*Федеральное государственное бюджетное образовательное учреждениевысшего профессионального образования*

|  |  |
| --- | --- |
| Gerb-BMSTU_01 | ***«Московский государственный технический университет  имени Н.Э. Баумана (национальный исследовательский институт)»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

**Отчёт**

**по лабораторной работе 2**

**Дисциплина: Анализ Алгоритмов**

**Тема лабораторной работы работы: Трудеомкость алгоритмов умножения матриц**

Студентки гр. ИУ7-51б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сушина А.Д.**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Волкова Л.Л.**

(Подпись, дата) (И.О. Фамилия)

Москва, 2019г

# Введение

1. Умножение матриц — это один из базовых алгоритмов, который широко применяется в различных численных методах, и в частности в алгоритмах машинного обучения.

Произведением матрицы *A**m*×*n* на матрицу *B**n*×*k* называется матрица *C**m*×*k* такая, что элемент матрицы *C*, стоящий в *i*-ой строке и *j*-ом столбце, т.е. элемент *c**i**j*, равен сумме произведений элементов *i*-ой строки матрицы *A* на соответствующие элементы *j*-ого столбца матрицы *B*.

**Алгоритм Копперсмита-Винограда**— алгоритм умножения квадратных матриц, предложенный в 1987 году Д. Копперсмитом и Ш. Виноградом. Алгоритм Копперсмита—Винограда, с учетом серии улучшений и доработок в последующие годы, обладает лучшей асимптотикой среди известных алгоритмов умножения матриц.

На практике алгоритм Копперсмита—Винограда не используется, так как он имеет очень большую константу пропорциональности и начинает выигрывать в быстродействии у других известных алгоритмов только для матриц, размер которых превышает память современных компьютеров.

**Цель** лабораторной работы изучить трудоемкости алгоритмов умножения матриц и способы оптимизации этих алгоритмов.

**Задачи:**

* Изучение алгоритмов стандартного умножения матриц и алгоритма Винограда.
* Получение практических навыков при реализации стандартного алгоритма умножения матриц и алгоритма Винограда.
* Оптимизация алгоритма Винограда тремя способами.
* Подсчет трудоемкости каждой из реализаций.
* экспериментальное подтверждение различий во временнóй эффективности работы оптимизированных и неоптимизированного алгоритмов Винограда
* описание и обоснование полученных результатов в отчете о выполненной лабораторной работе, выполненного как расчётно-пояснительная записка к работе.

# 1. Аналитическая часть

В этом разделе представлено описание алгоритмов.

## 1.1. Описание алгоритмов

**Стандартный алгоритм перемножения матриц**

Стандартный алгоритм умножения матриц предполагает следование определению произведения матриц.

Пусть даны две прямоугольные матрицы A и B размерности m\*n  и n\*q соответственно:



Тогда матрица C размерностью m\*q:

, в которой: ,

называется их *произведением*.

**Алгоритм Винограда**

Алгоритм Винограда умножения матриц основан на снижении доли умножений в алгоритме. Предполагается, что некоторые произведения можно вычислить заранее, а затем переиспользовать при вычислении произведений матриц.

Рассмотрим два вектора и .

Их скалярное произведение равно:

Это равенство можно переписать в виде:

Несмотря на то, что второе выражение требует вычисления большего количества операций, чем первое: 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 позволяет выполнять для каждого элемента лишь первые два умножения и последующие пять сложений, а также дополнительно два сложения.

Алгоритм Винограда состоит из следующих шагов:

