



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 Якуба Д. В.

Группа ИУ7-53Б

Оценка (баллы) _____

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

Москва — 2020 г.

Оглавление

| | |
|--|-----------|
| Введение | 3 |
| 1 Аналитическая часть | 5 |
| 1.1 Классический алгоритм умножения матриц | 5 |
| 1.2 Алгоритм Копперсмита-Винограда умножения матриц . . . | 6 |
| 2 Конструкторская часть | 7 |
| 2.1 Блок-схема классического алгоритма умножения матриц . | 7 |
| 2.2 Блок-схема алгоритма Копперсмита-Винограда | 7 |
| 2.3 Блок-схема улучшенного алгоритма Копперсмита-Винограда | 7 |
| 2.4 Модель вычислений | 17 |
| 2.5 Трудоемкость алгоритмов | 17 |
| 2.5.1 Классический алгоритм | 17 |
| 2.5.2 Алгоритм Копперсмита-Винограда | 18 |
| 2.5.3 Улучшенный алгоритм Копперсмита-Винограда . . . | 19 |
| 3 Технологическая часть | 20 |
| 3.1 Требования к программному обеспечению | 20 |
| 3.2 Средства реализации программного обеспечения | 20 |
| 3.3 Листинг кода | 21 |
| 3.4 Тестирование программного продукта | 24 |
| 4 Исследовательская часть | 25 |
| 4.1 Пример работы программного обеспечения | 25 |
| 4.2 Технические характеристики | 25 |
| 4.3 Время выполнения алгоритмов | 25 |
| Заключение | 28 |

Введение

Цели лабораторной работы

1. изучение алгоритмов умножения матриц: классического, Копперсмита-Винограда и модифицированного Копперсмита-Винограда;
2. реализация алгоритмов умножения матриц: классического, Копперсмита-Винограда и модифицированного Копперсмита-Винограда;
3. проведение сравнительного анализа трудоёмкости алгоритмов на основе теоретических расчетов и выбранной модели вычислений;
4. сравнительный анализ алгоритмов на основе экспериментальных данных;
5. подготовка отчёта по лабораторной работе.

Определение

Алгоритм Копперсмита-Винограда = это 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 - это размер стороны матрицы. Алгоритм Копперсмита-Винограда с учётом 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 [2]. Поэтому пользуются алгоритмом Штрассена по причинам простоты реализации и меньшей константе в оценке трудоемкости.

1 | Аналитическая часть

1.1 Классический алгоритм умножения матриц

Пусть даны две прямоугольные матрицы

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

тогда матрица C

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

где

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

будет называться произведением матриц A и B .

Реализация классического алгоритма умножения двух матриц заключается в реализации вычисления элементов итоговой матрицы по формуле 1.3

1.2 Алгоритм Копперсмита-Винограда умножения матриц

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 классический алгоритм умножения матриц и алгоритм Копперсмита-Винограда умножения матриц. Основное отличие данных алгоритмов заключается в наличии предварительной обработки данных и количестве проводящихся операций умножения.

2 | Конструкторская часть

2.1 Блок-схема классического алгоритма умножения матриц

Блок-схема классического алгоритма умножения матриц предоставлена на рисунке 2.1

2.2 Блок-схема алгоритма Копперсмита-Винограда

Блок-схема алгоритма Копперсмита-Винограда и алгоритмов предвычисления предоставлены на рисунках 2.2 - 2.5.

2.3 Блок-схема улучшенного алгоритма Копперсмита-Винограда

Блок-схема улучшенного алгоритма Копперсмита-Винограда и алгоритмов предвычисления предоставлены на рисунках 2.6 - 2.9.

