|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 **ПРОГРАМНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.03 ПРОГРАММНАЯ ИНЖЕНЕРИЯ**

**Отчет**

|  |  |
| --- | --- |
| **По лабораторной работе №** | 2 |

**Название:**

Алгоритмы умножения матриц

**Дисциплина:** Анализ Алгоритмов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-52Б |  |  | Н.А. Гарасев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Л.Л. Волкова |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

Оглавление

[Введение 3](#_Toc52317386)

[1. Аналитическая часть 4](#_Toc52317389)

[2. Конструкторская часть 6](#_Toc52317390)

[2.1. Схемы алгоритмов 6](#_Toc52317391)

[3. Технологическая часть 10](#_Toc52317392)

[3.1. Реализация алгоритмов 10](#_Toc52317393)

[3.2. Тестирование функций 12](#_Toc52317394)

[4. Экспериментальная часть 13](#_Toc52317395)

[4.1. Интерфейс 13](#_Toc52317396)

[4.2. Сравнение алгоритмов по времени работы реализаций 15](#_Toc52317397)

[Заключение 21](#_Toc52317398)

[Список литературы 22](#_Toc52317399)

# Введение

Цель лабораторной работы: изучить и применить на практике алгоритмы умножения матриц. В данной лабораторной работе рассматривается стандартный алгоритм умножения матриц, алгоритм Винограда и модифицированный (оптимизированный) Винограда. Также требуется изучить расчет сложности алгоритмов, получить навыки в улучшении алгоритмов.

Алгоритмы перемножения матриц активно применяются во всех областях, где используется линейная алгебра.

1. В компьютерной графике.
2. В физике.
3. В экономике.

## В ходе выполнения лабораторной работы требуется решить следующие задачи.

* 1. Изучить алгоритмы умножения матриц: стандартный и алгоритм Винограда.
  2. Реализовать стандартный алгоритм.
  3. Реализовать алгоритм Винограда.
  4. Реализовать улучшенную версию алгоритма Винограда.
  5. Дать теоретическую оценку всем вышеупомянутым алгоритмам.
  6. Сравнить алгоритмы по затраченным ресурсам (времени и памяти).
  7. Проанализировать полученные результаты и сделать вывод о временной и емкостной эффективности реализованных алгоритмов.

# Аналитическая часть

Умножение матриц — одна из основных операций над матрицами. Матрица, получаемая в результате операции умножения, называется произведением матриц. Матрицы **A** и **B** могут быть перемножены, если они совместимы в том смысле, что число столбцов матрицы **A** равно числу строк **B**.

* 1. **Классическое умножение матриц**

Ниже на рис. 1 представлена формула умножения. Пусть даны две прямоугольные матрица **A** и **B** размерами n \* m и m \* k соответственно. Тогда матрица **C** имеет размеры n \* k и является произведением матриц **A** и **B**, каждый элемент сij которой считается по формуле, представленной на рис. 1.

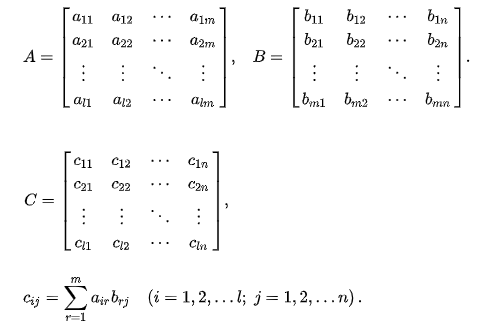


Рисунок 1. *Перемножение матриц A и B дает в результате матрицу C*

* 1. **Алгоритм Винограда**

Рассмотрим два вектора V = (v1, v2, v3, v4) и W = (w1, w2, w3, w4). Их скалярное произведение равно:

V • W = v1w1 + v2w2 + v3w3 + v4w4 (1)

Это равенство можно переписать в виде:

V • W = (v1 + w2)(v2 + w1) + (v3 + w4)(v4 + w3) - v1v2 - v3v4 - w1w2 - w3w4 (2)

Вторую часть данного выражения 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 дополнительные два сложения.

