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»
ДИСЦИПЛИНА «Анализ алгоритмов»

Лабораторная работа № 2

Дисциплина Анализ алгоритмов

Тема Перемножение матриц

Студент Боренко Анастасия

Группа ИУ7-52Б

Оценка (баллы) __________

Преподаватель Волкова Л.Л.

Москва — 2021г.

Содержание

Введение

Матрицы применяются в повседневной жизни и используются во всех отраслях деятельности при решении различных практических задач в математике, биологии, физике, технике, химии, экономике, маркетинге, психологии и других областях науки. В математике с их помощью решаются системы линейных уравнений.

Поэтому алгоритмы операций над матрицами сегодня очень актуальны. Умножение матриц - одна из таких операций. Она используется в компьютерной графике, значительно облегчая обработку изображений.

Целью данной лабораторной являются изучение метода динамического программирования на материале алгоритмов умножения матриц. Задачами данной лабораторной являются:

1. изучение стандартного алгоритма умножения матриц и алгоритма Винограда;
2. оптимизация алгоритма Винограда;
3. определение трудоемкостей исследуемых алгоритмов;
4. применение метода динамического программирования для матричных алгоритмов;
5. реализация указанных алгоритмов;
6. сравнительный анализ трех алгоритмов: стандартного алгоритма умножения матриц, алгоритма Винограда и оптимизированного алгоритма Винограда, - по затрачиваемым ресурсам(времени и памяти);
7. экспериментальное подтверждение различий в трудоемкости алгоритмов с указанием лучшего и худшего случаев;

8. описание и обоснование полученных результатов в отчете о выполненной лабораторной работе, выполненного как расчетно-пояснительная записка к работе.

1. Аналитическая часть

1.1 Описание задачи

Матрица – математический объект, записываемый в виде прямоугольной таблицы элементов кольца или поля (например, целых, действительных или комплексных чисел), который представляет собой совокупность строк и столбцов, на пересечении которых находятся его элементы. Количество строк и столбцов задает размер матрицы. Хотя исторически рассматривались, например, треугольные матрицы, в настоящее время говорят исключительно о матрицах прямоугольной формы, так как они являются наиболее удобными и общими.

Умножение матриц – одна из основных операций над матрицами. Матрица, получаемая в результате операции умножения, называется произведением матриц. Пусть даны 2 матрицы размерами $l * m$ и $m * n$ (формула 1.1):

$$A = \begin{bmatrix} a_{11}, a_{12}, ..., a_{1m} \\ a_{21}, a_{22}, ..., a_{2m} \\ \dots\dots\dots\dots\dots\dots \\ a_{n1}, a_{n2}, ..., a_{nm} \end{bmatrix} B = \begin{bmatrix} b_{11}, b_{12}, ..., b_{1l} \\ b_{21}, b_{22}, ..., b_{2l} \\ \dots\dots\dots\dots\dots\dots \\ b_{m1}, b_{m2}, ..., b_{ml} \end{bmatrix} \quad (1.1)$$

Произведением матриц называется матрица $C = A * B$, которая равна 1.1.

$$C = \begin{bmatrix} c_{11}, c_{12}, ..., c_{1l} \\ c_{21}, c_{22}, ..., c_{2l} \\ \dots\dots\dots\dots\dots\dots \\ c_{n1}, c_{n2}, ..., c_{nl} \end{bmatrix} \quad (1.2)$$

Каждый элемент вычисляется по формуле 1.1.

$$c_{ij} = \sum_{k=1}^{m} a_{ik} \cdot b_{kj} (i = 1, ..., n; j = 1, ..., l) \tag{1.3}$$

1.2 Стандартный алгоритм умножения матриц

На рисунке 1.1 показан способ вычисления значения ячеек матрицы C. Стандартный алгоритм умножения матриц заключается в последовательном вычислении значений ячеек матриц.

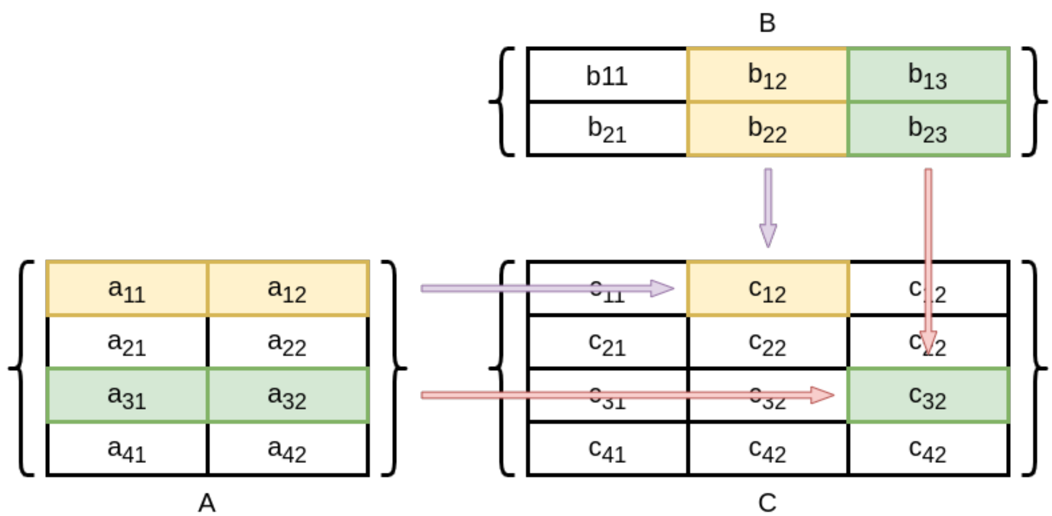


Рис. 1.1: Вычисление значения ячеек матрицы C

1.3 Алгоритм Винограда

Алгоритм Винограда разработан на идее уменьшения количества операций умножений в цикле стандартного умножения матриц с помощью предварительной обработки. Каждый элемент произведения матриц представляет собой скалярное произведение соответ ствующих строки и столбца исходных матриц.