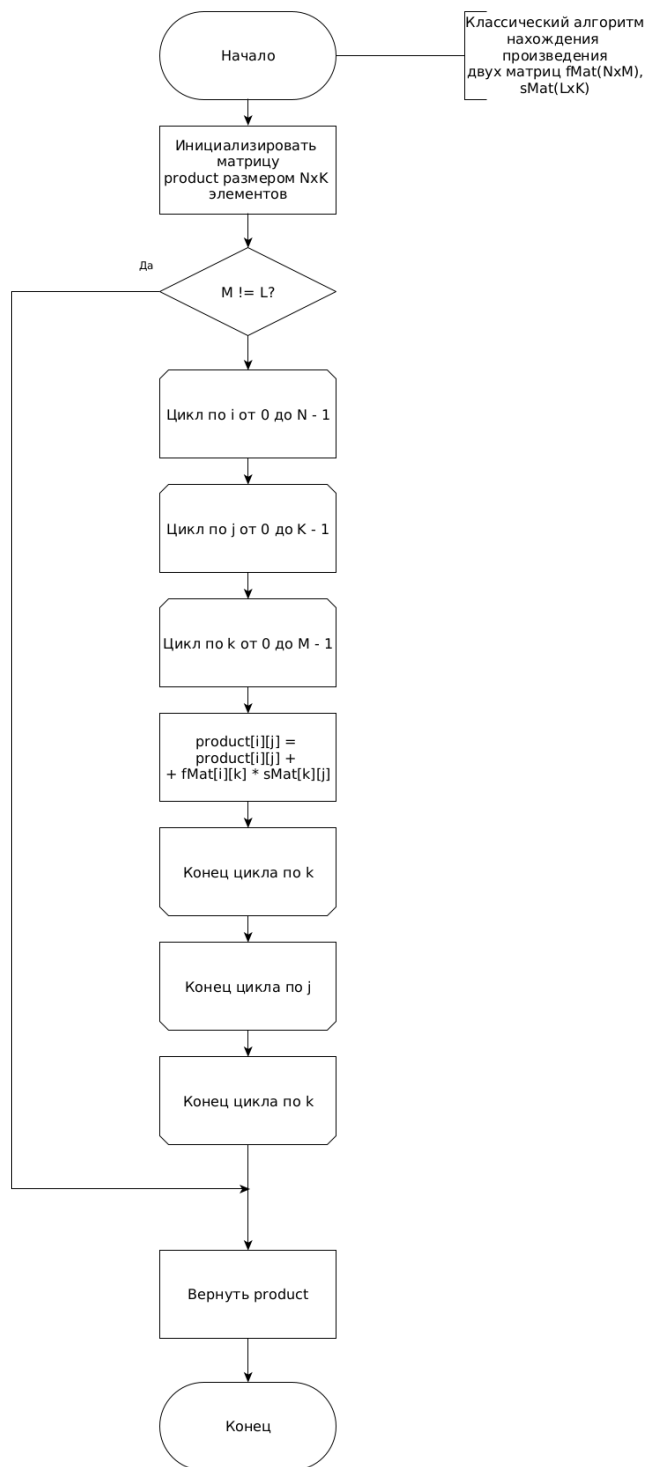


Рис. 2.1: Блок-схема классического алгоритма умножения матрицы.

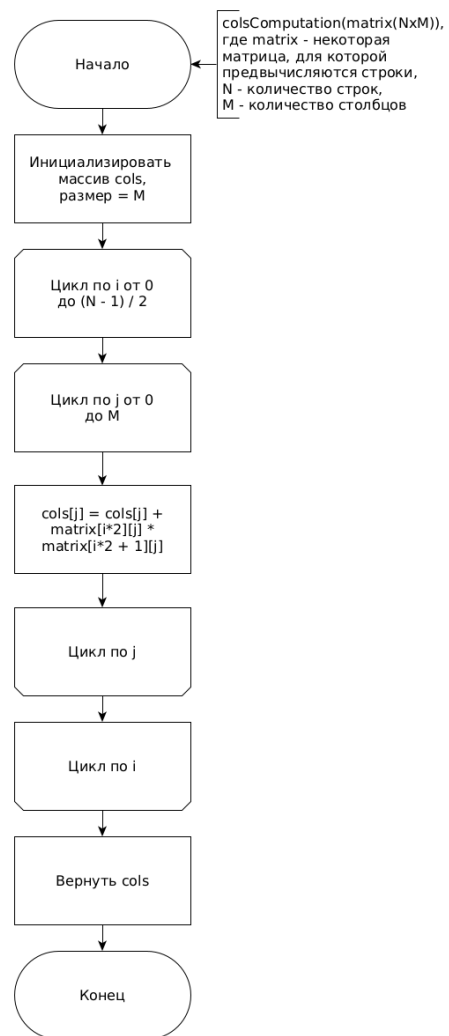


Рис. 2.2: Блок-схема алгоритма предвычисления столбцов.

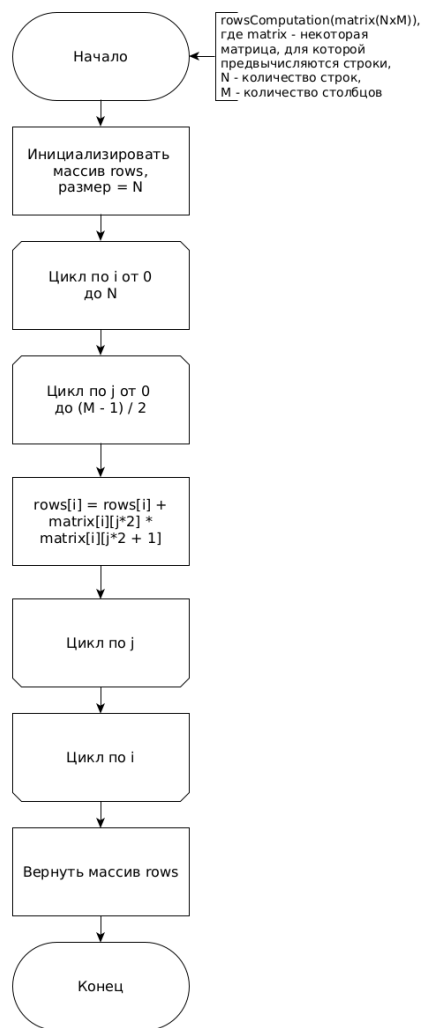


Рис. 2.3: Блок-схема алгоритма предвычисления строк.