# Конструкторская часть

На вход алгоритмы принимают две матрицы. На выходе выдают матрицу — результат произведения двух этих матриц.

# Схемы алгоритмов

На рис. 2-4 приведены схемы разработанных алгоритмов:

1. стандартный алгоритм перемножения матриц;
2. алгоритм Винограда;
3. модифицированный алгоритм Винограда.

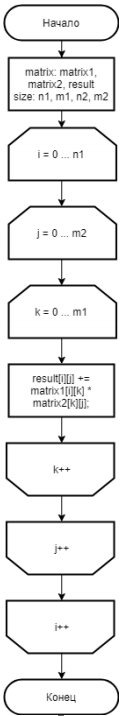


Рисунок 2. *Стандартный алгоритм перемножения матриц*

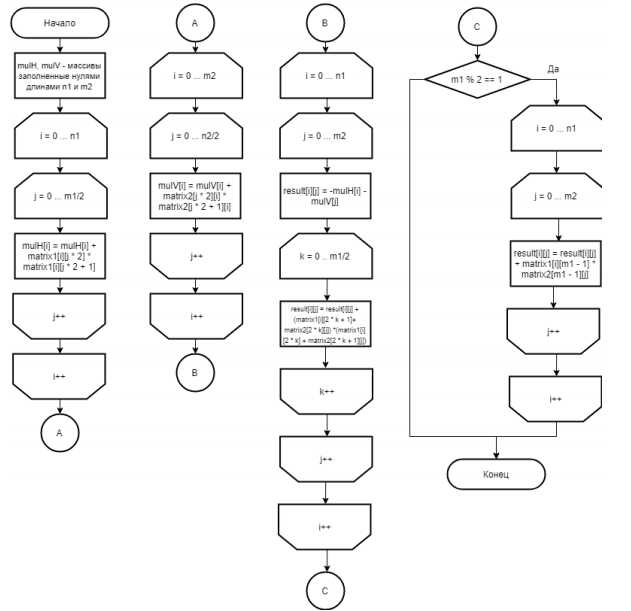


Рисунок 3. *Алгоритм Винограда*

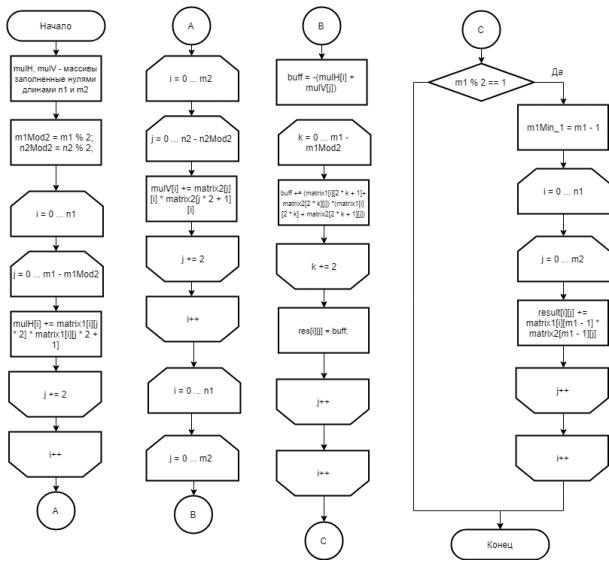
**

Рисунок 4. *Модифицированный алгоритм Винограда*

# Технологическая часть

В качестве языка программирования был выбран python, т.к. данный язык программирования имеет большое количество полезных библиотек для различных необходимостей, а также язык предоставляет средства для быстрого прототипирования и динамической семантики. Для замера процессорного времени была использована функция process\_time(), стандартной библиотеки python – time [2].

## Реализация алгоритмов

В листингах 1-4 представлена реализация алгоритмов умножения матриц. В листинге 5 представлена функция для замера времени выполнения заданной функции на заданном количестве итераций на матрицах указанного размера.

