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 №3 по курсу "Анализ алгоритмов"

Тема Алгоритмы сортировки

Студент Якуба Д. В.

Группа ИУ7-53Б

Оценка (баллы) _____

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

Москва — 2020 г.

Оглавление

Введение

Цели лабораторной работы

1. изучение алгоритмов сортировки пузырьком, вставками и выбором;
2. реализация алгоритмов сортировки пузырьком, вставками и выбором;
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;
6. получение практических навыков реализации алгоритмов на ЯП Kotlin.

Определение

Сортировка - это процесс перегруппироваки заданной последовательности объектов в некотором определённом порядке. Такой определённый порядок позволяет, в некоторых случаях, эффективнее работать с заданной последовательностью.

Пусть ребуется упорядочить N элементов: $E_1, E_2, ..., E_n$. Каждый элемент представляет из себя запись E_j, содержащую некоторую информацию и ключ K_j, управляющий процессом сортировки. На множестве ключей определено отношение порядка $<$ так, чтобы для любых трёх значений ключей a, b, c выполнялись следующие условия:

- Либо $a < b$, либо $b < c$, либо $a = b$;
- Если $a < b$ и $b < c$, то $a < c$.

Данные условия определяют математическое понятие линейного и совершенного упорядочения, а удовлетворяющие им множества поддаются сортировке болшинством методов.

Задачуй сортировки является нахождение такой перестановки занисей $p(1)p(2)...p(n)$ с индексами $1, 2, ..., N$, после которой ключи расположились бы в порядке неубывания.

$$K_{p(1)} \leq K_{p(2)} \leq ... \leq K_{p(n)} \tag{1}$$

1 Аналитическая часть

1.1 Классический алгоритм умножения матриц

1.2 Алгоритм Копперсмита-Винограда умножения матриц

Вывод

2 Конструкторская часть

2.1 Блок-схема классического алгоритма умножения матриц

2.2 Блок-схема алгоритма Копперсмита-Винограда

2.3 Блок-схема улучшенного алгоритма Копперсмита-Винограда

2.4 Модель вычислений

2.5 Трудоёмкость алгоритмов

2.5.1 Алгоритм сортировки пузырьком

2.5.2 Алгоритм сортировки вставками

2.5.3 Улучшенный сортировки выбором

Вывод

На основе теоретичесих данных, полученных из аналитического раздела, были построены схемы рассматриваемых алгоритмов сортировок, оценены их трудоёмкости в лучшем и худшем случаях.

3 | Технологическая часть

3.1 Требования к программному обеспечению

3.2 Средства реализации программного обеспечения

При написании программного продукта был использован язык программирования Kotlin [5].

Данный выбор обусловлен следующими факторами:

- Высокая вычислительная производительность;
- Большое количество справочной литературы, связанной с ЯП Java.

Для тестирования производительности реализаций алгоритмов использовалась утилита measureTimedValue.

При написаннии 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 3.1 - 3.3 предоставлены реализации рассматриваемых алгоритмов.

Листинг 3.1: Функция реализации алгоритма классического умножения матриц

Листинг 3.2: Функция реализации алгоритма Копперсмита-Винограда

Листинг 3.3: Функция реализации улучшенного алгоритма Копперсмита-Винограда

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
$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 3 & 3 \\ 1 & 3 & 3 \\ 1 & 2 & 2 \end{pmatrix}$	$\begin{pmatrix} 6 & 15 & 15 \\ 6 & 15 & 15 \\ 3 & 8 & 8 \end{pmatrix}$
$\begin{pmatrix} 1 & 2 & 4 \\ 1 & 2 & 4 \end{pmatrix}$	$\begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix}$	$\begin{pmatrix} 32 \\ 32 \end{pmatrix}$
$\begin{pmatrix} 5 \end{pmatrix}$	$\begin{pmatrix} 666 \end{pmatrix}$	$\begin{pmatrix} 3330 \end{pmatrix}$
$\begin{pmatrix} -1 & -2 & 3 \\ 1 & 2 & 3 \\ -1 & -2 & 3 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{pmatrix}$	$\begin{pmatrix} 4 & 4 & 4 \\ 14 & 14 & 14 \\ 4 & 4 & 4 \end{pmatrix}$
$\begin{pmatrix} 666 & 666 \end{pmatrix}$	$\begin{pmatrix} 777 & 777 \end{pmatrix}$	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

Ниже на рисунках 4.1- 4.2 предоставлены примеры работы каждого из алгоритмов на случайных данных, сгенерированных один раз, и введённых пользователем данных.

