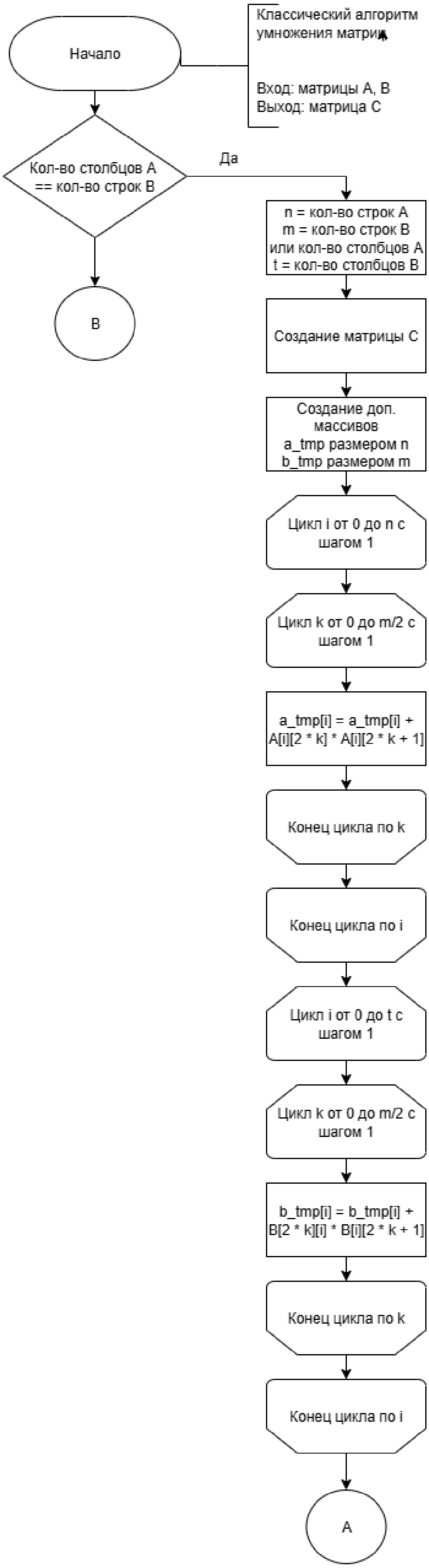


Рисунок 2.2 — Схема алгоритма Винограда (часть 1)