Рассмотрим скалярное произведение векторов

$V = (v_1, v_2, v_3, v_4)$ и $W = (w_1, w_2, w_3, w_4)$

Вот оно:

$$C = VW = v1w1 + v2w2 + v3w3 + v4w4 = \tag{1.4}$$
$$(v1 + w2)(v2 + w1) + (v3 + w4)(v4 + w3) \underbrace{-v1v2 - v3v4 - w1w2 - w3w4}_{\text{Можно вычислить заранее}} \tag{1.5}$$

Несмотря на то, что конечный вариант формулы (1.3) выглядит громоздко, и в нем целых 6 операций умножения вместо 4. Однако последние 4 операции можно вычислить заранее. Тогда над обработанными данными надо будет выполнять только 2 умножения и 5 сложений (и еще 2 сложения, чтобы добавить вычисленные заранее данные).

1.4 Вычисление трудоемкости

В данной работе необходимо расчитать трудоемкость исследуемых алгоритмов. Будет использоваться следующая модель вычислений:

1. Операции, трудоемкость которых равна 1: =, +, -, ++, −, +=, -=, <=, >=, ==, !=, [ ], «, »
2. Операции, трудоемкость которых равна 2: *, /, *=. /=, %, %=
3. Трудоемкость условного перехода равна 1.
4. Трудоемкость цикла равна

   $f_{for} = f_{init} + f_{comp} + N \cdot (f_{inc} + f_{comp} + f_{body})$

1.5 Вывод аналитической части

В данной работе стоит задача реализации следующих алгоритмов: стандартного алгоритма умножения матриц, алгоритма Винограда и задача оптимизации алгоритма Винограда. Необходимо сравнить алгоритмы умножения матриц по эффективности по времени. Подтвердить эффективность оптимизированного алгоритма Винограда.

2. Конструкторская часть

2.1 Схемы алгоритмов

2.1.1 Схема стандартного алгоритма умножения матриц

На рисунке 2.1 показана схема алгоритма расчета расстояния Левенштейна матричным способом.

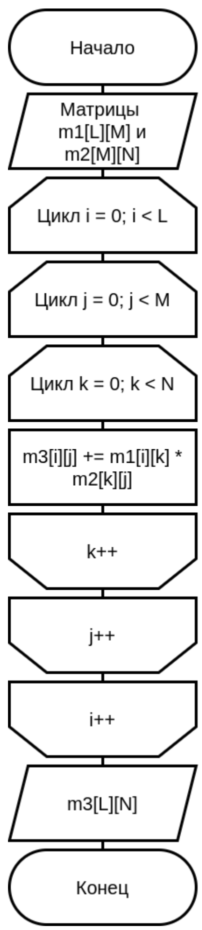


Рис. 2.1: Схема стандартного алгоритма умножения матриц

2.1.2 Схема алгоритма Винограда

На рисунке 2.2 показана схема алгоритма расчета расстояния Левенштейна с помощью рекурсии.

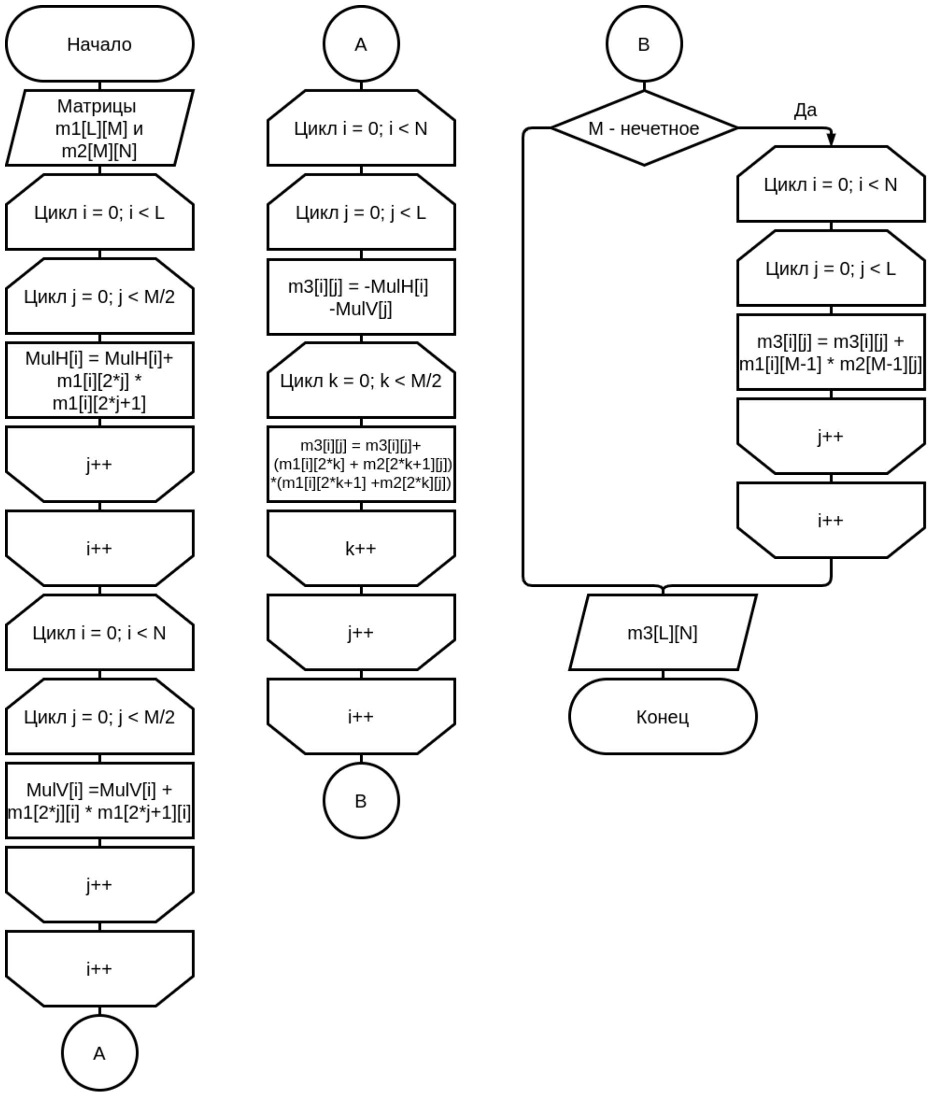


Рис. 2.2: Схема алгоритма Винограда

2.1.3 Схема оптимизированного алгоритма Винограда

На рисунке 2.3 показана схема алгоритма расчета расстояния Левенштейна с помощью рекурсии.