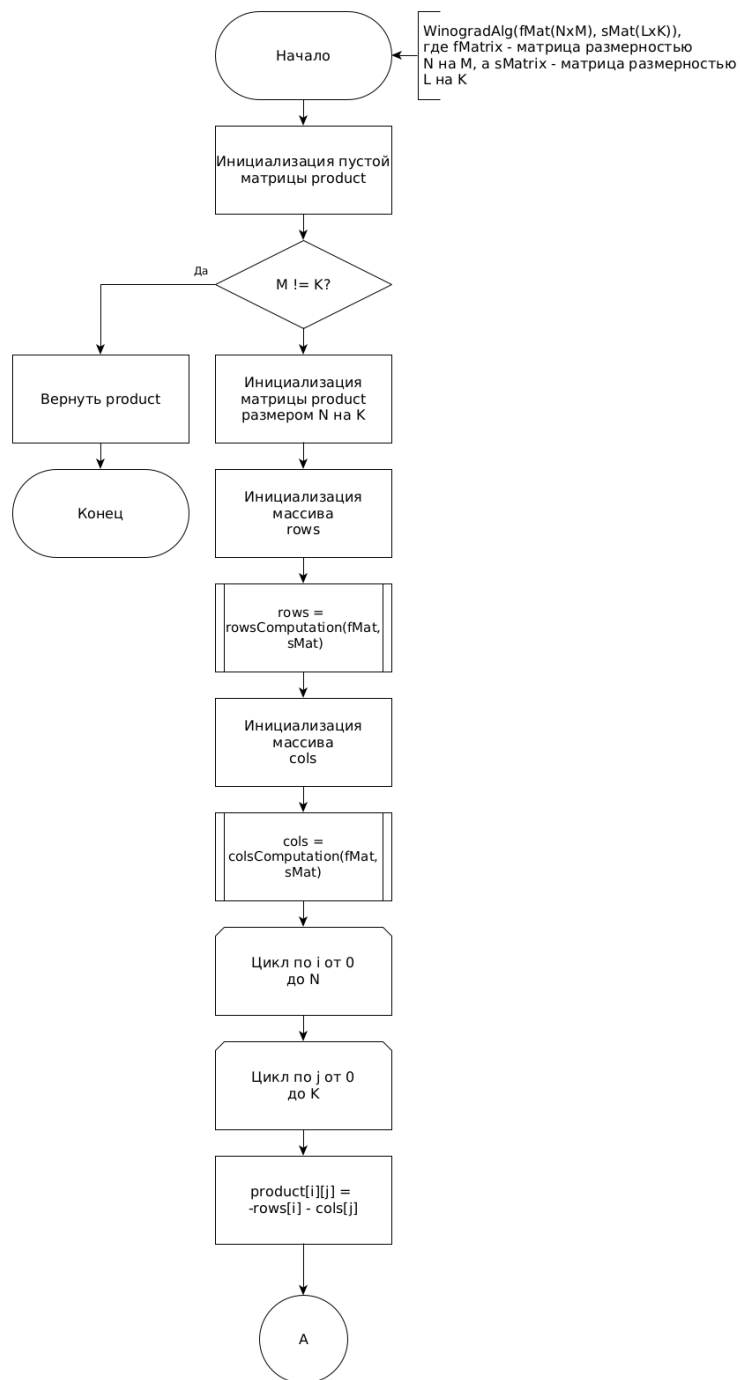


Рис. 2.4: Блок-схема алгоритма Копперсмита-Винограда.

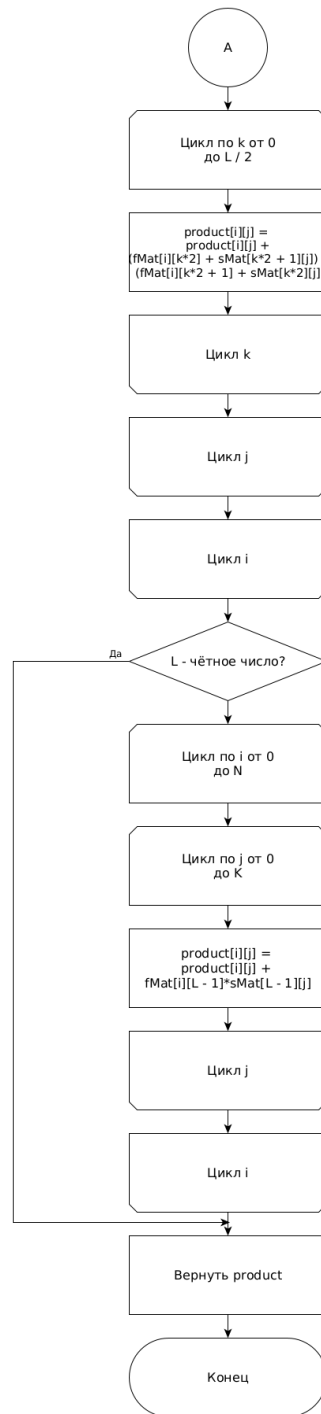


Рис. 2.5: Продолжение блок-схемы алгоритма Копперсмита-Винограда.