Листинг 1. Реализация стандартного алгоритма умножения матриц

**def** classic**(**n**,** m**,** k**,** matr1**,** matr2**,** res**):**

i **=** 0

**while** i **<** n**:**

j **=** 0

**while** j **<** k**:**

l **=** 0

**while** l **<** m**:**

res**[**i**][**j**]** **+=** matr1**[**i**][**l**]** **\*** matr2**[**l**][**j**]**

l **+=** 1

j **+=** 1

i **+=** 1

**return** res

Листинг 2. Реализация алгоритма Винограда

**def** vinograd**(**n**,** m**,** k**,** matr1**,** matr2**,** res**):**

mulH **=** **[**0**]** **\*** n

mulV **=** **[**0**]** **\*** k

i **=** 0

**while** i **<** n**:**

j **=** 0

**while** j **<** m **//** 2**:**

mulH**[**i**]** **=** mulH**[**i**]** **+** matr1**[**i**][**j **\*** 2**]** **\*** matr1**[**i**][**j **\*** 2 **+** 1**]**

j **+=** 1

i **+=** 1

i **=** 0

**while** i **<** k**:**

j **=** 0

**while** j **<** m **//** 2**:**

mulV**[**i**]** **=** mulV**[**i**]** **+** matr2**[**j **\*** 2**][**i**]** **\*** matr2**[**j **\*** 2 **+** 1**][**i**]**

j **+=** 1

i **+=** 1

i **=** 0

**while** i **<** n**:**

j **=** 0

**while** j **<** k**:**

res**[**i**][**j**]** **=** **-**mulH**[**i**]** **-** mulV**[**j**]**

l **=** 0

**while** l **<** m **//** 2**:**

res**[**i**][**j**]** **=** res**[**i**][**j**]** **+** **(**matr1**[**i**][**2 **\*** l **+** 1**]** **+** matr2**[**2 **\*** l**][**j**])** **\*** \

**(**matr1**[**i**][**2 **\*** l**]** **+** matr2**[**2 **\*** l **+** 1**][**j**])**

l **+=** 1

j **+=** 1

i **+=** 1

**if** m **%** 2 **==** 1**:**

i **=** 0

**while** i **<** n**:**

j **=** 0

**while** j **<** k**:**

res**[**i**][**j**]** **=** res**[**i**][**j**]** **+** matr1**[**i**][**m **-** 1**]** **\*** matr2**[**m **-** 1**][**j**]**

j **+=** 1

i **+=** 1

**return** res

Листинг 3. Реализация модифицированного алгоритма Винограда

**def** vinograd\_opt**(**n**,** m**,** k**,** matr1**,** matr2**,** res**):**

mulH **=** **[**0**]** **\*** n

mulV **=** **[**0**]** **\*** k

tmp **=** m **-** m **%** 2

i **=** 0

**while** i **<** n**:**

j **=** 0

**while** j **<** tmp**:**

mulH**[**i**]** **+=** matr1**[**i**][**j**]** **\*** matr1**[**i**][**j **+** 1**]**

j **+=** 2

i **+=** 1

i **=** 0

**while** i **<** k**:**

j **=** 0

**while** j **<** tmp**:**

mulV**[**i**]** **+=** matr2**[**j**][**i**]** **\*** matr2**[**j **+** 1**][**i**]**

j **+=** 2

i **+=** 1

i **=** 0

**while** i **<** n**:**

j **=** 0

**while** j **<** k**:**

buff **=** **-**mulH**[**i**]** **-** mulV**[**j**]**

l **=** 0

**while** l **<** tmp**:**

buff **+=** **(**matr1**[**i**][**l **+** 1**]** **+** matr2**[**l**][**j**])** **\*** **(**matr1**[**i**][**l**]** **+** matr2**[**l **+** 1**][**j**])**

l **+=** 2

res**[**i**][**j**]** **=** buff

j **+=** 1

i **+=** 1

**if** m **%** 2 **==** 1**:**

i **=** 0

tmp **=** m **-** 1

**while** i **<** n**:**

j **=** 0

**while** j **<** k**:**

res**[**i**][**j**]** **+=** matr1**[**i**][**tmp**]** **\*** matr2**[**tmp**][**j**]**

j **+=** 1

i **+=** 1

**return** res

Листинг 4. Функция подсчета среднего времени выполнения программы для матриц размерами length.

