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 Алгоритмы умножения матриц

Студент Ковалев Д.А.

Группа ИУ7-53Б

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

Москва — 2020 г.

Оглавление

Введение

Цель работы: изучение алгоритмов умножения матриц. В данной лабораторной работе рассматривается стандартный алгоритм умножения матриц, алгоритм Винограда и модифицированный алгоритм Винограда. Также требуется изучить рассчет сложности алгоритмов, получить навыки в улучшении алгоритмов.

Матрицей A размера $[m * n]$ называется таблица элементов, расположение которых определяется при помощи порядкового номера столбца и строки. Важнейшие характеристики матрицы - число строк и число столбцов. Сами числа называют элементами матрицы и характеризуют их положением в матрицы, задавая номер строки и столбца и записывая их в виде двойного индекса, причем вначале записывают номер строки, затем номер столбца. Пусть есть два конечных множества.

- номера строк: M = 1, 2, ..., n;
- Номера столбцов N = 1, 2, ..., m;

где m и n - натуральные числа. Тогда матрица A представлена на рисунке 1:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{pmatrix}$$

Рис. 1: Сравнение времени работы при разном размере матрицы

Простейшими действиями над матрицами являются следующие операции.

1. Умножение матрицы на число. Для этого необходимо умножить каждый элемент матрицы на данное число.
2. Сложение матриц. Складывать можно только матрицы одинакового размера, то есть, имеющие одинаковое число строк и одинаковое число столбцов. При сложении матриц соответствующие их элементы складываются;

3. Транспонирование матрицы. При транспонировании у матрицы строки становятся столбцами и наоборот.

В ходе лабораторной работы предстоит:

- изучить алгоритмы умножения матриц: стандартный алгоритм и алгоритм Винограда;
- оптимизировать алгоритм Винограда;
- дать теоретическую оценку базового алгоритма умножения матриц, алгоритма Винограда и улучшенного алгоритма Винограда;
- реализовать три алгоритма умножения матриц на одном из языков программирования;
- сравнить алгоритмы умножения матриц.

1. Аналитическая часть

Произведение матриц AB состоит из всех возможных комбинаций скалярных произведений вектор-строк матрицы A и вектор-столбцов матрицы B. Операция умножения двух матриц выполнима только в том случае, если число столбцов в первой матрицы равно числу строк во второй.

1.1 Стандартный алгоритм

Пусть даны две прямоугольные матрицы

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

тогда матрица C

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

где

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

будет называться произведением матриц A и B. Стандартный алгоритм реализует данную формулу.

1.2 Алгоритм Винограда

Подход Алгоритма Винограда является иллюстрацией общей методологии, начатой в 1979-х годах на основе билинейных и трилинейных форм, благодаря которым большинство усовершенствований для умножения матриц были получены. [1].

Рассмотрим два вектора $V = (v1, v2, v3, v4)$ и $W = (w1, w2, w3, w4)$.

Их скалярное произведение равно (1.4)

$$V \cdot W = v_1 \cdot w_1 + v_2 \cdot w_2 + v_3 \cdot w_3 + v_4 \cdot w_4 \tag{1.4}$$

Равенство (1.4) можно переписать в виде (1.5)

$$V \cdot W = (v_1 + w_2) \cdot (v_2 + w_1) + (v_3 + w_4) \cdot (v_4 + w_3) - v_1 \cdot v_2 - v_3 \cdot v_4 - w_1 \cdot w_2 - w_3 \cdot w_4 \tag{1.5}$$

Менее очевидно, что 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. Это означает, что над предварительно обработанными элементами нам придется выполнять лишь первые два умножения и последующие пять сложений, а также дополнительно два сложения.

1.3 Вывод

Были рассмотрены алгоритмы классического умножения матриц и алгоритм Винограда, основное отличие которых — наличие предварительной обработки, а также количество операций умножения.

2. Конструкторская часть

Требования к вводу: На вход подаются две матрицы

Требования к программе:

- корректное умножение двух матриц;
- при матрицах неправильных размеров программа не должна аварийно завершаться.

2.1 Схемы алгоритмов

В данной части будут рассмотрены схемы алгоритмов.

На рисунке 2.1 представлен алгоритм классического умножения матриц.