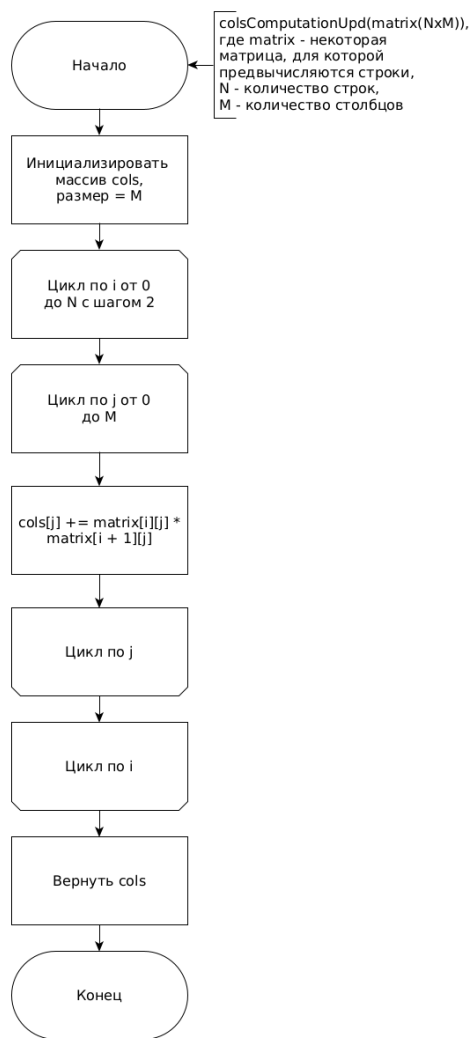


Рис. 2.6: Блок-схема алгоритма предвычисления столбцов.

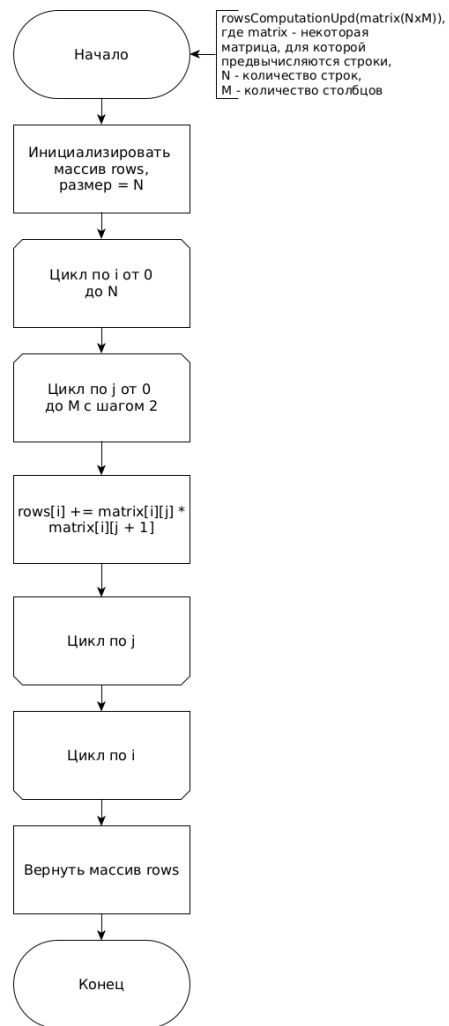


Рис. 2.7: Блок-схема алгоритма предвычисления строк.