**def** time\_analyze**(**function**,** iterations**=**100**,** length**=**5**):**

size **=** **[]**

**for** \_ **in** **range(**3**):**

size**.**append**(**length**)**

t1 **=** process\_time**()**

**for** \_ **in** **range(**iterations**):**

start**(**size**,** function**,** **False)**

t2 **=** process\_time**()**

**return** **(**t2 **-** t1**)** **/** iterations

## Тестирование функций

Для модульного тестирования реализованных алгоритмах (см. листинги 1-4) была использована стандартная библиотека языка python – unittest. Все функции протестированы на различных входных данных.

Листинг 5. Функция сравнения матриц

**def** comp\_matr**(**matr1**,** matr2**):**

**if** **len(**matr1**)** **!=** **len(**matr2**):**

**return** **False**

**if** **len(**matr1**)** **==** 0 **or** **len(**matr1**[**0**])** **!=** **len(**matr2**[**0**]):**

**return** **False**

**for** i **in** **range(len(**matr1**)):**

**for** j **in** **range(len(**matr1**[**i**])):**

**if** matr1**[**i**][**j**]** **!=** matr2**[**i**][**j**]:**

**return** **False**

**return** **True**

Все тесты были пройдены успешно.

# Экспериментальная часть

При запуске программы первое, что видит пользователь, – это интерфейс.

## Интерфейс

На рис. 5 представлено главное меню программы. В зависимости от выбранного пункта меню запускается соответствующий алгоритм

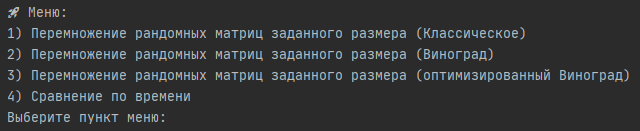


Рисунок 5. Интерфейс – меню

На рис. 6-9 приведены примеры работы программы при вводе строк «*тело*» и «*столб*» при выборе пунктов меню 1-4.

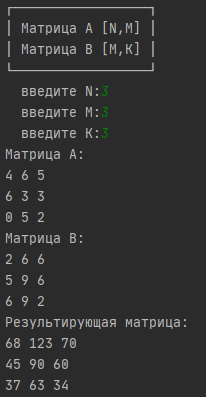


Рисунок 6. Пример работы программы при выборе пункта 1

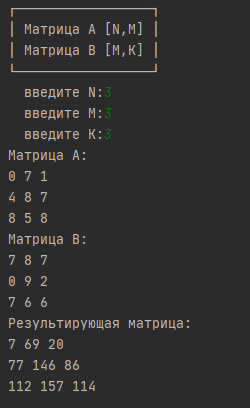


Рисунок 7. Пример работы программы при выборе пункта 2

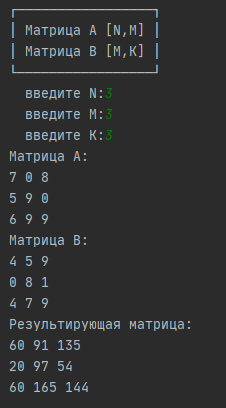


Рисунок 8. Пример работы программы при выборе пункта 3

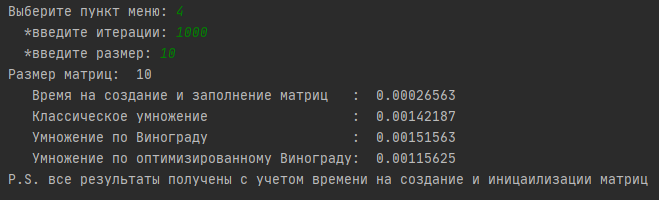


Рисунок 9. Пример работы программы при выборе пункта 4

## Сравнение алгоритмов по времени работы реализаций

Для сравнения в программе необходимо провести замеры процессорного времени для выполнения n – количества вычислений для матриц одинакового размера.