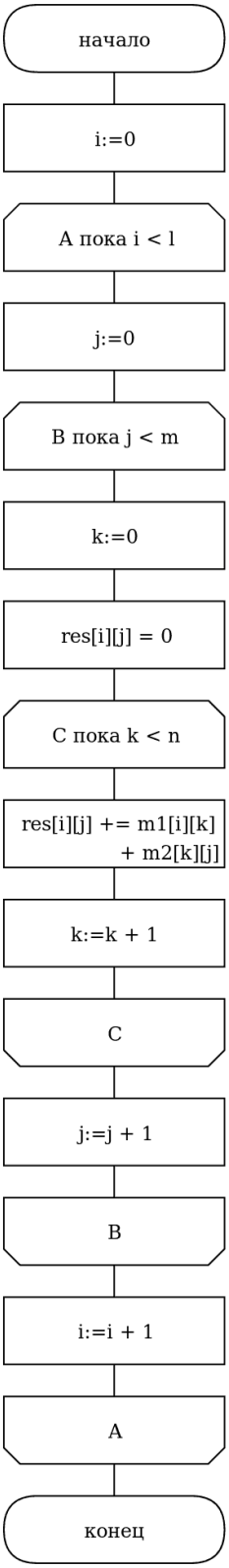

Рис. 2.1: Схема классического умножения матриц

На рисунках 2.2 - 2.3 представлен алгоритм умножения матриц по Винограду.

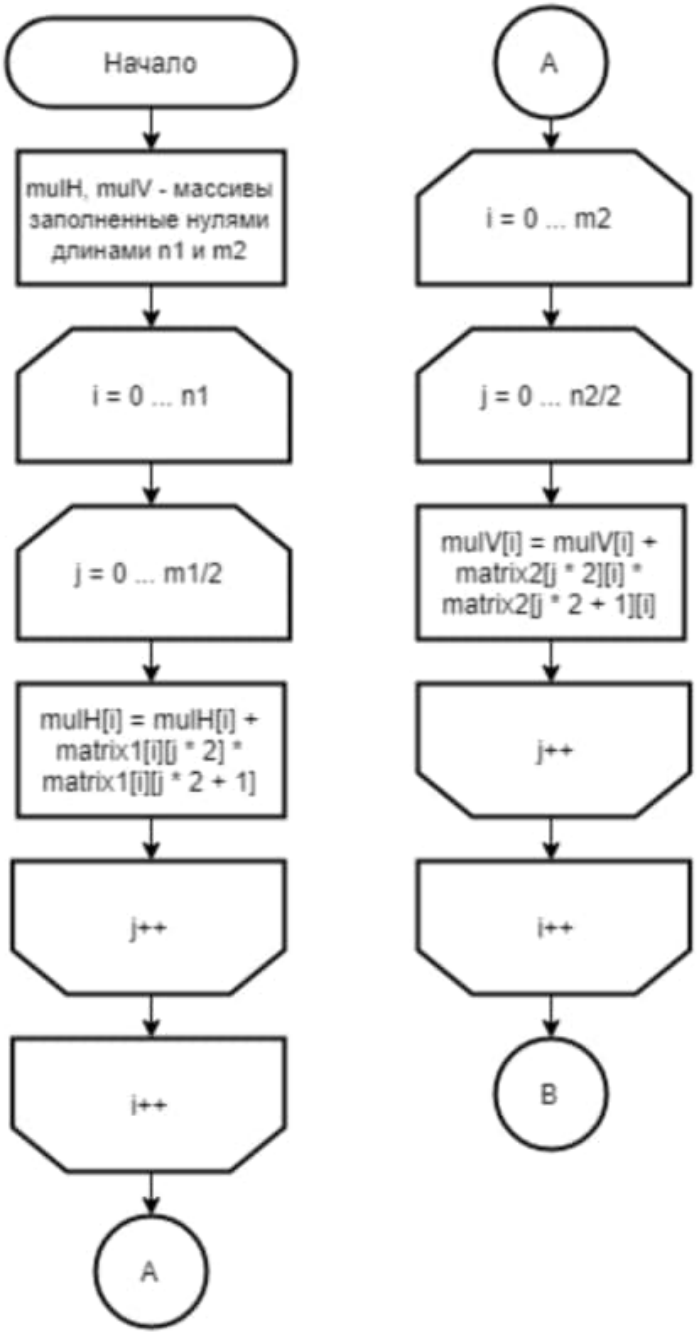

Рис. 2.2:

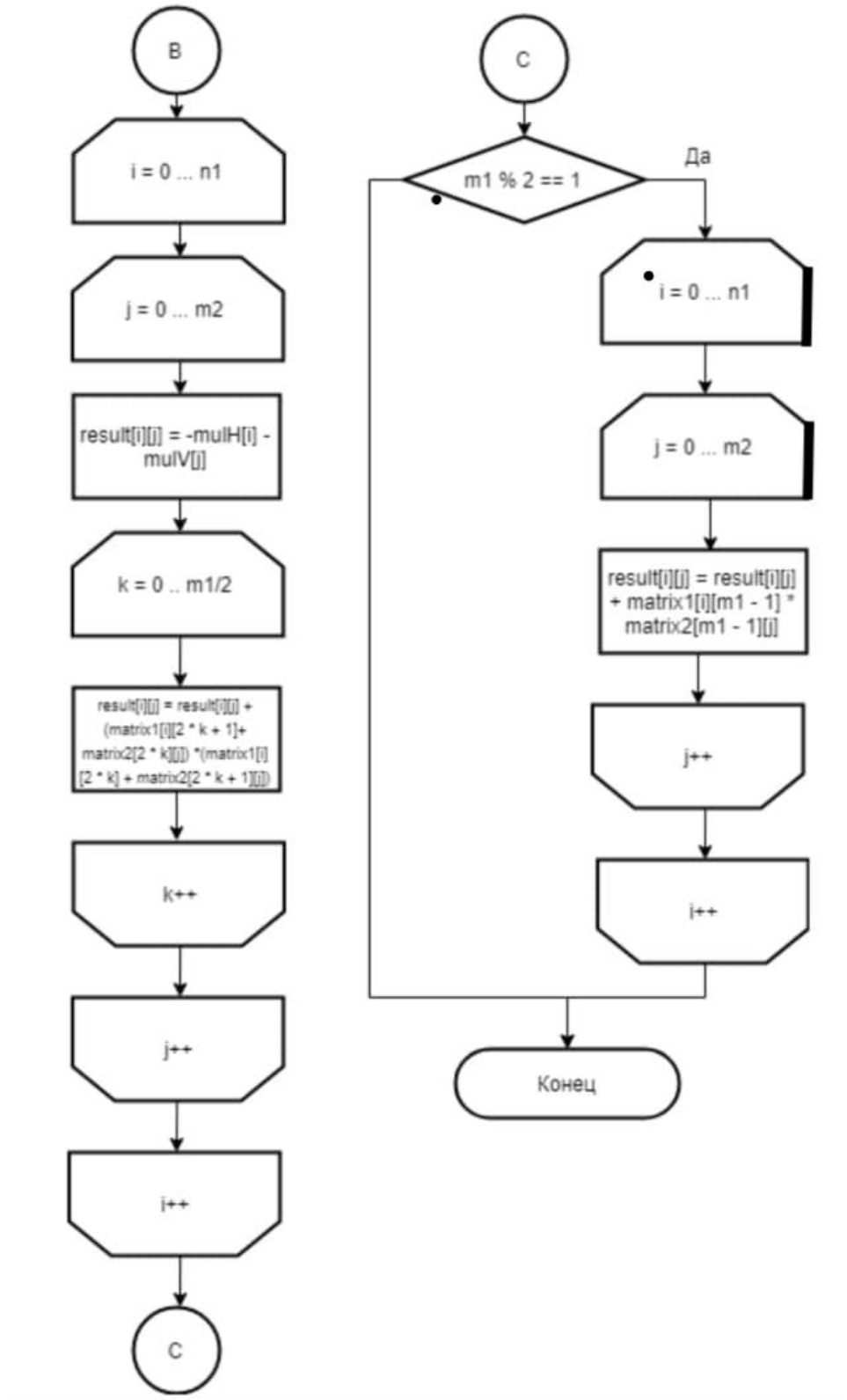

Рис. 2.3: Схема алгоритма Винограда

На рисунках 2.4 - 2.5 представлен оптимизированный алгоритм умножения матриц по Винограду.