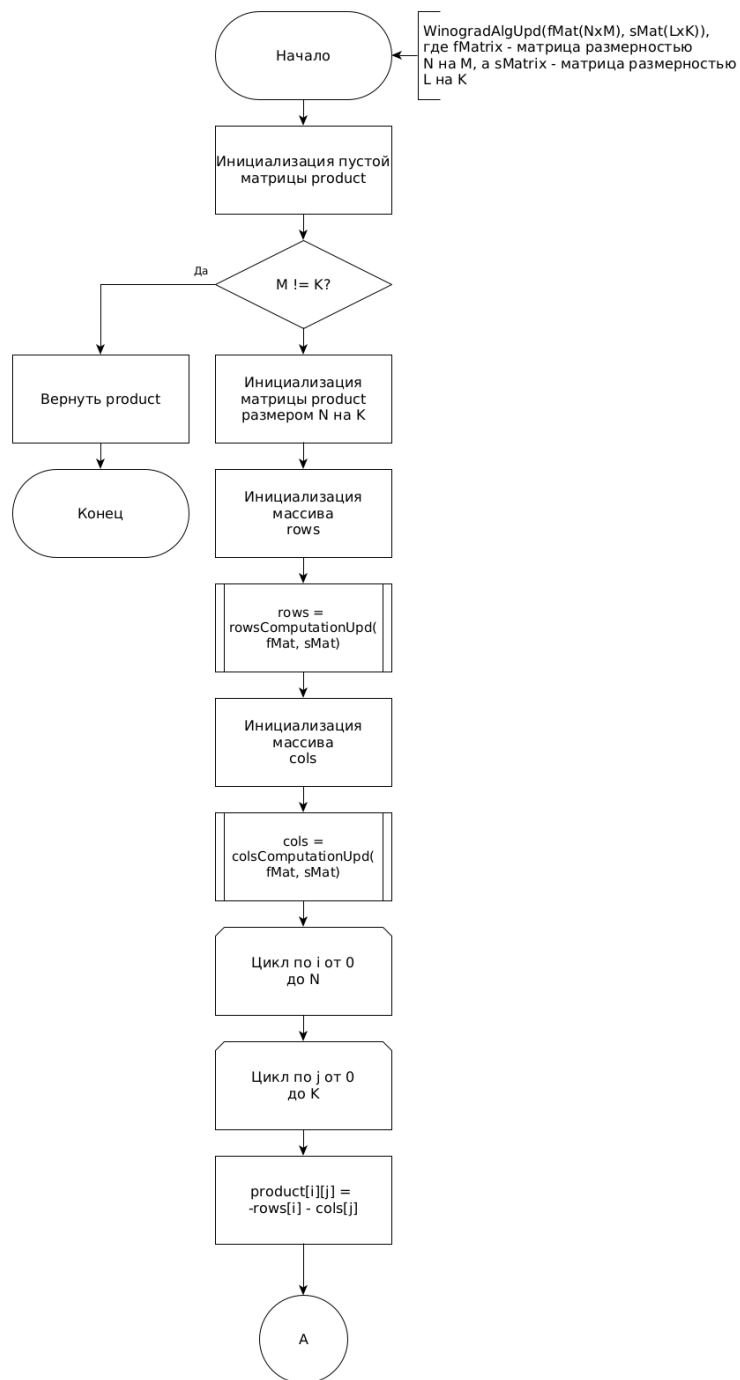


Рис. 2.8: Блок-схема улучшенного алгоритма Копперсмита-Винограда.

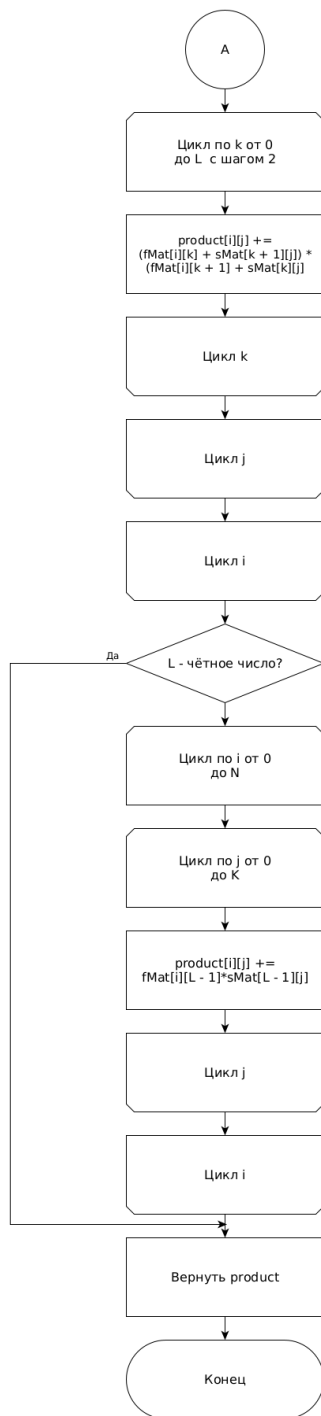


Рис. 2.9: Продолжение блок-схемы улучшенного алгоритма Копперсмита-Винограда.

2.4 Модель вычислений

Для последующего вычисления трудоемкости необходимо ввести модель вычислений:

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

$$+, -, /, \%, ==, !=, <, >, <=, >=, [], ++, --, + =, - = \quad (2.1)$$

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

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

3. трудоемкость цикла рассчитывается, как (2.3);

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

4. трудоемкость вызова функции равна 0.

2.5 Трудоемкость алгоритмов

2.5.1 Классический алгоритм

Трудоемкость стандартного алгоритма умножения двух матриц будет включать в себя:

- Цикл по $i \in [1..N]$, где N - количество строк первой переданной матрицы;
- Цикл по $j \in [1..K]$, где K - количество столбцов второй переданной матрицы;
- Цикл по $k \in [1..M]$, где M - количество столбцов первой переданной матрицы.

При этом для цикла по $i \in [1..N]$ будем иметь трудоёмкость f_i такую, что: $f_i = 2 + N \cdot (2 + f_{body})$, где f_{body} - трудоёмкость тела цикла.

Для цикла по $j \in [1..K]$ трудоёмкость $f_j = 2 + K \cdot (2 + f_{body})$.

Для цикла по $k \in [1..M]$ трудоёмкость $f_k = 2 + 8 \cdot M$.