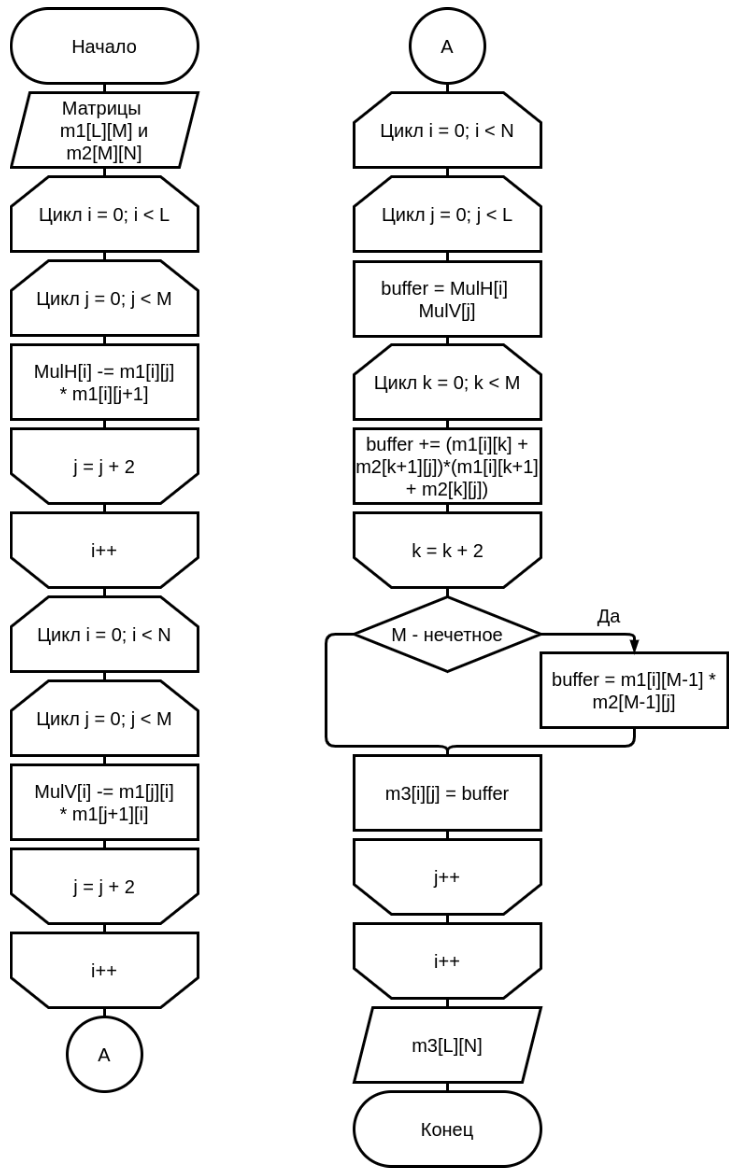


Рис. 2.3: Схема оптимизированного алгоритма Винограда

2.2 Трудоемкость алгоритмов

2.2.1 Трудоемкость стандартного алгоритма умножения матриц

$f = 2 + L \cdot (2 + 2 + M \cdot (2 + 2 + N \cdot (2 + 8 + 1 + 1 + 2)))$

2.2.2 Трудоемкость алгоритма Винограда

Для наглядного подсчета можно заполнить таблицу 2.1, а после посчитать общую трудоемкоть.

Таблица 2.1 – Трудоемкость алгоритма Винограда

Часть алгоритма	Трудоемкость
Инициализация MulH и MulV	2
Заполнение MulH	$2 + L \cdot (2 + 2 + M/2 \cdot (3 + 6 + 6))$
Заполнение MulV	$2 + N \cdot (2 + 2 + M/2 \cdot (3 + 6 + 6))$
Подсчет результата	$2 + L \cdot (2 + 2 + N \cdot (2 + 7 + 2 + M/2 \cdot (3 + 23)))$
Усл. оператор (на нечет. m)	2
Тело усл. оператора (нечет. m)	$2 + N \cdot (2 + 2 + L \cdot (9 + 2))$

Общая трудоемкость:

$$f = 2+2+L\cdot(2+2+M/2\cdot(3+6+6))+2+N\cdot(2+2+M/2\cdot(3+6+6))+2+L\cdot(2+2+N\cdot(2+7+2+M/2\cdot(3+23)))+2+\left\{\begin{array}{c} 2 + N \cdot (2 + 2 + L \cdot (13 + 2)), (m - \text{нечет.}) \\ 0, (m - \text{чет.}) \end{array}\right\}$$

$$f = 13 \cdot M \cdot N \cdot L + 11 \cdot L \cdot N + 7,5 \cdot M \cdot N + 7,5 \cdot M \cdot L + 8 \cdot L + 4 \cdot N + 10 + \left\{\begin{array}{c} 15 \cdot N \cdot L + 4 \cdot N + 2, (m - \text{нечет.}) \\ 0, (m - \text{чет.}) \end{array}\right\}$$

2.2.3 Трудоемкость оптимизированного алгоритма Винограда

Таблица трудоемкостей частей алгоритма: 2.2, а после посчитать общую трудоемкоть.