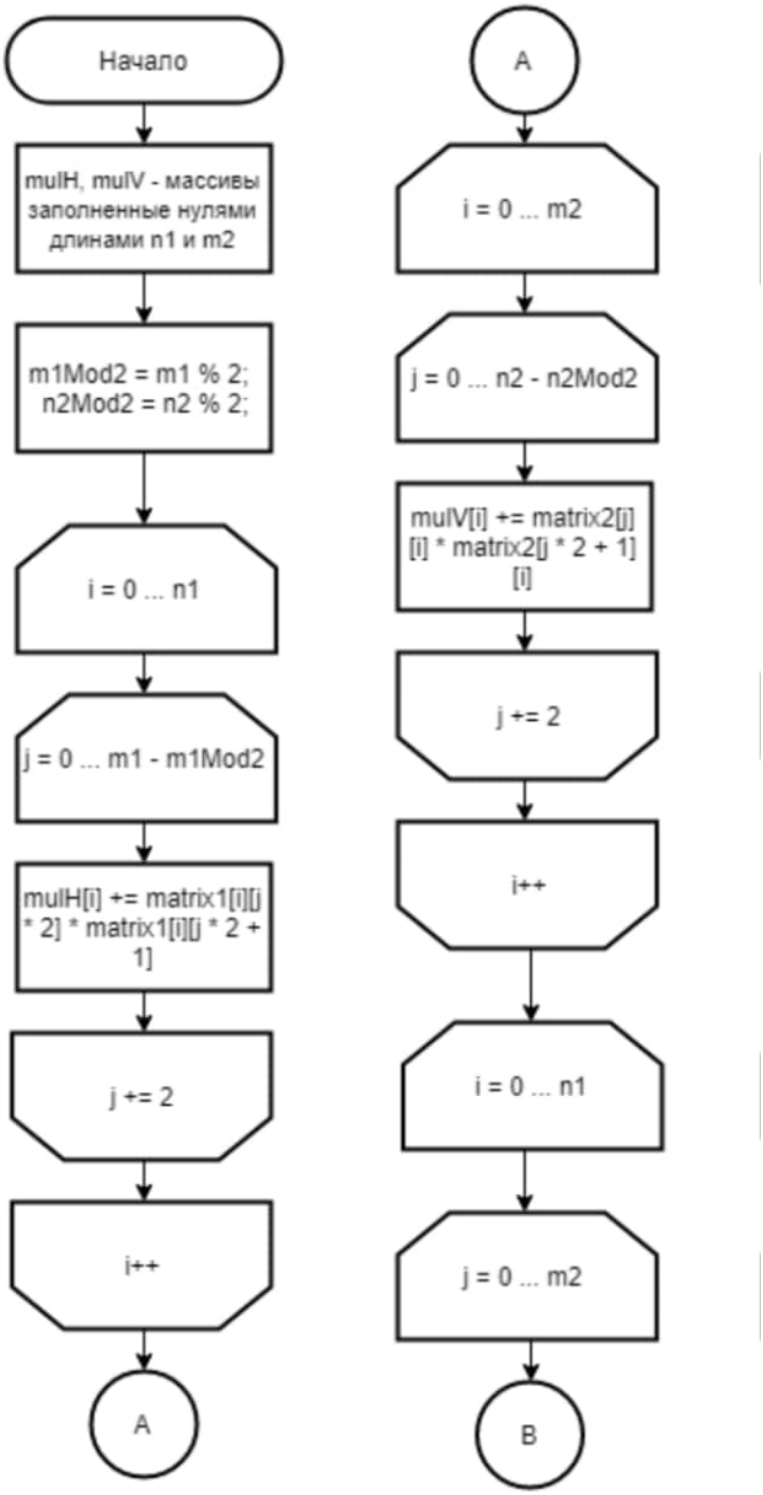

Рис. 2.4:

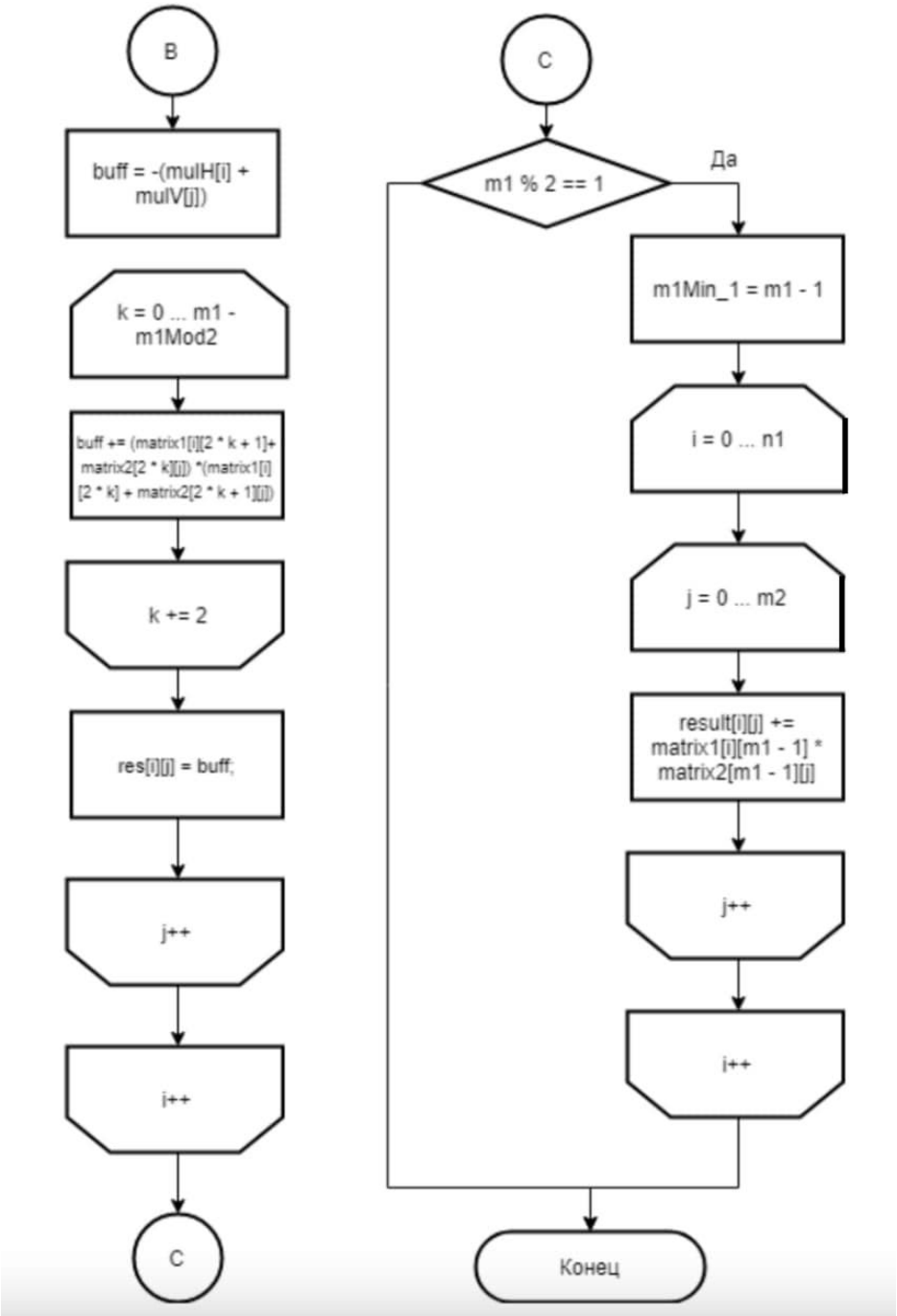

Рис. 2.5: Схема оптимизированного алгоритма Винограда

3. Технологическая часть

3.1 Выбор ЯП

В качестве языка программирования был выбран Golang, Goland была выбрана в качестве среды разработки. [2] [3] Такой выбор средств выполнения лабораторной работы был связан с наличием в языке Golang системы тестирования "Benchmark которая позволяет быстро и качественно произвести тестирование программы и замерить время с высокой точностью.

3.2 Сведения о модулях программы

Программа состоит из:

- main.go - главный файл программы, в котором располагается точка входа в программу и функции перемножения матриц.
- algh_test.go - файл с benchmark тестами.

3.3 Листинг кода алгоритмов

В листинге 3.1 представлена функция инициализации матрицы.

Листинг 3.1: Инициализация матрицы

```
1 func createMatrix(row, col int) [][]int {
2   matrix := make([][]int, row)
3   for i := range matrix {
4     matrix[i] = make([]int, col)
5   }
6   return matrix
7 }
```

В листинге 3.2 представлена функция стандартного умножения матриц.