Как итог, общая трудоёмкость алгоритма может быть представлена выражением 2.4:

$$f = 2 + N \cdot (2 + (2 + K \cdot (2 + 2 + 8 \cdot M))) \quad (2.4)$$

При этом выражение 2.4 преобразуется в 2.5:

$$f = 2 + 4 \cdot N + 4 \cdot K \cdot N + 8 \cdot N \cdot K \cdot M \quad (2.5)$$

2.5.2 Алгоритм Копперсмита-Винограда

Трудоёмкость алгоритма Копперсмита-Винограда умножения двух матриц будет включать в себя:

- Создание и инициализация массивов *cols* и *rows*;
- Заполнение массива *cols*;
- Заполнение массива *rows*;
- Цикл заполнения итоговой матрицы для чётных размеров матриц;
- Цикл для дополнительного заполнения при нечётных размерах матриц;

Трудоёмкость создания и инициализации массивов *cols* и *rows*: $f_{rc} = N + M$, где M - размер массива *cols*, количество столбцов второй переданной матрицы, а N - размер массива *rows*, количество строк первой переданной матрицы.

Трудоёмкость заполнения массива *cols*: $f_{cols} = 5 + \frac{K}{2}(2 + 11 \cdot K)$, где K - количество строк второй переданной матрицы, L - количество столбцов второй переданной матрицы.

Трудоёмкость заполнения массива *rows*: $f_{rows} = 2 + N \cdot (5 + 11 \cdot \frac{K}{2})$, где M - количество строк первой переданной матрицы, K - количество столбцов первой переданной матрицы.

Трудоёмкость цикла заполнения итоговой матрицы для чётных размеров матриц: $f_{even cyc} = 2 + N \cdot (2 + L \cdot (5 + 2 + 22 \cdot \frac{K}{2}))$, где N - количество

строк первой матрицы, L - количество столбцов второй матрицы, K - количество строк второй матрицы.

$$f_{evencyc} = 2 + 2 \cdot N + 7 \cdot N \cdot L + 11 \cdot N \cdot L \cdot K.$$

Трудоёмкость цикла для дополнительного заполнения при нечётных размерах матриц: $f_{end} = 2 + M \cdot (2 + \frac{K}{2} \cdot 12)$. При этом рассматривается два случая, представленных в выражении 2.6:

$$f_{end} = \begin{cases} 2, & \text{чётная,} \\ 4 + 22 \cdot M + 6 \cdot K \cdot M, & \text{иначе.} \end{cases} \quad (2.6)$$

Общую формулу для вычисления трудоёмкости можно представить как 2.7:

$$f = f_{rc} + f_{cols} + f_{rows} + f_{evencyc} + f_{end} \quad (2.7)$$

Для худшего случая, когда размеры матриц нечётные, из 2.7:

$$f = 6 + 2 \cdot N + 5 \cdot K + 22 \cdot M + 7 \cdot N \cdot L + 17 \cdot M \cdot K + 11 \cdot N \cdot L \cdot K \quad (2.8)$$

Для лучшего случая, когда размеры матриц чётные, из 2.7:

$$f = 4 + 2 \cdot N + 11 \cdot M + 7 \cdot N \cdot L + 11 \cdot N \cdot L \cdot K \quad (2.9)$$

2.5.3 Улучшенный алгоритм Копперсмита-Винограда

3 | Технологическая часть

3.1 Требования к программному обеспечению

- Входные данные - две матрицы размерностью $M \times N$ и $K \times L$;
- Выходные данные - результат умножения двух переданных матриц.

3.2 Средства реализации программного обеспечения

При написании программного продукта был использован язык программирования Kotlin [4].

Данный выбор обусловлен следующими факторами:

- Высокая вычислительная производительность;
- Большое количество справочной литературы.

Для тестирования производительности реализаций алгоритмов использовалась утилита `measureNanoTime`.

При написании программного продукта использовалась среда разработки IntelliJ IDEA.

Данный выбор обусловлен тем, что язык программирования Kotlin - это разработка компании JetBrains, поставляющей данную среду разработки.

3.3 Листинг кода

В листингах ?? - ?? предоставлены реализации рассматриваемых алгоритмов.

Листинг 3.1: Функция реализации алгоритма классического умножения матриц

```
1 fun matricesMult(fMatrix: Array<IntArray>, sMatrix: Array<
  IntArray>) : Array<IntArray>
2 {
3     if (fMatrix[0].size != sMatrix.size)
4         return emptyArray()
5
6     val product = Array(fMatrix.size) { IntArray(sMatrix[0].size) }
7     for (i in fMatrix.indices)
8         for (j in sMatrix[0].indices)
9             for (k in fMatrix[0].indices)
10                product[i][j] += fMatrix[i][k] * sMatrix[k][j]
11     return product
12 }
```