Таблица 3. Сравнение алгоритмов по времени

|  |  |  |
| --- | --- | --- |
| Алгоритм | Размер матриц | Среднее время, сек |
| Стандартный алгоритм | 10 | 0.00136250 |
| Алгоритм Винограда | 10 | 0.00155625 |
| Модифицированный алгоритм Винограда | 10 | 0.00119062 |
| Стандартный алгоритм | 25 | 0.02106250 |
| Алгоритм Винограда | 25 | 0.02340625 |
| Модифицированный алгоритм Винограда | 25 | 0.01775000 |
| Стандартный алгоритм | 50 | 0.16750000 |
| Алгоритм Винограда | 50 | 0.17093750 |
| Модифицированный алгоритм Винограда | 50 | 0.12218750 |

Замеры времени усреднялись для каждого набора экспериментов. Для этого все вычисления производились на случайных матрицах при 100-10000 итераций. Количество итераций обратно пропорционально размеру матриц. Это связано с тем, что время умножения больших матриц велико.

Таблица 4. Сравнение времени на малых размерах

|  |  |  |
| --- | --- | --- |
| Алгоритм | Размер матриц | Ср. время, сек |
| Стандартный алгоритм | 2 | 0.00001297 |
| Модифицированный алгоритм Винограда | 2 | 0.00001859 |
| Стандартный алгоритм | 3 | 0.00003906 |
| Модифицированный алгоритм Винограда | 3 | 0.00004844 |
| Стандартный алгоритм | 4 | 0.00009531 |
| Модифицированный алгоритм Винограда | 4 | 0.00010312 |
| Стандартный алгоритм | 5 | 0.00017656 |
| Модифицированный алгоритм Винограда | 5 | 0.00018906 |
| Стандартный алгоритм | 6 | 0.00030469 |
| Модифицированный алгоритм Винограда | 6 | 0.00030937 |
| Стандартный алгоритм | 7 | 0.00047813 |
| Модифицированный алгоритм Винограда | 7 | 0.00046406 |

**Вывод**: модифицированный алгоритм Винограда показывает наилучшее время на средних размерах, хотя обычный алгоритм Винограда показывает наихудшее время. Это связано с тем, что обычный алгоритм Винограда мало оптимизирован и приходится вычислять одни и те же значения несколько раз. Стандартный алгоритм работает быстрее всех на маленьких размерах. Выбирать алгоритм необходимо в зависимости от поставленной задачи. Если вам требуется быстро перемножать огромные матрицы, то модифицированный алгоритм Винограда справится с этой задачей лучше всех. Но если вам необходимо работать с матрица малых размеров (< 7), то стандартный алгоритм подойдет больше.

* 1. **Оценка трудоемкости алгоритмов**

Трудоемкость алгоритма — это количество операций, которые необходимо выполнить в процессе решения поставленной задачи с помощью этого алгоритма.

* + 1. **Стандартный алгоритм**

Рассмотри трудоемкость стандартного алгоритма умножения матриц

Подсчет: 2 + n \* (2 + 2 + k \* (2 + 2 + m \* (2 + 6 + 2)))

Итог: 10\*n\*k\*m + 4 \* k \* n + 4 \* n + 2, где n, k, m – константы для матриц **A** с размерами n \* m и **B** с размерами m \* k