Листинг 3.2: Стандартный алгоритм умножения матриц

```
func StandartMult(m1, m2 [][]int) [][]int {
  if (len(m1) == 0 || len(m2) == 0) || (len(m1[0]) != len(m2)) {
    return nil
  }
  res := createMatrix(len(m1), len(m2[0]))
  for i := 0; i < len(m1); i++ {
    for j := 0; j < len(m2[0]); j++ {
      for k := 0; k < len(m2); k++ {
        res[i][j] += m1[i][k] * m2[k][j]
      }
    }
  }
  return res
}
```

В листинге 3.3 представлена функция умножения матриц по Винограду.

Листинг 3.3: Алгоритм Винограда

```
func VinogradMult(m1, m2 [][]int) [][]int {
  if (len(m1) == 0 || len(m2) == 0) || (len(m1[0]) != len(m2)) {
    return nil
  }
  res := createMatrix(len(m1), len(m2[0]))
  rowF := make([]int, len(m1))
  colF := make([]int, len(m2[0]))

  for i := 0; i < len(m1); i++ {
    for j := 0; j < len(m1[0])/2; j++ {
      rowF[i] += m1[i][j*2] * m1[i][j*2+1]
    }
  }

  for i := 0; i < len(m2[0]); i++ {
```

```
16      for j := 0; j < len(m2)/2; j++ {
17        colF[i] += m2[j*2][i] * m2[j*2+1][i]
18      }
19    }
20
21    for i := 0; i < len(m1); i++ {
22      for j := 0; j < len(m2[0]); j++ {
23        res[i][j] = -rowF[i] - colF[j]
24        for k := 0; k < len(m1[0])/2; k++ {
25          res[i][j] += (m1[i][2*k+1] + m2[2*k][j]) * (m1[i][2*k] + m2[2*k+1][j
                ])
26        }
27      }
28    }
29
30    if len(m1[0])%2 == 1 {
31      for i := 0; i < len(m1); i++ {
32        for j := 0; j < len(m2[0]); j++ {
33          res[i][j] += m1[i][len(m1[0])-1] * m2[len(m1[0])-1][j]
34        }
35      }
36    }
37    return res
38  }
```

В листинге 3.4 представлен код функции оптимизированного умножения матриц по Винограду.

Листинг 3.4: Оптимизированный алгоритм Винограда

```
1  func VinOptimMult(matrix1 [][]int, matrix2 [][]int) [][]int {
2    var n1 int = len(matrix1)
3    var n2 int = len(matrix2)
4
5    if n1 == 0 || n2 == 0 {
```

```
6       return nil
7     }
8
9     var m1 int = len(matrix1[0])
10    var m2 int = len(matrix2[0])
11
12    if m1 != n2 {
13      return nil
14    }
15
16    mulH := make([]int, n1)
17    mulV := make([]int, m2)
18    result := createMatrix(n1, m2)
19
20    var m1Mod2 int = m1 % 2
21    var n2Mod2 int = n2 % 2
22
23    for i := 0; i < n1; i++ {
24      for j := 0; j < m1-m1Mod2; j += 2 {
25        mulH[i] += matrix1[i][j] * matrix1[i][j+1]
26      }
27    }
28
29    for i := 0; i < m2; i++ {
30      for j := 0; j < n2-n2Mod2; j += 2 {
31        mulV[i] += matrix2[j][i] * matrix2[j+1][i]
32      }
33    }
34
35    var buff int
36    for i := 0; i < n1; i++ {
37      for j := 0; j < m2; j++ {
38        buff = -mulH[i] - mulV[j]
```

```
39          for k := 0; k < m1-m1Mod2; k += 2 {
40            buff += (matrix1[i][k+1] + matrix2[k][j]) * (matrix1[i][k] + matrix2
                  [k+1][j])
41          }
42          result[i][j] = buff
43        }
44      }
45
46      if m1Mod2 == 1 {
47        var m1Min1 int = m1 - 1
48        for i := 0; i < n1; i++ {
49          for j := 0; j < m2; j++ {
50            result[i][j] += matrix1[i][m1Min1] * matrix2[m1Min1][j]
51          }
52        }
53      }
54
55      return result
56  }
```

3.4 Вывод

В данном разделе мы рассмотрели листинг программы и структуру программы.

4. Исследовательская часть

4.1 Сравнительный анализ на основе замеров времени работы алгоритмов

Был проведен замер времени работы каждого из алгоритмов. Замер производился на ноутбуке ThinkPad x280 на базе процессоре Intel core i5, который обладает 1.4 GHz тактовой частоты, а также 8 гигабайтами оперативной памяти.[4][5] Для произведения замера времени использовалась система тестирования Benchmark языка Golang. Данная система тестирования позволяет измерить скорость работы алгоритма с минимальной погрешностью.

