|  |  |
| --- | --- |
| 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 Программное обеспечение ЭВМ и информационные технологии

**Лабораторная работа №2**

**по курсу “Анализ алгоритмов”**

**по теме “Алгоритм Копперсмита – Винограда”**

|  |  |
| --- | --- |
| Студент: | Уласик Е.А. |
| Группа: | ИУ7-51 |
| Преподаватель: | Волкова Л.Л. |

*2019 г.*

**Оглавление**

[Введение 3](#_Toc24027025)

[Задачи работы 4](#_Toc24027026)

[1. Аналитическая часть 5](#_Toc24027027)

[1.1 Классический алгоритм 5](#_Toc24027028)

[1.2 Алгоритм умножения матриц Винограда 6](#_Toc24027029)

[2. Конструкторская часть 7](#_Toc24027030)

[2.1 Разработка реализаций алгоритмов 7](#_Toc24027031)

[2.2 Расчёт трудоёмкости 14](#_Toc24027032)

[3. Технологическая часть 15](#_Toc24027033)

[3.1 Требования к программному обеспечению 15](#_Toc24027034)

[3.2 Средства реализации 15](#_Toc24027035)

[3.3 Листинг кода 15](#_Toc24027036)

[4. Экспериментальная часть 18](#_Toc24027037)

[Примеры работы 18](#_Toc24027038)

[4.1 Результаты тестирования 19](#_Toc24027039)

[4.2 Постановка эксперимента по замеру времени 20](#_Toc24027040)

[4.3 Сравнительный анализ на материале экспериментальных данных 20](#_Toc24027041)

[Заключение 22](#_Toc24027042)

[Список литературы 23](#_Toc24027043)

# **Введение**

Алгоритм Копперсмита—Винограда — алгоритм умножения квадратных матриц, предложенный̆ в 1987 году Д. Копперсмитом и Ш. Виноградом. В исходной̆ версии асимптотическая сложность алгоритма составляла O(n2,3755), где n — размер стороны матрицы. Алгоритм Копперсмита—Винограда, с учетом серии улучшений и доработок в последующие годы, обладает лучшей асимптотикой среди известных алгоритмов умножения матриц [1].

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].

# **Задачи работы**

Задачами данной лабораторной работы являются:

1. Реализовать алгоритмы умножения матриц:
2. Классический алгоритм умножения;
3. Алгоритм Копперсмита-Винограда;
4. Улучшенный Алгоритм Копперсмита-Винограда.
5. Проанализировать трудоёмкость данных алгоритмов.
6. Провести экспериментальный анализ по замерам времени.

# **Аналитическая часть**

В данном разделе будут даны описания алгоритмов.

## 1.1 Классический алгоритм

Пусть даны две прямоугольные матрицы A и B размерности n m, m k соответственно:

,

Тогда матрица C размерностью n k:

,

в которой:

называется их произведением.

Операция умножения двух матриц выполнима только в том случае, если число столбцов в первом сомножителе равно числу строк во втором; в этом случае говорят, что матрицы “согласованы”. В частности, умножение всегда выполнимо, если размерности матриц совпадают.

## 1.2 Алгоритм умножения матриц Винограда

Если посмотреть на результат умножения двух матриц, то видно, что каждый элемент в нём представляет собой скалярное произведение соответствующих строки и столбца исходных матриц. Можно заметить также, что такое умножение допускает предварительную обработку, позволяющую часть работы выполнять заранее.

Рассмотрим два вектора: и .

Их скалярное произведение равно: .

Это равенство можно переписать в виде:

Не очевидным остаётся тот факт, что выражение в правой части формулы 2 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 5 сложений, а также дополнительно два сложения.

# **Конструкторская часть**

В данном разделе размещены схемы алгоритмов.

## 2.1 Разработка реализаций алгоритмов

Ниже на приведены схемы алгоритмов решения поставленных задач

Классический алгоритм умножения матриц



Алгоритм умножения Винограда







Оптимизированный алгоритм Винограда







## 2.2 Расчёт трудоёмкости

Пусть заданы 2 матрица размерностями m\*n u n \* q. Используя выше представленную модель вычислений, произведем расчет трудоемкости алгоритмов умножения матриц.