1. Вычисление горизонтальных произведений MulH

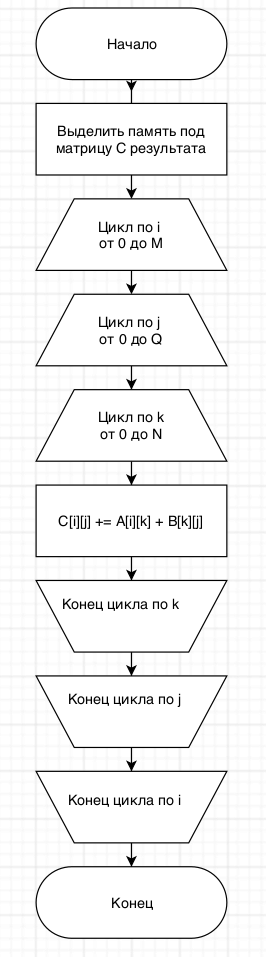
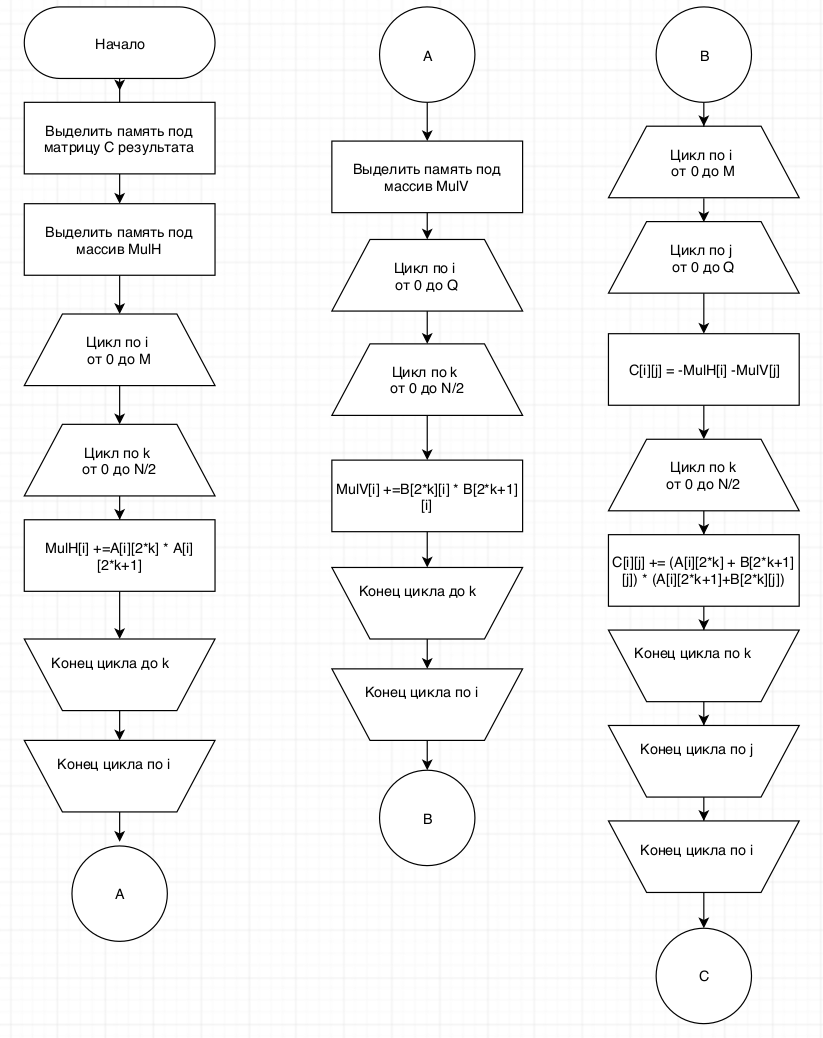
2. Вычисление вертикальных произведений MulV

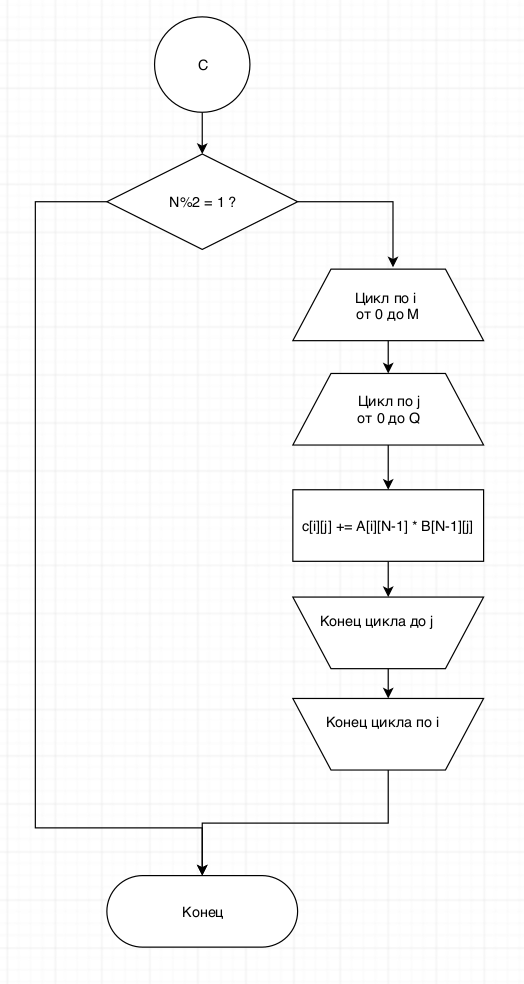
3. Вычисление матрицы результата

4. Корректирование матрицы в случае нечетного N

# 2. Конструкторская часть

## 2.1. Разработка алгоритмов

* 1. В этом разделе представлены блок схемы алгоритмов.
  2. 
  3. Рис 1. Стандартный алгоритм умножения матриц
  4. рис 2. Алгоритм Винограда(часть 1)