Общая трудоемкость:

$$f = 2 + 2 + L \cdot (2 + 2 + M/2 \cdot (2 + 8)) + 2 + N \cdot (2 + 2 + M/2 \cdot (2 + 8)) + 2 + L \cdot (2 + 2 + N \cdot (2 + 10 + M/2 \cdot (2 + 14))) + 2 \cdot N \cdot L + \left\{\begin{array}{c} 9 \cdot N \cdot L, (m - \text{нечет.}) \\ 0, (m - \text{чет.}) \end{array}\right\}$$

$$f = 8\cdot M\cdot N\cdot L+14\cdot L\cdot N+5\cdot M\cdot N+5\cdot M\cdot L+8\cdot L+4\cdot N+8+\left\{\begin{array}{c} 9 \cdot N \cdot L, (m - \text{нечет.}) \\ 0, (m - \text{чет.}) \end{array}\right\}$$

Таблица 2.2 – Трудоемкость оптимизированного алгоритма Винограда

Часть алгоритма	Трудоемкость
Инициализация MulH и MulV	2
Заполнение MulH	$2 + L \cdot (2 + 2 + M/2 \cdot (2 + 8))$
Заполнение MulV	$2 + N \cdot (2 + 2 + M/2 \cdot (2 + 8))$
Подсчет результата	$2 + L \cdot (2 + 2 + N \cdot (2 + 10 + M/2 \cdot (2 + 14)))$
Усл. оператор (на нечет. m)	$2 \cdot N \cdot L$
Тело усл. оператора (нечет. m)	$9 \cdot N \cdot L$

2.3 Сравнительный анализ трудоемкостей алгоритмов

Сравнив трудоемкости, можно сделать вывод, что самым быстрым алгоритмом из рассматриваемых является оптимизированный алгоритм Винограда. По трудоемкости он быстрее обычного алгоритма Винограда, значит он был оптимизирован верно (проделанные над ним изменения можно называть оптимизацией). Самым неэффективным по времени алгоритмом является стандартный алгоритм умножения матриц. Проверим предположения, проанализировав полученные экспериментально данные.

2.4 Структуры данных

При реализации приведенных алгоритмов потребуются следующие типы данных: матрица, массив.

2.5 Тестирование

2.5.1 Классы эквивалентности

Для алгоритма умножения матриц можно выделить следующие классы эквивалентности (даны матрицы N*M и K*S):

1. $M \neq K$

2. $M = K$

2.5.2 Способы тестирования

При разработке программы удобно использовать следующие методы тестирования:

1. Модульные тесты

2. Функциональные тесты

2.6 Вывод конструкторской части

На основе данных, полученных в аналитическом разделе, были построены схемы используемых алгоритмов, выделены необходимые для реализации структуры данных и методы тестирования.

3. Технологическая часть

3.1 Требования к ПО

Требования к программному обеспечению:

1. Наличие меню для реализации выбора пользователя.
2. На вход подаются 2 матрицы
3. Результат в зависимости от выбора пользователя:
   (a) произведение введеных матриц
   (b) замеры времени работы каждого из исследуемых алгоритмов

3.2 Выбор языка программирования

Был выбран язык Go, поскольку он удовлетворяет требованиям задания. Средой разработки выбрана Visual Studio Code.

3.3 Структуры данных

На листинге 3.1 представлено описание структуры массива.

Листинг 3.1: Структура массива

```
type vector_t struct{
  elem[] float32
  size int
}
```

На листинге 3.2 представлено описание структуры матрицы.

Листинг 3.2: Структура матрицы

```
type matrix_t struct{
  elem [][] float32
  rows int
  columns int
}
```

3.4 Реализация алгоритмов

На листинге 3.3 представлена реализация алгоритма стандартного умножения матриц.

Листинг 3.3: Реализация стандартного алгоритма умножения матриц

```
func multiplicate_matrix_norm(m1 matrix_t, m2 matrix_t, rc *bool)matrix_t{
  var m3 matrix_t = make_empty_matrix(m1.rows, m2.columns)
  *rc = true
  if (m2.rows != m1.columns){
    *rc = false
    return m3;
  }
  for i := 0;i < m1.rows;i++{
    for j := 0; j < m2.columns;j++{
      for k := 0; k < m1.columns;k++{
        m3.elem[i][j] += m1.elem[i][k] * m2.elem[k][j]
      }
    }
  }
  return m3
}
```

На листинге 3.4 представлена реализация алгоритма Винограда.

Листинг 3.4: Реализация алгоритма Винограда

```
func multiplicate_matrix_vinograd(m1 matrix_t, m2 matrix_t, rc *bool)
    matrix_t{
  var m3 matrix_t = make_empty_matrix(m1.rows, m2.columns)
```

```
  *rc = true
  if (m2.rows != m1.columns){
    *rc = false
    return m3;
  }

  var rows_factor vector_t = make_empty_vector(m1.rows)
  var columns_factor vector_t = make_empty_vector(m2.columns)

  for i:=0;i<m1.rows;i++{
    for j:=1;j<m1.columns;j+=2{
      rows_factor.elem[i] += m1.elem[i][j - 1] * m1.elem[i][j]
    }
  }
  for i:=0;i<m2.columns;i++{
    for j:=1;j<m2.rows;j+=2{
      columns_factor.elem[i] += m2.elem[j - 1][i] * m2.elem[j][i]
    }
  }

  for i := 0;i < m1.rows;i++{
    for j := 0; j < m2.columns;j++{
      m3.elem[i][j] = - rows_factor.elem[i] - columns_factor.elem[j]
      for k := 0; k < m1.columns/2;k++{
        m3.elem[i][j] += (m1.elem[i][k*2] + m2.elem[2*k+1][j])*(m1.elem[i][
            k*2+1] + m2.elem[2*k][j])
      }
    }
  }

  if (m1.columns % 2 == 1){
    for i:= 0; i<m1.rows;i++{
      for j:=0;j<m2.columns;j++{
        m3.elem[i][j] += m1.elem[i][m1.columns-1]*m2.elem[m1.columns - 1][j
            ]
      }
    }
```

```
  }

  return m3
}
```

На листинге 3.5 представлена реализация оптимизированного алгоритма Винограда.