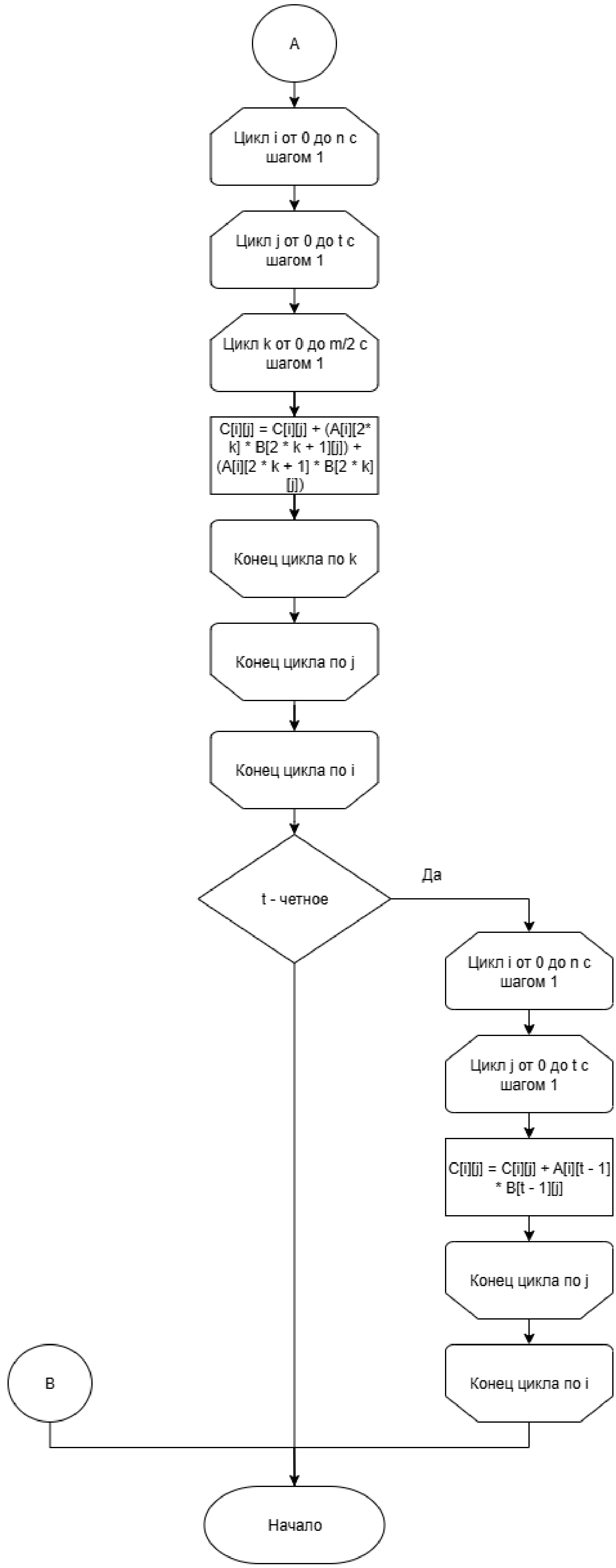


Рисунок 2.3 — Схема алгоритма Винограда (часть 2)

Для алгоритма Винограда были предложены следующие пути оптимизации:

1) использование предварительно вычисленных значений (например, $N-1$, $N/2$);

2) хранение промежуточных сумм в буферах при заполнении массивов и результирующей матрицы.

На рисунках 2.4–2.5 представлена схема оптимизированного алгоритма Винограда.

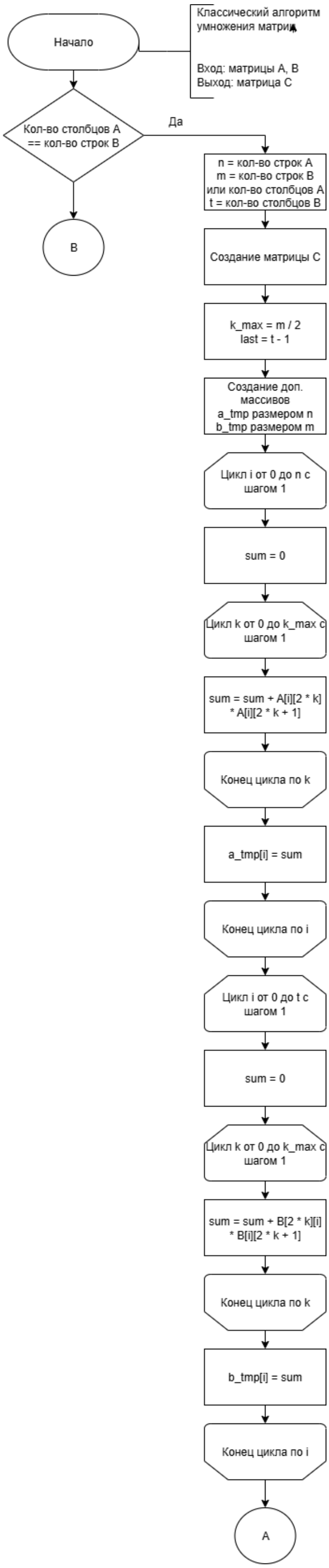


Рисунок 2.4 — Схема оптимизированного алгоритма Винограда (часть 1)