рис 3. Алгоритм Винограда(часть 2)

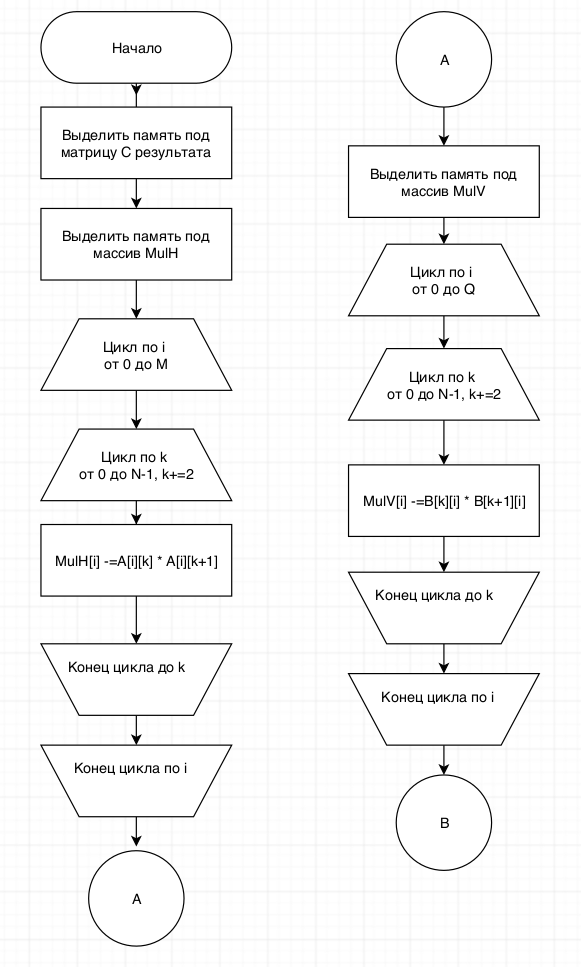
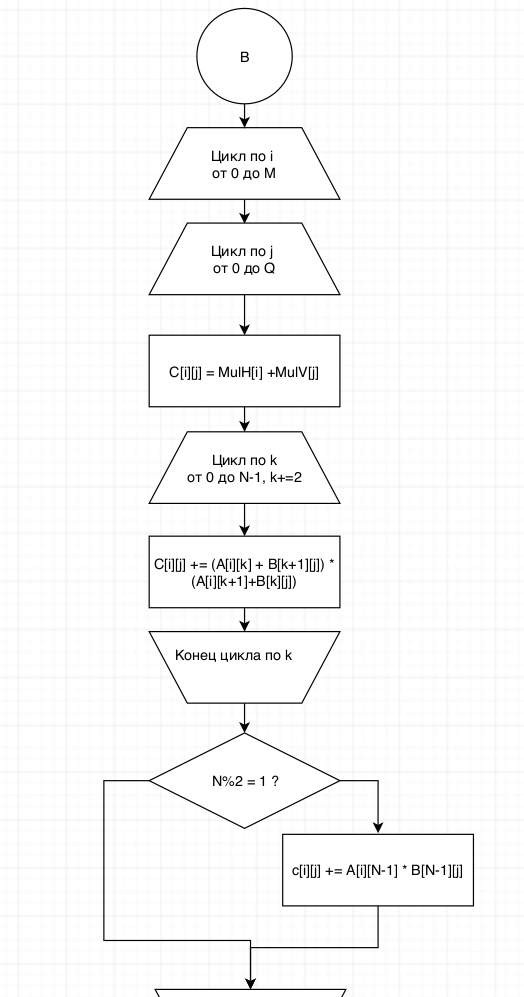