Листинг 3.5: Реализация оптимизированного алгоритма Винограда

```
func multiplicate_matrix_fast_vinograd(m1 matrix_t, m2 matrix_t, rc *bool)
   matrix_t{
  var m3 matrix_t = make_empty_matrix(m1.rows, m2.columns)
  *rc = true
  if (m2.rows != m1.columns){
    *rc = false
    return m3;
  }

  var rows_factor vector_t = make_empty_vector(m1.rows)
  var columns_factor vector_t = make_empty_vector(m2.columns)

  for i:=0;i<m1.rows;i++{
    for j:=1;j<m1.columns;j+=2{
      rows_factor.elem[i] += m1.elem[i][j - 1] * m1.elem[i][j]
    }
  }
  for i:=0;i<m2.columns;i++{
    for j:=1;j<m2.rows;j+=2{
      columns_factor.elem[i] += m2.elem[j - 1][i] * m2.elem[j][i]
    }
  }

  var flag bool = false
  if (m1.columns % 2 == 1){
    flag = true
  }

  for i := 0;i < m1.rows;i++{
```

```
    for j := 0; j < m2.columns; j++{
      m3.elem[i][j] = - rows_factor.elem[i] - columns_factor.elem[j]
      for k := 0; k < m1.columns/2; k++{
        m3.elem[i][j] += (m1.elem[i][k*2] + m2.elem[2*k+1][j])*(m1.elem[i][
            k*2+1] + m2.elem[2*k][j])
      }
      if (flag){
        m3.elem[i][j] += m1.elem[i][m1.columns-1]*m2.elem[m1.columns - 1][j
            ]
      }
    }
  }
  return m3
}
```

3.5 Тестирование

Модульные тесты

Таблица 3.1 – Тесты (матричное представление)

№	Матрица 1	Матрица 2	Произведение матриц 1 и 2
1	$\begin{array}{c\|c\|c\|c} 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 \end{array}$	$\begin{array}{c\|c\|c\|c} 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 \end{array}$	Умножение не возможно
2	$\begin{array}{c\|c\|} 1 & 2 \\ \hline 3 & 4 \end{array}$	$\begin{array}{c\|c\|} 1 & 2 \\ \hline 3 & 4 \end{array}$	$\begin{array}{c\|c\|} 7 & 10 \\ \hline 15 & 22 \end{array}$
3	$\begin{array}{c\|c\|c\|} 1 & 2 & 3 \\ \hline 4 & 5 & 6 \end{array}$	$\begin{array}{c\|} 1 \\ \hline 2 \\ \hline 3 \end{array}$	$\begin{array}{c\|} 14 \\ \hline 32 \end{array}$

Таблица 3.2 – Тесты (ввод числами)

№	Матрица 1	Матрица 2	Произведение матриц 1 и 2
1	4 2 1 1 1 1 1 1 1 1	4 2 1 1 1 1 1 1 1 1	Умножение не возможно
2	2 2 1 2 3 4	2 2 1 2 3 4	2 2 7 10 15 22
3	2 3 1 2 3 4 5 6	3 1 1 2 3	2 1 14 32

3.6 Вывод технологической части

Были реализованы исследуемые алгоритмы, программа прошла тесты и удовлетворяет требованиям.

4. Экспериментальная часть

Оценка качества работы алгоритмов. Экспериментальное сравнение работы различных алгоритмов умножения матриц (зависимость времени выполнения от размерности матриц).

4.1 Примеры работы

На рисунках 4.1, 4.2, 4.3, 4.4, 4.5 показаны примеры работы.