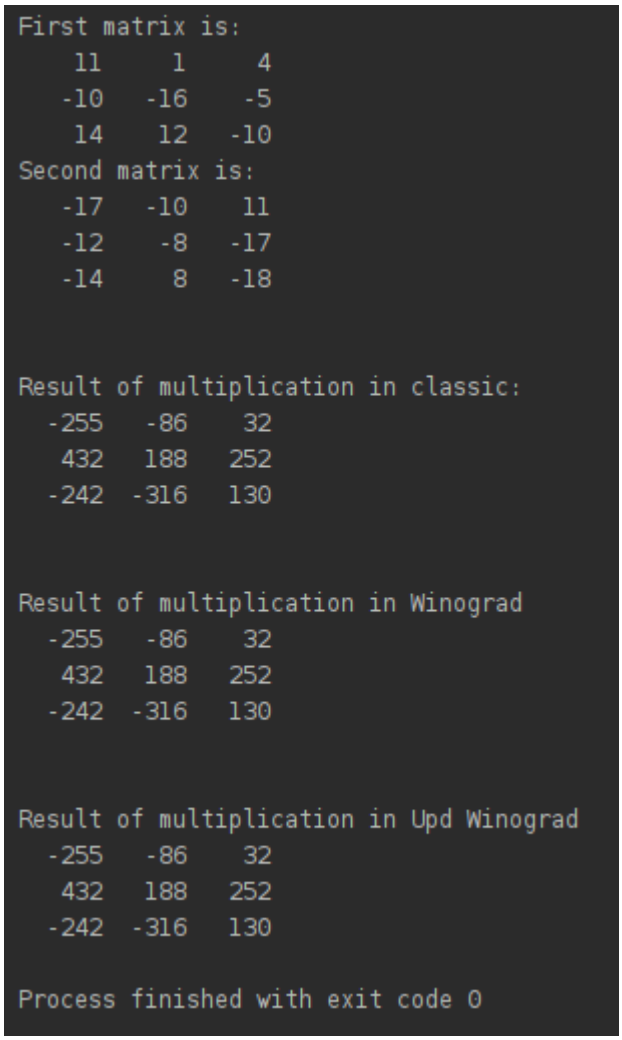

Рис. 4.1: Пример работы ПО.

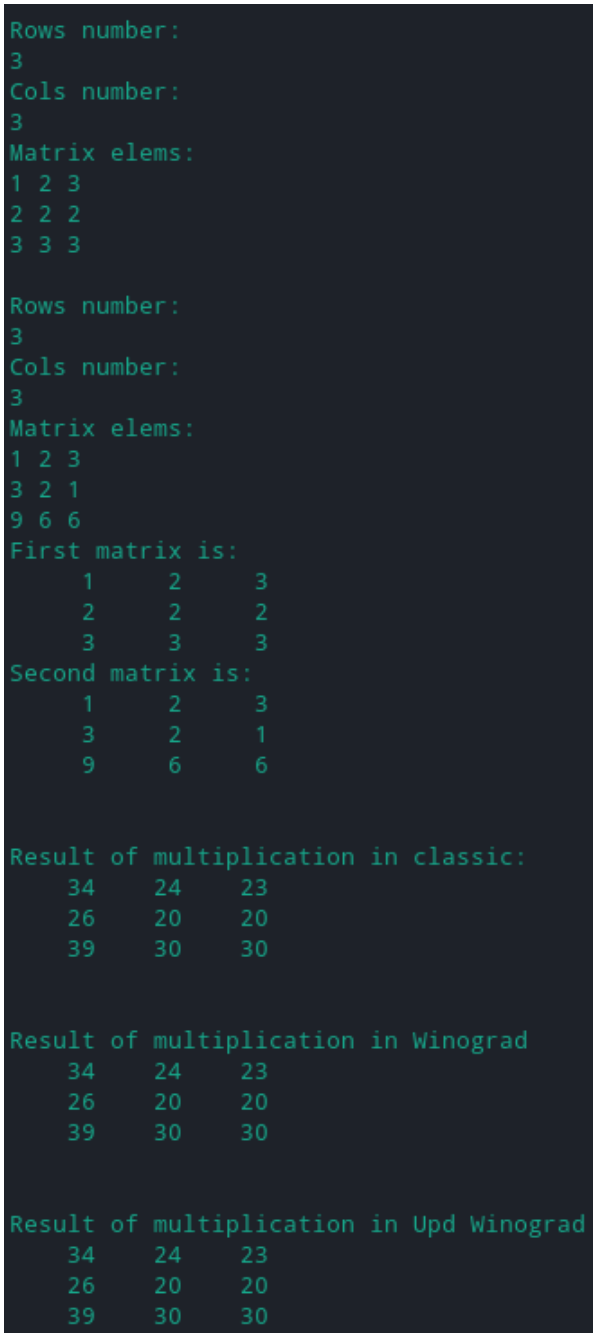

Рис. 4.2: Пример работы ПО.