рис 4. Алгоритм Винограда(оптимизированный) часть 1



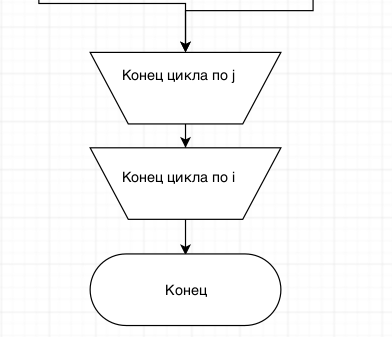


рис 5. Алгоритм Винограда (оптимизированный) часть 2

## 2.2. Подсчет трудоемкости алгоритмов

### Стандартный алгоритм

### Алгоритм Винограда

### Алгоритм Винограда (оптимизированный)

# 3.Технологическая часть

## 3.1.Требования к программному обеспечению

* 1. На вход программа получает две матрицы с размерами MxN и NxQ. На выходе получается матрица размером MxQ.   
     Так же должна быть реализована функция для тестирования и функции замера времени.

## 3.2.Средства реализации

* 1. Для реализации программы был выбран язык С++ в связи с возможностью прибегать к использованию ООП, а так же с моими личным опытом работы с этим ЯП. Среда разработки — Qtcreator. Для работы с матрицами были реализован свой класс Matrix.

## 3.3.Листинг кода

* 1. Далее представлен листинг программы.

Стандартный алгоритм Стандартный алгоритм(оптимизация)

Matrix standart(Matrix m1, Matrix m2) {

int N = m1.cols();

int M = m1.rows();

int Q = m2.cols();

Matrix c(M, Q);

for (int i = 0; i < M; i++) {

for (int j = 0; j < Q; j++) {

for (int k = 0; k < N; k++) {

c[i][j] = c[i][j]+m1[i][k]\*m2[k][j];

}

}

}

return c;

}

Matrix standartO(Matrix m1, Matrix m2) {

int N = m1.cols();

int M = m1.rows();

int Q = m2.cols();

Matrix c(M, Q);

for (int i = 0; i < M; i++) {

for (int j = 0; j < Q; j++) {

for (int k = 0; k < N; k++) {

c[i][j] += m1[i][k]\*m2[k][j];

}

}

}

return c;

}

Алгоритм Винограда

Matrix Vinograd(Matrix A, Matrix B) {

int N = A.cols();

int M = A.rows();

int Q = B.cols();

Matrix c(M, Q);

std::vector<int> MulH(M, 0);

for (int i = 0; i < M; i++)

{

for (int k = 0; k < N/2; k++)

{

MulH[i] = MulH[i] + A[i][2\*k] \* A[i][2\*k+1];

}

}

std::vector<int> MulV(Q, 0);

for (int i = 0; i < Q; i++)

{

for (int k = 0; k < N/2; k++)

{

MulV[i] = MulV[i] + B[2\*k][i]\*B[2\*k+1][i];

}

}

for (int i = 0; i < M; i++) {

for (int j = 0; j < Q; j++) {

c[i][j] = -MulH[i] - MulV[j];

for (int k = 0; k < N/2; k++) {

c[i][j] = c[i][j] + (A[i][2\*k] + B[2\*k+1][j])\*(A[i][2\*k+1] + B[2\*k][j]);

}

}

}

if (N%2 == 1) {

for (int i = 0; i < M; i++) {

for (int j = 0; j < Q; j++) {

c[i][j] = c[i][j] + A[i][N-1]\*B[N-1][j];

}

}

}

return c;

}

Алгоритм Винограда(оптимизированный)

Matrix Vinograd2(Matrix A, Matrix B) {

int N = A.cols();

int M = A.rows();

int Q = B.cols();

Matrix c(M, Q);

std::vector<int> MulH(M,0);

for (int i = 0; i < M; i++)

{

for (int k = 0; k < N-1; k += 2)

{

MulH[i] -= A[i][k] \* A[i][k+1];

}

}

std::vector<int> MulV(Q,0);

for (int i = 0; i < Q; i++)

{

for (int k = 0; k < N-1; k += 2)

{

MulV[i] -= B[k][i]\*B[k+1][i];

}

}

for (int i = 0; i < M; i++) {

for (int j = 0; j < Q; j++) {

c[i][j] = MulH[i] + MulV[j];

for (int k = 0; k < N-1; k += 2) {

c[i][j] += (A[i][k] + B[k+1][j])\*(A[i][k+1] + B[k][j]);

}

if (N%2 == 1) {

c[i][j] += A[i][N-1]\*B[N-1][j];

}

}

}

return c;

}

## 3.4.Описание тестирования

# 4.Экспериментальная часть

## 4.1.Примеры работы

## 4.2.Результаты тестирования

## 4.3.Постановка эксперимента по замеру времени

## Заключение