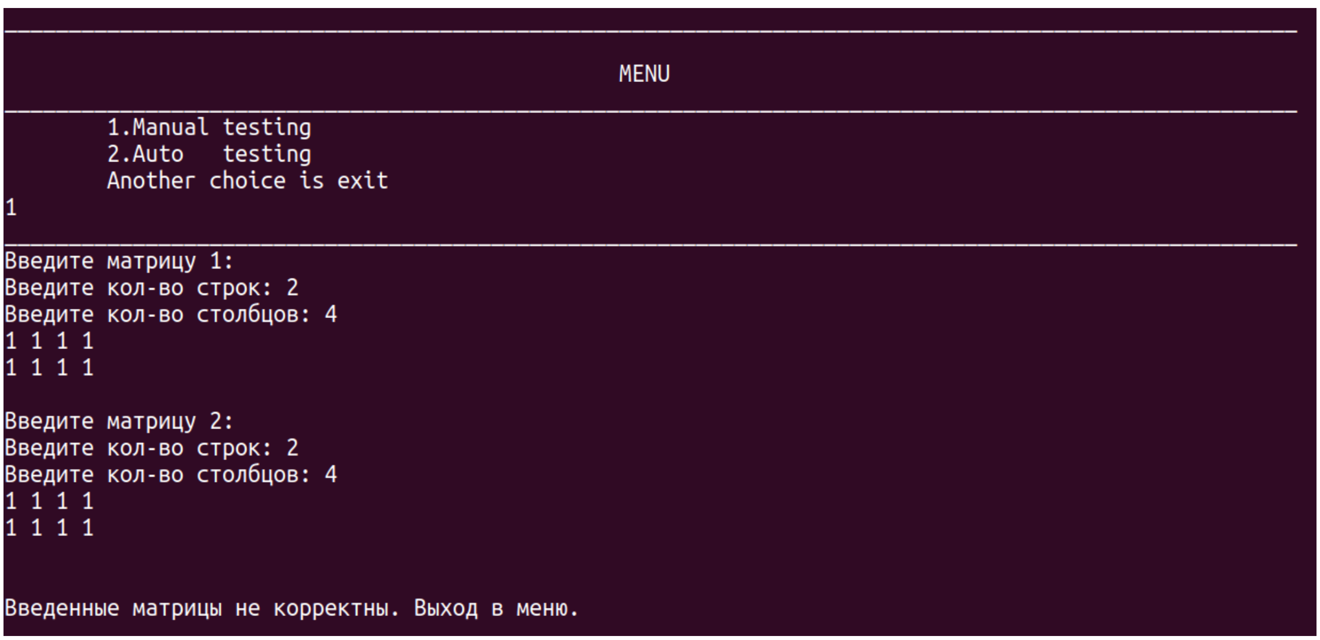
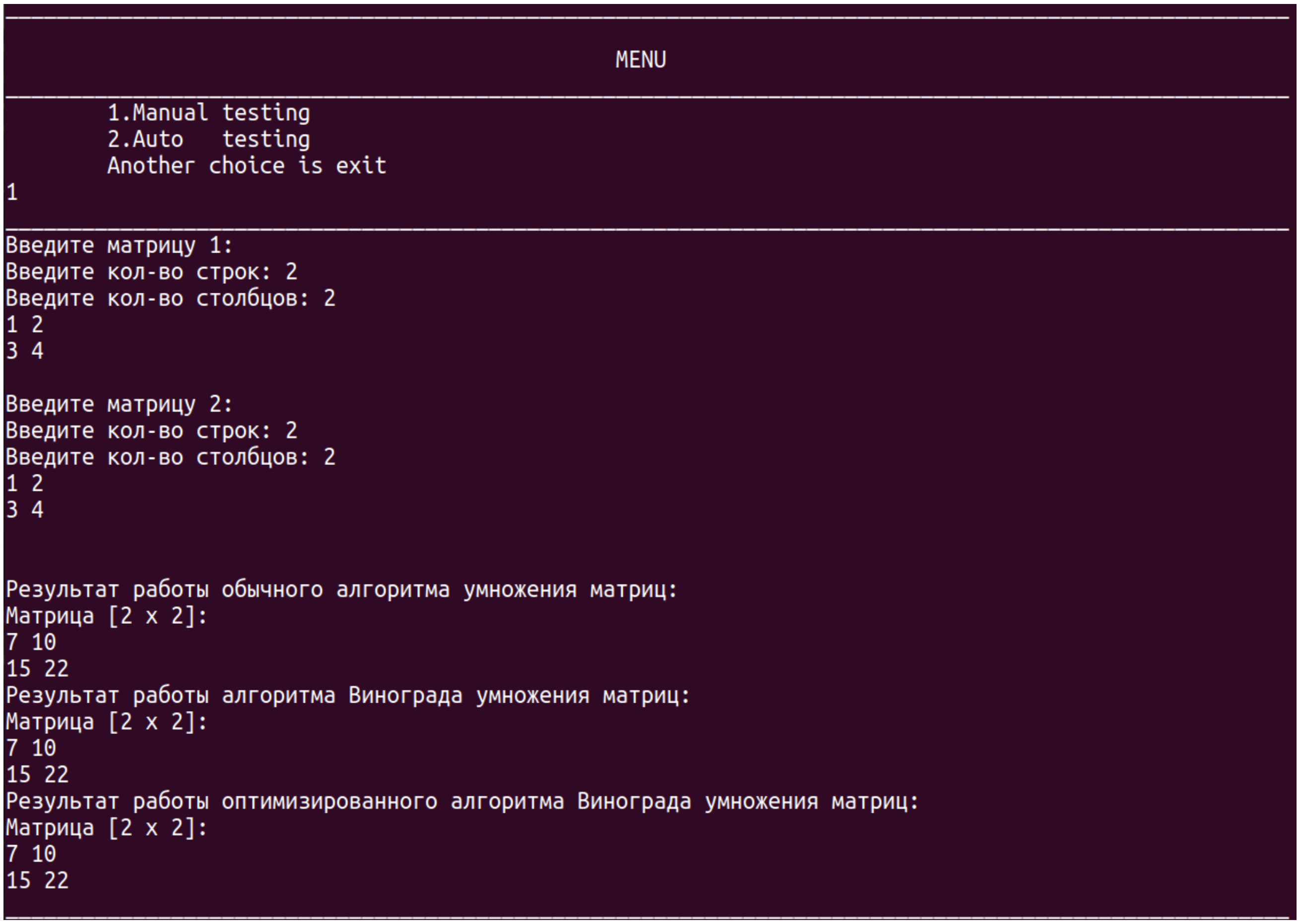


Рис. 4.1: Ручное тестирование: тест 1



```
MENU
        1.Manual testing
        2.Auto   testing
        Another choice is exit
1
Введите матрицу 1:
Введите кол-во строк: 2
Введите кол-во столбцов: 2
1 2
3 4

Введите матрицу 2:
Введите кол-во строк: 2
Введите кол-во столбцов: 2
1 2
3 4


Результат работы обычного алгоритма умножения матриц:
Матрица [2 x 2]:
7 10
15 22
Результат работы алгоритма Винограда умножения матриц:
Матрица [2 x 2]:
7 10
15 22
Результат работы оптимизированного алгоритма Винограда умножения матриц:
Матрица [2 x 2]:
7 10
15 22
```

Рис. 4.2: Ручное тестирование: тест 2

```
MENU

        1.Manual testing
        2.Auto   testing
        Another choice is exit
1

Введите матрицу 1:
Введите кол-во строк: 2
Введите кол-во столбцов: 3
1 2 3
4 5 6

Введите матрицу 2:
Введите кол-во строк: 3
Введите кол-во столбцов: 1
1
2
3


Результат работы обычного алгоритма умножения матриц:
Матрица [2 x 1]:
14
32
Результат работы алгоритма Винограда умножения матриц:
Матрица [2 x 1]:
14
32
Результат работы оптимизированного алгоритма Винограда умножения матриц:
Матрица [2 x 1]:
14
32
```