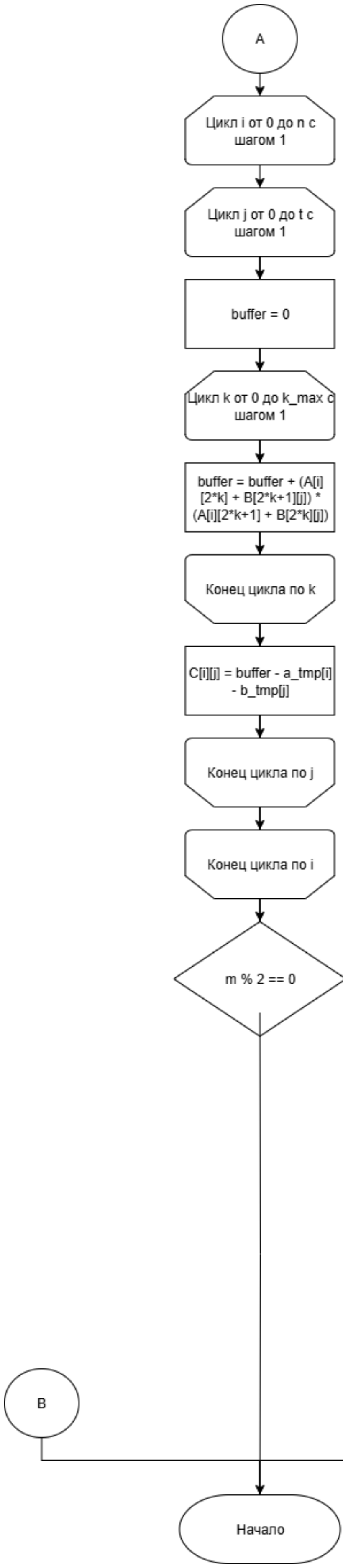


Рисунок 2.5 — Схема оптимизированного алгоритма Винограда (часть 2)

## 2.3 Модель вычислений

Для анализа трудоёмкости алгоритмов вводится следующая модель:

1) стоимость операций $=, ==, != , >, <, >=, <=, \&\&, ||, \&=, |=, +=, -=, +, -, []$, $++$, $--$, $<<$, $>>$ принимается равной 1;

2) стоимость операций $\cdot, /, \cdot=, /=, \%, \%=$ принимается равной 2;

3) трудоёмкость условного оператора вида *if (Условие) {Блок X} else {Блок Y}*, где вычисление условия и блоков X и Y обозначено как $c_{cond}$, $c_x$, $c_y$, оценивается по формуле (2.1):

$$c_{if} = c_{cond} + \begin{cases} \min(c_x, c_y), & \text{лучший случай} \\ \max(c_x, c_y), & \text{худший случай} \end{cases} \tag{2.1}$$

4) трудоёмкость цикла вида *for (Начало, Условие, Приращение) {тело}*, где трудоёмкость начальной установки, условия, приращения и тела соответственно равны $c_{start}$, $c_{cond}$, $c_{step}$, $c_{body}$, вычисляется по формуле (2.2), где $M$ — число итераций:

$$c_{for} = c_{start} + c_{cond} + M \cdot (c_{body} + c_{step} + c_{cond}) \tag{2.2}$$

## 2.4 Расчёт трудоёмкости алгоритмов

### 2.4.1 Стандартный алгоритм

Трудоёмкость стандартного алгоритма (рис. 2.1) складывается из трёх вложенных циклов и набора операций во внутреннем цикле. Итоговая трудоёмкость рассчитывается по формуле (2.3):

$$f_{std} = 2 + P \cdot (2 + 2 + Q \cdot (2 + 2 + R \cdot (2 + 12))) = 14PQR + 4PQ + 4P + 2 \approx 14PQR \tag{2.3}$$

где $P$, $Q$, $R$ — размеры соответствующих измерений исходных матриц.

### 2.4.2 Алгоритм Винограда

Алгоритм Винограда состоит из четырёх этапов:

1) **Заполнение массива *mulH***: Для каждой строки A суммируются произведения пар элементов.

$$f_1 = 2 + M \cdot (2 + 1 + 4 + \frac{N}{2} \cdot (4 + 1 + 6 + 2 + 3 \cdot 2)) = \frac{19}{2}MN + 7M + 2 \approx \frac{19}{2}MN \tag{2.4}$$

2) **Заполнение массива *mulV***: Для каждого столбца B вычисляются промежуточные

суммы.

$$f_2 = 2 + K \cdot (2 + 1 + 4 + \frac{N}{2} \cdot (4 + 1 + 6 + 2 + 3 \cdot 2)) = \frac{19}{2}KN + 7K + 2 \approx \frac{19}{2}KN \quad (2.5)$$

3) **Заполнение матрицы *C***: Используются заранее посчитанные суммы. Каждое вычисление ячейки включает доступы к буферам, сложения, умножения и корректировку.