Листинг 3.2: Функция реализации алгоритма Кошперсмита-Винограда

```
1 fun rowsComputation(matrix: Array<IntArray>) : IntArray
2 {
3     val computedRows = IntArray(matrix.size)
4
5     for (i in matrix.indices)
6         for (j in 0 until (matrix[0].size - 1) / 2)
7             computedRows[i] = computedRows[i] + matrix[i][j * 2]
8                 * matrix[i][j * 2 + 1]
9
10    return computedRows
11 }
12 fun colsComputation(matrix: Array<IntArray>) : IntArray
13 {
14     val computedCols = IntArray(matrix[0].size)
15
16     for (i in 0 until (matrix.size - 1) / 2)
17         for (j in matrix[0].indices)
18             computedCols[j] = computedCols[j] + matrix[i * 2][j]
19                 * matrix[i * 2 + 1][j]
```

```

19
20     return computedCols
21 }
22
23 fun WinogradMultiplication(fMatrix: Array<IntArray>, sMatrix:
    Array<IntArray>) : Array<IntArray>
24 {
25     if (fMatrix[0].size != sMatrix.size)
26         return emptyArray()
27
28     val computedRows = rowsComputation(fMatrix)
29     val computedCols = colsComputation(sMatrix)
30
31     val product = Array(fMatrix.size) { IntArray(sMatrix[0].size)
32     }
33     for (i in product.indices)
34         for (j in product[0].indices)
35         {
36             product[i][j] = -computedRows[i] - computedCols[j]
37
38             for (k in 0 until (sMatrix.size / 2))
39                 product[i][j] = product[i][j] + (fMatrix[i][k *
40                 2] + sMatrix[k * 2 + 1][j]) * (fMatrix[i][k *
41                 2 + 1] + sMatrix[k * 2][j])
42         }
43
44     if (sMatrix.size % 2 != 0)
45     {
46         for (i in product.indices)
47             for (j in product[0].indices)
48                 product[i][j] = product[i][j] + fMatrix[i][
49                 sMatrix.size - 1] * sMatrix[sMatrix.size -
50                 1][j]
51     }
52
53     return product
54 }

```

Листинг 3.3: Функция реализации улучшенного алгоритма
Копперсмита-Винограда

```

1 fun rowsComputationModified(matrix: Array<IntArray>) : IntArray
2 {
3     val computedRows = IntArray(matrix.size)
4     for (i in matrix.indices)

```

```

5         for (j in matrix[0].indices step 2)
6             computedRows[i] += matrix[i][j] * matrix[i][j + 1]
7
8     return computedRows
9 }
10
11 fun colsComputationModified(matrix: Array<IntArray>) : IntArray
12 {
13     val computedCols = IntArray(matrix[0].size)
14
15     for (i in matrix.indices step 2)
16         for (j in matrix[0].indices)
17             computedCols[j] += matrix[i][j] * matrix[i + 1][j]
18
19     return computedCols
20 }
21
22 fun WinogradMultiplicationModified(fMatrix: Array<IntArray>,
23 sMatrix: Array<IntArray>) : Array<IntArray>
24 {
25     if (fMatrix[0].size != sMatrix.size)
26         return emptyArray()
27
28     val computedRows = rowsComputationModified(fMatrix)
29     val computedCols = colsComputationModified(sMatrix)
30
31     val product = Array(fMatrix.size) { IntArray(sMatrix[0].size) }
32     for (i in product.indices)
33         for (j in product[0].indices)
34         {
35             product[i][j] = -computedRows[i] - computedCols[j]
36
37             for (k in 0 until (sMatrix.size - 1) step 2)
38                 product[i][j] += (fMatrix[i][k] + sMatrix[k + 1][j]) * (fMatrix[i][k + 1] + sMatrix[k][j])
39         }
40
41     if (sMatrix.size % 2 != 0)
42     {
43         for (i in product.indices)
44             for (j in product[0].indices)
45                 product[i][j] += fMatrix[i][sMatrix.size - 1] * sMatrix[sMatrix.size - 1][j]
46     }
47 }

```



```

46 |
47 |     return product
48 | }

```

3.4 Тестирование программного продукта

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

| Матрица 1 | Матрица 2 | Ожидаемый результат |
|---|---|---|
| $\begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix}$ | $\begin{pmatrix} 1 & 3 & 3 \\ 1 & 3 & 3 \\ 1 & 2 & 2 \end{pmatrix}$ | $\begin{pmatrix} 6 & 15 & 15 \\ 6 & 15 & 15 \\ 3 & 8 & 8 \end{pmatrix}$ |
| $\begin{pmatrix} 1 & 2 & 4 \\ 1 & 2 & 4 \end{pmatrix}$ | $\begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix}$ | $\begin{pmatrix} 32 \\ 32 \end{pmatrix}$ |
| (5) | (666) | (3330) |
| $\begin{pmatrix} -1 & -2 & 3 \\ 1 & 2 & 3 \\ -1 & -2 & 3 \end{pmatrix}$ | $\begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{pmatrix}$ | $\begin{pmatrix} 4 & 4 & 4 \\ 14 & 14 & 14 \\ 4 & 4 & 4 \end{pmatrix}$ |
| (666 666) | (777 777) | Ошибка |

Таблица 3.1: Тестирование функций

Вывод

Спроектированные алгоритмы вычисления произведения двух матриц были реализованы и протестированы.