На рисунке 4.1 представлена сравнительная характеристика времени работы программы при четных размерах матрицы.

Разм. матриц	Станд.	Виноград	Виноград(опт)
100	0.2416320	0.2362540	0.1908588
110	0.3123829	0.2860229	0.2532279
120	0.4027150	0.3842103	0.3294053
130	0.5241411	0.4977670	0.4920838
140	0.6389620	0.6099429	0.5358248
150	0.7846432	0.7222561	0.6594992
160	0.9597850	0.9241459	0.7794201
170	1.1764548	1.1488168	0.9409909
180	1.3599980	1.2331799	1.2199771
190	1.6334410	1.5685019	1.3010969
200	1.8603230	1.8250951	1.5311081

Рис. 4.1: Сравнительная характеристика времени работы программы при четных размерах матрицы

Первый эксперимент производится для лучшего случая на квадратных матрицах размером от 100 x 100 до 1000 x 1000 с шагом 100. Сравним результаты для разных алгоритмов:

На рисунке 4.2 представлено сравнение времени работы при разном размере матрицы.

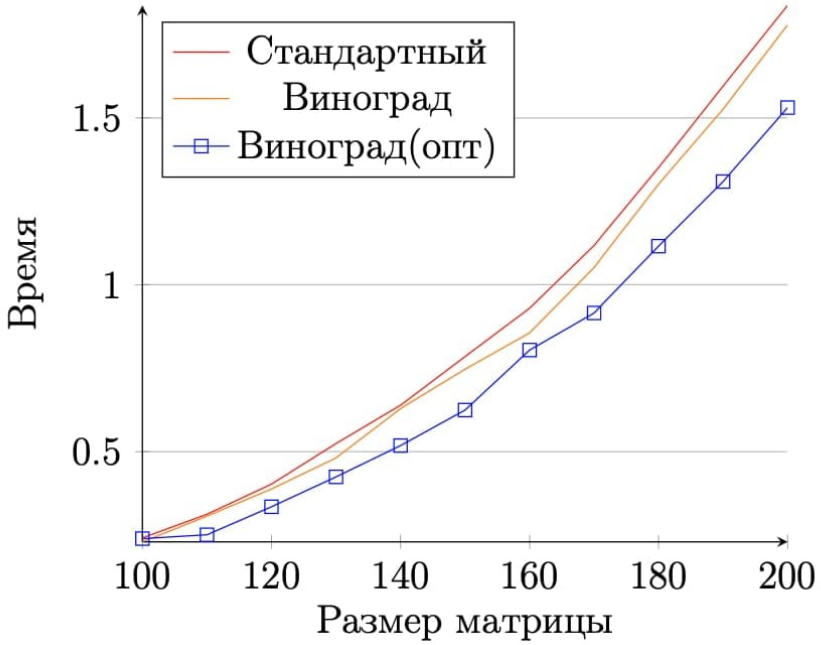

Рис. 4.2: Сравнение времени работы при разном размере матрицы

4.2 Вывод

По результатам тестирования, а также оценке времени работы, данные алгоритмы реализованы верно. Самым медлен По результатам тестирования все рассматриваемые алгоритмы реализованы правильно. Самым медленным алгоритмом оказался алгоритм классического умножения матриц, а самым быстрым — оптимизированный алгоритм Винограда.

Литература

[1] Stein C. Introduction to Algorithms, 3rd Edition. – M.: The MIT Press, 2009. Vol. 1251. P. 102–143.

[2] Саммерфильд Марк. Программирование на Go. Разработка приложений XXI века. –М.: ДМК Пресс, 2013, 2013. Т. 580. С. 130–216.

[3] Среда разработки Goland. Режим доступа: https://www.jetbrains.com/go/promo/ (дата обращения: 29.9.2020).

[4] Технические характеристики ноутбука Apple MacBook Pro. Режим доступа: https://www.apple.com/macbook-pro-13/ (дата обращения: 5.10.2020).

[5] Процессор Intel® Core™ i5 10 gen. [Электронный ресурс]. Режим доступа: https://https://www.intel.ru/content/www/ru/ru/ products/docs/processors/core/ 10th-gen-processors.html (дата обращения: 30.09.2020).