Рис. 4.3: Ручное тестирование: тест 3

```
MENU

        1.Manual testing
        2.Auto   testing
        Another choice is exit
2

Четный размер матриц

|  N|   Обычный алг.|       Виноград|   Оптимиз. Виноград|
| 10|      21.172000|      20.630000|           24.376000|
| 20|      67.394000|      36.970000|           29.146000|
| 30|     153.616000|     113.344000|           94.082000|
| 40|     336.304000|     251.928000|          215.302000|
| 50|     606.904000|     520.060000|          412.980000|
| 60|    1038.550000|     873.250000|          701.404000|
| 70|    1686.466000|    1350.374000|         1186.360000|
| 80|    2565.432000|    2061.870000|         1775.104000|
| 90|    3595.638000|    2763.956000|         2427.748000|
|100|    5007.276000|    3804.958000|         3368.530000|

Нечетный размер матриц

|  N|   Обычный алг.|       Виноград|   Оптимиз. Виноград|
| 11|       7.390000|       8.434000|            5.738000|
| 21|      52.864000|      43.512000|           34.856000|
| 31|     151.326000|     138.640000|          112.108000|
| 41|     342.884000|     280.384000|          226.714000|
| 51|     643.186000|     552.102000|          437.224000|
| 61|    1097.358000|     909.050000|          744.520000|
| 71|    1741.564000|    1396.738000|         1156.816000|
| 81|    2675.552000|    2165.284000|         1867.594000|
| 91|    3791.614000|    2912.026000|         2537.368000|
|101|    5126.074000|    3907.792000|         3425.130000|
```

Рис. 4.4: Автотестирование

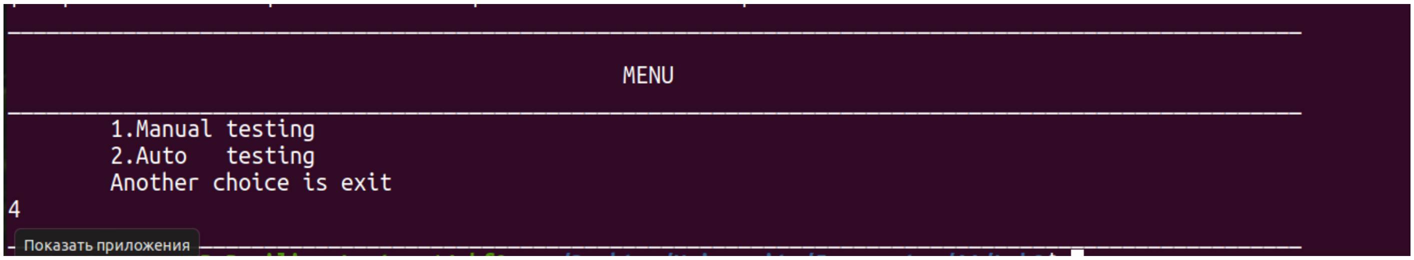


Рис. 4.5: Выход

4.2 Замеры времени

На рисунках 4.6, 4.7, и 4.8 и 4.9 показаны графические результаты сравнения исследуемых алгоритмов по времени.

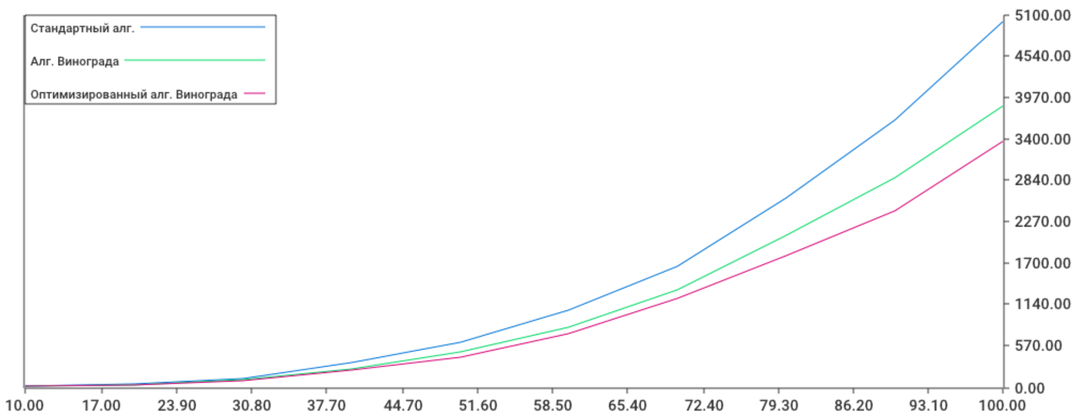


Рис. 4.6: Сравнение 3 исследуемых алгоритмов по времени (размер матрицы четный)

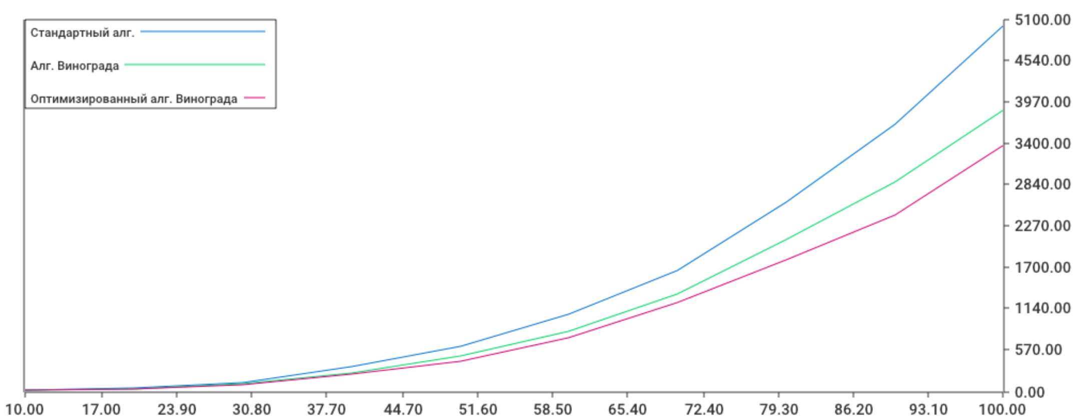


Рис. 4.7: Сравнение 3 исследуемых алгоритмов по времени (размер матрицы нечетный)