* Стандартный алгоритм
* Алгоритм Винограда
* Оптимизированный Алгоритм Винограда

Оптимизированный алгоритм Винограда схематично представляет собой обычный алгоритм Винограда, за исключением следующих оптимизаций:

1. Вычисление заранее N – 1 и flag = N % 2
2. Вычисление
3. Замена C[i][j] = C[i][j]+… на C[i][j] +=
4. последний цикл для нечетных элементов включен в основной цикл, используя дополнительные операции в случае нечетности

# **Технологическая часть**

В данном разделе приведены требования к программному обеспечению, средства реализации, листинг кода и примеры тестирования.

## 3.1 Требования к программному обеспечению

Требуется ввести две целочисленные матрицы и вернуть целочисленную матрицу – результат умножения или сообщение о невозможности их умножения.

## 3.2 Средства реализации

В качестве языка программирования я выбрал Python в связи с его широким функционалом, простотой и быстротой работы. Средства разработки PyCharm. Время работы процессора замеряется с помощью модуля time.

## 3.3 Листинг кода

1. Реализация классического алгоритма умножения матриц

def matrix\_multiplication(matrix\_a: list, matrix\_b: list) -> list:  
 if len(matrix\_b) != len(matrix\_a[0]):  
 print("Different sizes!")  
 return []  
  
 res = [[0 for \_ in range(len(matrix\_b[0]))] for \_ in range(len(matrix\_a))]  
  
 for i in range(len(matrix\_a)):  
 for j in range(len(matrix\_a[0])):  
 for k in range(len(matrix\_b[0])):  
 res[i][k] = res[i][k] + matrix\_a[i][j] \* matrix\_b[j][k]  
 return res

1. Реализация алгоритма Винограда

def matrix\_multiplication\_winograd(matrix\_a: list, matrix\_b: list) -> list:  
 b = len(matrix\_b)  
 if b != len(matrix\_a[0]):  
 print("Different sizes!")  
 return []  
   
 a = len(matrix\_a)  
 c = len(matrix\_b[0])  
   
 d = b // 2  
 row\_factor = [0 for \_ in range(a)]  
 col\_factor = [0 for \_ in range(c)]  
  
 # Row Factor calc  
 for i in range(a):  
 for j in range(d):  
 row\_factor[i] += matrix\_a[i][2 \* j] \* matrix\_a[i][2 \* j + 1]  
  
 # Col Factor calc  
 for i in range(c):  
 for j in range(d):  
 col\_factor[i] += matrix\_b[2 \* j][i] \* matrix\_b[2 \* j + 1][i]  
  
 res = [[0 for \_ in range(c)] for \_ in range(a)]  
 for i in range(a):  
 for j in range(c):  
 res[i][j] = - row\_factor[i] - col\_factor[j]  
 for k in range(d):  
 res[i][j] += ((matrix\_a[i][2 \* k] + matrix\_b[2 \* k + 1][j]) \*  
 (matrix\_a[i][2 \* k + 1] + matrix\_b[2 \* k][j]))  
  
 if b % 2:  
 for i in range(a):  
 for j in range(c):  
 res[i][j] += matrix\_a[i][b - 1] \* matrix\_b[b - 1][j]  
  
 return res

1. Реализация модифицированного алгоритма Винограда

def matrix\_multiplication\_improved\_winograd(matrix\_a: list, matrix\_b: list) -> list:  
 b = len(matrix\_b)  
   
 if b != len(matrix\_a[0]):  
 print("Different dimension of the matrics")  
 return []  
   
 a = len(matrix\_a)  
 c = len(matrix\_b[0])  
  
 d = b // 2  
  
 row\_factor = [0 for \_ in range(a)]  
 col\_factor = [0 for \_ in range(c)]  
  
 # Row Factor calc  
 for i in range(a):  
 row\_factor[i] = sum(matrix\_a[i][2 \* j] \* matrix\_a[i][2 \* j + 1] for j in range(d))  
  