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 4.3 и 4.4 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

При сравнении результатов замеров времени заметно, что скорость работы классического алгоритма однозначно отстаёт от скорости работы улучшенного Алгоритма Копперсмита-Винограда. Уже на 600 элементах улучшенный алгоритм Копперсмита-Винограда работает быстрее классического на $\approx 8\%$. При нечётном количестве строк и столбцов матриц улучшенный алгоритм способен быть медленнее $\approx 4\%$, при факте того, что классический алгоритм похожей динамики не имеет. Обычный алгоритм Копперсмита-Винограда начинает выигрывать по скорости классический только по достижению 300 строк и столбцов в матрице, при факте того, что в случае матрицы с нечётной размерностью он всё ещё будет проигрывать. При размерности 600 он будет выгрывать у классической реализации на $\approx 3\%$. В случае матриц размера меньше 400 на 400 его использование не будет целесообразным.

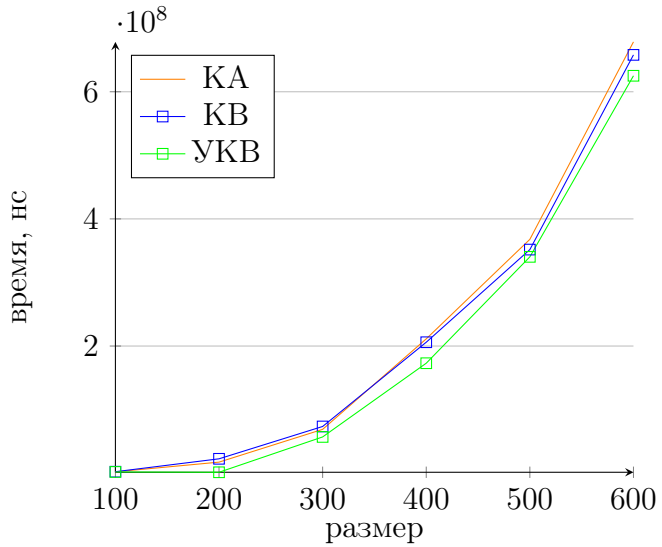

Рис. 4.3: Зависимость времени работы от размера матриц (чётные значения размерностей)

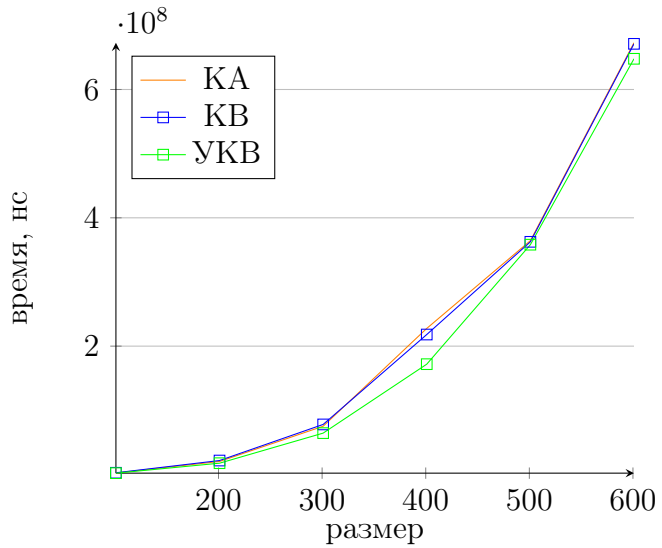

Рис. 4.4: Зависимость времени работы от размера матриц (нечётные значения размерностей)

Заключение

В ходе выполнения лабораторной работы:

- были изучены алгоритмы умножения матриц: классический, Копперсмита-Винограда и улучшенный Копперсмита-Винограда;
- были реализованы алгоритмы умножения матриц: классический, Копперсмита-Винограда и улучшенный Копперсмита-Винограда;
- был произведён анализ трудоёмкости указанных алгоритмов на основе теоретических расчётов и выбранной модели вычислений;
- был выполнен сравнительный анализ производительности алгоритмов на основе полученных экспериментальных данных;
- был подготовлен отчёт по проделанной работе;
- были получены практические навыки реализации алгоритмов на ЯП Kotlin.

Исследования показали, что использование алгоритма Копперсмита-Винограда способно оправдать себя только в случае матриц, размерность которых не менее 400. При этом выигрыш будет составлять $\approx 0.2\%$ только в случае чётной размерности. Реализация улучшенного алгоритма Копперсмита-Винограда показывает результаты быстрее классического алгоритма уже при размерности матрицы 100. Чем больше элементов в матрице, тем заметнее разница во времени работы этих двух алгоритмов. При размерности матрицы 600 модифицированный алгоритм Копперсмита-Винограда показывает себя лучше классического алгоритма на $\approx 8\%$.

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] Strassen V. Gaussian Elimination is not Optimal // Numerische Mathematik. 2005. Vol. 13, no. 9. P. 354–356.

[4] Погорелов Дмитрий Александрович Таразанов Артемий Михайлович Волкова Лилия Леонидовна. Оптимизация классического алгоритма Винограда для перемножения матриц // Журнал №1. 2019. Т. 49.

[5] Kotlin language specification [Электронный ресурс]. Режим доступа: `https://kotlinlang.org/spec/introduction.html` (дата обращения 09.10.2020.