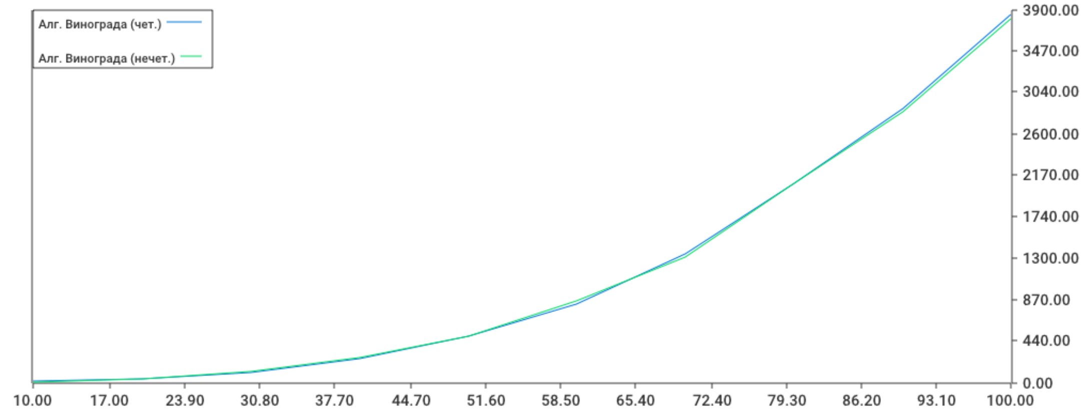


Рис. 4.8: Сравнение алгоритма Винограда четные и нечетные размеры матриц

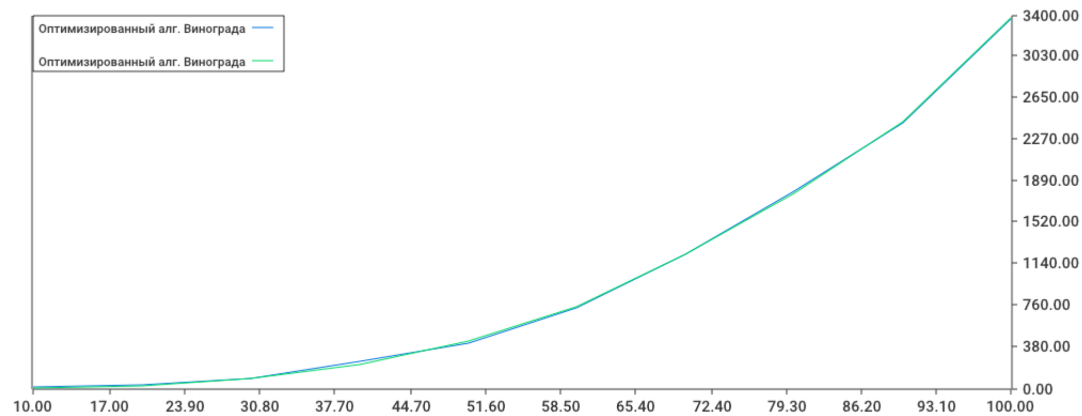


Рис. 4.9: Сравнение оптимизированного алгоритма Винограда четные и нечетные размеры матриц

4.3 Сравнительный анализ алгоритмов

Из проведенных замеров видно, что оптимизация алгоритма Винограда получилась эффективнее по времени, чем сам алгоритм Винограда, следовательно предположение, сделанное из трудоемкостей алгоритмов, верно. Самым выгодным по времени алгоритмом из исследуемых является оптимизированный алгоритм Винограда. Самым медленным из исследуемых алгоритмов оказался стандартный алгоритм, как и было предположено. Различия в скорости при обработке матриц четных и нечетных размеров незначительны, как это видно из графиков 4.8 и 4.9, как для обычного алгоритма Винограда, так и для оптимизированного.

4.4 Вывод экспериментальной части

Таким образом, подтвердилось предположение, что самый эффективный алгоритм из рассматриваемых в работе - оптимизированный алгоритм Винограда. Различия в скорости при обработке матриц четных и нечетных размеров незначительны, как для обычного алгоритма Винограда, так и для оптимизированного.

5. Заключение

В данной работе были изучены алгоритмы умножения матриц. Получены практические навыки реализации исследуемых алгоритмов. Была подсчитана трудоемкость исследуемых алгоритмов. Проведён сравнительный анализ алгоритмов по времени и трудоемкости. Показано, что наименее трудоемкий и наименее затратный по времени при больших размерах матриц алгоритм Винограда оптимизированный. Экспериментально подтверждены различия в эффективности алгоритмов с указанием лучших и худших случаев. Цель работы достигнута, решены поставленные задачи. Показано, что наименее трудоемкий и наименее затратный по времени при больших размерах матриц алгоритм Винограда оптимизированный. Получены практические навыки реализации алгоритмов Винограда и стандартного алгоритма, а также проведена исследовательская работа по оптимизации и вычислении трудоемкости алгоритмов.

Список литературы

[1] Gratzer George A. More Math Into LaTeX. 4th изд. Boston: Birkhauser, 2007.

[2] Go Documentation [Электронный ресурс]. Режим доступа: https://go.dev/doc/. Дата обращения: 20.09.2020.

[3] Debian – универсальная операционная система [Электронный ресурс]. Режим доступа: https://www.debian.org/. Дата обращения: 20.09.2020.

[4] Linux – Getting Started [Электронный ресурс]. Режим доступа: https://linux.org. Дата обращения: 20.09.2020.

[5] Открытый урок. Первое сентября. [Электронный ресурс]. Режим доступа: https://urok.1sept.ru/articles/637896. Дата обращения: 03.12.2021.

[6] ПОИВС [Электронный ресурс]. Режим доступа: http://poivs.tsput.ru/ru/Math/Algebra/LinearAlgebra/Matrices. Дата обращения: 03.12.2021.