$$f_3 = 2 + M \cdot (2 + 2 + K \cdot (2 + 4 + 1 + 2 + 4 + \frac{N}{2} \cdot (4 + 12 + 1 + 5 + 5 \cdot 2))) = 16MNK + 13MK + 4M + 2 \approx 16 \quad (2.6)$$

4) **Корректировка для нечётного N**:

$$f_{if} = \begin{cases} 3, & N \text{ — чётное} \\ 16MK + 4M + 5, & N \text{ — нечётное} \end{cases} \quad (2.7)$$

Общая трудоёмкость:

$$f_{total} = \begin{cases} 16MNK + 13MK + \frac{19}{2}(MN + KN) + 11M + 7K + 9, & N \text{ — чётное} \\ 16MNK + 29MK + \frac{19}{2}(MN + KN) + 15M + 7K + 11, & N \text{ — нечётное} \end{cases} \quad (2.8)$$

### 2.4.3 Оптимизированный алгоритм Винограда

В оптимизированном алгоритме Винограда введены улучшения: предварительное вычисление промежуточных значений ($N - 1$, $N/2$ и др.) и использование буферов для накопления промежуточных сумм при заполнении массивов и результирующей матрицы. Эти меры сокращают количество арифметических операций и обращений к памяти.

Трудоёмкость этапов алгоритма (см. рисунки 2.4–2.5) оценивается следующим образом:

1) **Заполнение массива *mulH*.** Сумма произведений элементов строк первой матрицы вычисляется заранее и сохраняется в буфер *tmp_a*. Трудоёмкость этого этапа задаётся формулой:

$$f_1 = 2 + P \cdot \left(2 + 1 + 2 + \frac{Q}{2} \cdot (2 + 1 + 6 + 1 + 1 + 1 \cdot 2)\right) = \frac{13}{2}PQ + 5P + 2 \approx \frac{13}{2}PQ \quad (2.9)$$

2) **Заполнение массива *mulV*.** Аналогично формируется буфер *tmp_b* для столбцов второй матрицы:

$$f_2 = 2 + R \cdot \left(2 + 1 + 2 + \frac{Q}{2} \cdot (2 + 1 + 6 + 1 + 1 + 1 \cdot 2)\right) = \frac{13}{2}RQ + 5R + 2 \approx \frac{13}{2}RQ \quad (2.10)$$

3) **Вычисление результирующей матрицы *C*.** Основные вычисления включают использование предварительно накопленных сумм из буферов и коррекцию для нечётного

$Q$:

3.1) Для чётного $Q$:

$$f_{31} = 3 + 2 + M \cdot \Big(2 + 2 + K \cdot (2 + 4 + 1 + 2 + 2 + \frac{Q}{2} \cdot (2 + 12 + 1 + 3 + 2 + 1 \cdot 2))\Big) = 11PQR + 11PR + 4P + 5 \approx 11PQR \quad (2.11)$$

3.2) Для нечётного $Q$:

$$f_{32} = 3 + 2 + M \cdot \Big(2 + 2 + K \cdot (2 + 8 + 1 + 4 + 1 + 1 \cdot 2 + \frac{Q}{2} \cdot (2 + 12 + 1 + 3 + 2 + 1 \cdot 2))\Big) = 11PQR + 18PR + 4P + 5 \approx 11PQR \quad (2.12)$$

Таким образом, суммарная трудоёмкость оптимизированного алгоритма Винограда вычисляется по формуле:

$$f_{total} = \begin{cases} 11PQR + 11PR + \frac{13}{2}(PQ + RQ) + 9P + 5R + 7, & \text{л.с. } (Q \text{ — чётное}) \\ 11PQR + 18PR + \frac{13}{2}(PQ + RQ) + 9P + 5R + 7, & \text{х.с. } (Q \text{ — нечётное}) \end{cases} \quad (2.13)$$

**Вывод.** Предварительное вычисление промежуточных сумм и использование буферов позволяют сократить число операций. Оптимизированный алгоритм Винограда улучшает трудоёмкость примерно в 1.27 раза по сравнению с классическим алгоритмом и в 1.45 раза по сравнению с исходной версией алгоритма Винограда.