4 | Исследовательская часть

4.1 Пример работы программного обеспечения

4.2 Технические характеристики

Технические характеристики ЭВМ, на котором выполнялись исследования:

- ОС: Manjaro Linux 20.1.1 Mikah
- Оперативная память: 16 Гб
- Процессор: Intel Core i7-10510U

При проведении замеров времени ноутбук был подключен к сети электропитания.

4.3 Время выполнения алгоритмов

Алгоритмы тестировались на данных, сгенерированных случайным образом один раз.

Результаты замеров времени приведены в таблице 4.1. На рисунках 4.1 и 4.2 приведены графики зависимостей времени работы алгоритмов от количества строк и столбцов матриц (в чётном и нечётном вариантах). В таблице КА - Классический Алгоритм, КВ - Алгоритм Копперсмита-Винограда, УКВ - Улучшенный Алгоритм Копперсмита-Винограда.

Таблица 4.1: Замеры времени для квадратных матриц различных размеров

| Размер матрицы | КА | КВ | УКВ |
|----------------|-----------|-----------|-----------|
| 100 | 2148121 | 2302331 | 1921005 |
| 101 | 2312114 | 2623891 | 2032155 |
| 200 | 17350292 | 22592034 | 1425033 |
| 201 | 20247410 | 21694235 | 17554153 |
| 300 | 68920554 | 73453362 | 57000923 |
| 301 | 75166547 | 77955778 | 64421195 |
| 400 | 211301483 | 205981760 | 172968826 |
| 401 | 227614782 | 218087162 | 171881527 |
| 500 | 367822853 | 351730341 | 340284336 |
| 501 | 364368768 | 362588416 | 358108198 |
| 600 | 678478122 | 658012453 | 625149992 |
| 601 | 672846913 | 671159157 | 647843183 |

Вывод

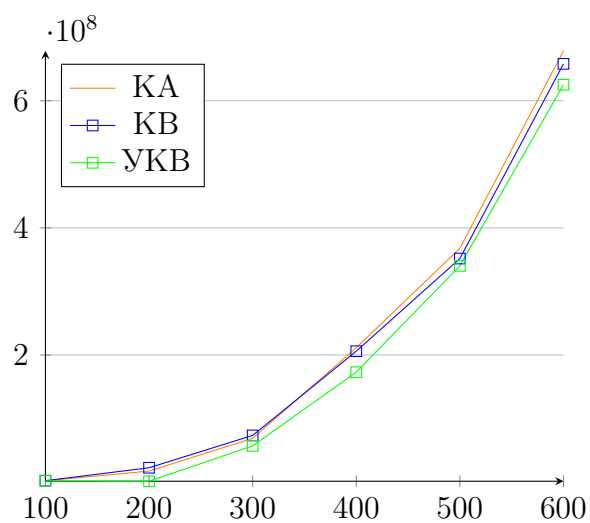


Рис. 4.1: Зависимость времени работы от размера матриц (чётные значения размерностей)

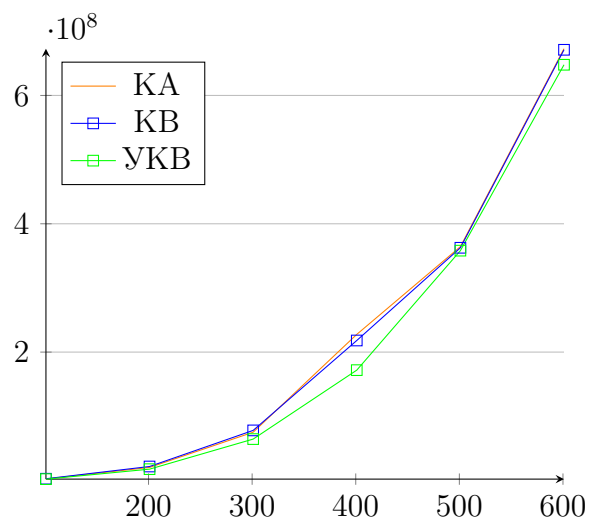


Рис. 4.2: Зависимость времени работы от размера матриц (нечётные значения размерностей)

Заключение

В ходе выполнения лабораторной работы:

-
-
-
-
-
- Были получены практические навыки реализации алгоритмов на ЯП Kotlin.

Литература

- [1] Coppersmith D., Winograd S. Matrix multiplication via arithmetic progressions // Journal of Symbolic Computation. 1990. no. 9. P. 251–280.
- [2] Robinson S. Toward an Optimal Algorithm for Matrix Multiplication // SIAM News. 2005. November. Vol. 38, no. 9.
- [3] Погорелов Дмитрий Александрович Таразанов Артемий Михайлович Волкова Лилия Леонидовна. Оптимизация классического алгоритма Винограда для перемножения матриц // Журнал №1. 2019. Т. 49.
- [4] Kotlin language specification [Электронный ресурс]. Режим доступа: <https://kotlinlang.org/spec/introduction.html> (дата обращения 09.10.2020).