* + 1. **Алгоритм Винограда**

Рассмотри трудоемкость алгоритма Винограда для умножения матриц

Таблица 1. Трудоемкость алгоритма Винограда

|  |  |
| --- | --- |
| Части алгоритма | Трудоемкость |
| Инициализация mulH и mulV | 2 \* 3 |
| Заполнение mulH | 2 + n \* (2 + 2 + m / 2 \* (3 + 6 + 6)) |
| Заполнение mulV | 2 + k \* (2 + 2 + m / 2 \* (3 + 6 + 6)) |
| Подсчет результата | 2 + n \* (2 + 2 + k \* (2 + 7 + 2 + m / 2 \* (3 + 23))) |
| Условие нечетности m | 2 |
| Для матриц с нечетным m | 2 + n \*(2 + 2 + k \* (2 + 8 + 5)) |

Итог: 13 \* m \* n \* k + 7.5 \* m \* n + 7.5 \* k \* m + 11 \* k \* n + 8 \* n + 4 \* k + 14 +

+, где n, k, m – константы для матриц **A** с размерами n \* m и **B** с размерами m \* k. Выражение в [] принимает значение сверху, если m четное, а нижнее — иначе.

* + 1. **Модифицированный алгоритм Винограда**

Рассмотри трудоемкость модифицированного алгоритма Винограда для умножения матриц

Таблица 2. Трудоемкость модифицированного алгоритма Винограда

|  |  |
| --- | --- |
| Части алгоритма | Трудоемкость |
| Инициализация mulH и mulV | 2 \* 3 |
| Переменная tmp | 3 |
| Заполнение mulH | 2 + n \* (2 + 2 + m / 2 \* (2 + 5 + 3)) |
| Заполнение mulV | 2 + k \* (2 + 2 + m / 2 \* (2 + 5 + 3)) |
| Подсчет результата | 2 + n \* (2 + 2 + k \* (2 + 5 + 3 + 2 + m / 2 \* (2 + 14))) |
| Условие нечетности m | 2 |
| Для матриц с нечетным m | 2 + 2 + n \* (2 + 2 + k \* (2 + 6 + 2)) |

Итог: 8 \* m \* n \* k + 5 \* m \* n + 5 \* k \* m + 12 \* k \* n + 8 \* n + 4 \* k + 17 +

+, где n, k, m – константы для матриц **A** с размерами n \* m и **B** с размерами m \* k. Выражение в [] принимает значение сверху, если m четное, а нижнее — иначе.

* 1. **Вывод**

Можно представить графики зависимости трудоемкости алгоритмов для умножения квадратных матриц с параметром x. На рис. 10 – 11 представлены графики таких функций.

1. Синий график – алгоритм Винограда
2. Зеленый график – модифицированный алгоритм Винограда
3. Красный график – стандартный алгоритм

****

Рисунок 10. *Графики трех алгоритмов по умножению матриц наилучший случай*

1. Фиолетовый график – алгоритм Винограда
2. Черный график – модифицированный алгоритм Винограда
3. Красный график – стандартный алгоритм