 # Col Factor calc  
 for i in range(c):  
 col\_factor[i] = sum(matrix\_b[2 \* j][i] \* matrix\_b[2 \* j + 1][i] for j in range(d))  
  
 res = [[0 for \_ in range(c)] for j in range(a)]  
 for i in range(a):  
 for j in range(c):  
 res[i][j] = sum((matrix\_a[i][2 \* k] + matrix\_b[2 \* k + 1][j]) \*  
 (matrix\_a[i][2 \* k + 1] + matrix\_b[2 \* k][j]) for k in range(d)) \  
 - row\_factor[i] - col\_factor[j]  
  
 if b % 2:  
 for i in range(a):  
 for j in range(c):  
 res[i][j] = sum(matrix\_a[i][b - 1] \* matrix\_b[b - 1][j])  
  
 return res

# **Экспериментальная часть**

В данном разделе будут показаны результаты тестов и замеры времени.

**Примеры работы**

**Пример 1:**  
Матрица А:  
1 2 3  
4 5 6  
Матрица B:  
0  
0  
0

Результат:  
0  
0  
0

**Пример 2:**  
Матрица А:  
1 2   
4 5   
Матрица B:  
 0 3   
-6 1

Результат:  
-12 5  
-30 17

**Пример 3:**

Матрица A:

10 -6

Матрица B:

654

753

Результат:

18 20 22

## 4.1 Результаты тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | A | B | Ожидаемый результат | Полученный результат |
| 1 | Нулевая матрица | Нулевая матрица | Нулевая матрица | Нулевая матрица |
| 2 | Нулевая | случайная | нулевая | нулевая |
| 3 | Случайная | Нулевая | Нулевая | Нулевая |
| 4 | Единичная | Квадратная | B | B |
| 5 | Квадратная | Единичная | A | A |
| 6 | Размера M x N | Размера Q x N | Ошибка размерностей | Ошибка  размерностей |
| 7 | Размера M x N | Размера N х M | Размера M х M | Размера M х M |

В таблице 1 в столбцах результатов указаны результаты тестирования. 1 столбец – номер тестового случая, 2 и 3 столбцы – исходные матрицы, 4 – ожидаемый результат, 5 – полученный результат. Если в исходных матрицах не указаны размерности – значит, что они одинакового порядка.

Программа успешно прошла все тестовые случаи, все полученные результаты совпали с ожидаемыми.

## 4.2 Постановка эксперимента по замеру времени

Для произведения замеров времени выполнения реализаций алгоритмов будет использована следующая формула , где t – время выполнения, N – количество замеров. Неоднократное измерение времени необходимо для построения более гладкого графика.

Количество замеров будет взято равным 50.

Тестирование будет проведено на одинаковых входных данных. 1) Матрицы размерностями от 100х100 до 800х800 с шагом 100 для матриц четных размеров и от 101х101 до 801х801 с шагом 100 для матриц с нечетных размеров.

## 4.3 Сравнительный анализ на материале экспериментальных данных

Ниже приведены графики зависимости временных затрат (в тиках процессора) от размеров входных данных.

При тестировании на чётных матрицах:

На графике видно, что оптимизированный алгоритм Винограда превосходит стандартный алгоритм на 30 % и неоптимизированный алгоритм почти на 45%

При тестировании на нечётных матрицах:

На графике видно, что оптимизированный алгоритм не потерял свое превосходство, стандартный алгоритм не изменил своего времени работы, а алгоритм Винограда стал работать медленнее .

# **Заключение**

В ходе работы были изучены и реализованы алгоритмы стандартного умножения матриц, алгоритма Винограда и оптимизированного алгоритма Винограда. Был проведен сравнительный анализ перечисленных алгоритмов по трудоемкости и экспериментально выявлена временная разница работы алгоритмов. Классический алгоритм в неоптимизированном виде является более эффективным чем алгоритм винограда, однако после ряда оптимизаций, алгоритм Винограда становится значительно быстрее классического.

# **Список литературы**

[1]  Henry Cohn, Robert Kleinberg, Balazs Szegedy, and Chris Umans. Group- theoretic Algorithms for Matrix Multiplication. arXiv:math.GR/0511460. Proceedings of the 46th Annual Symposium on Foundations of Computer Science, 23-25 October 2005, Pittsburgh, PA, IEEE Computer Society, pp. 379—388..

[2]  Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. Journal of Symbolic Computation, 9:251-280, 1990.