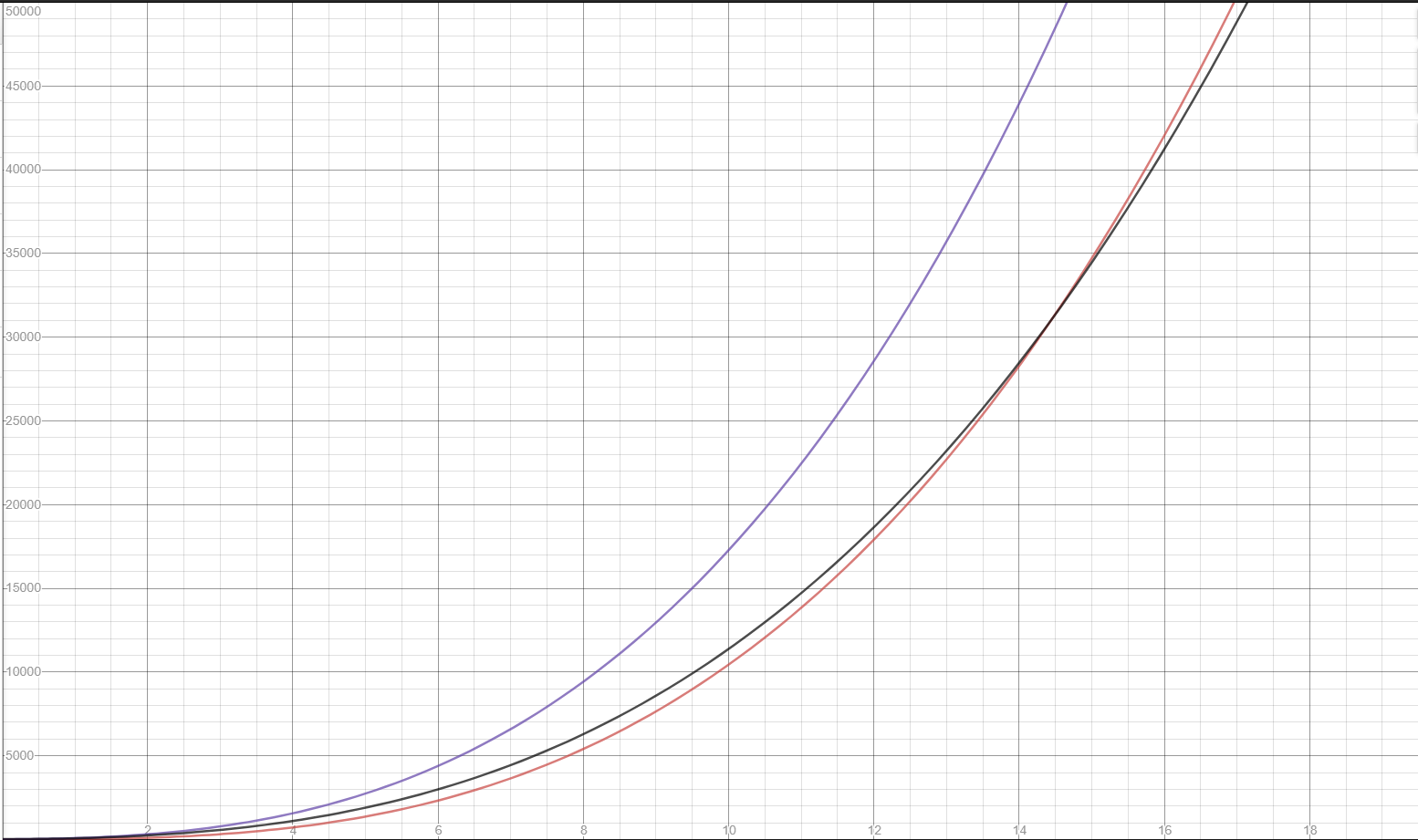
****

Рисунок 11. *Графики трех алгоритмов по умножению матриц наихудший случай*

**Вывод:** исходя из графиков, можно сделать вывод, что до определенного момента стандартный алгоритм умножения матриц наиболее быстрый, а также этот алгоритм стабилен, т.е. не зависит от четности m. Не модифицированный алгоритм Винограда показывает наихудшее время вне зависимости от четности. Модифицированный алгоритм показывает лучшее время с некоторого момента.

Заключение

В ходе работы были изучены алгоритмы умножения матриц. Реализованы 3 алгоритма, приведен программный код реализации алгоритмов по умножению матриц.

Была подсчитана трудоемкость каждого из алгоритмов. А также было проведено сравнение алгоритмов по времени и трудоемкости.

Цель работы достигнута. Получены практические навыки реализации алгоритмов Винограда и стандартного алгоритма, а также проведена исследовательская работа по оптимизации и вычислении трудоемкости алгоритмов.

Список литературы

1. Дж. Макконнелл. Анализ алгоритмов. Активный обучающий подход. –М.: Техносфера, 2017. – 267 c.

2. Официальный сайт Python, документация [Электронный ресурс]. Режим доступа: <https://docs.python.org/3/library/time.html>, свободный (дата обращения: 16.09.20).