Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский химико-технологический университет имени Д.И. Менделеева»

Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

Выполнил студент группы КС-36 Акулинин Андрей Игоревич

Ссылка на репозиторий: <https://github.com/MUCTR-IKT-CPP/AkulininAI_36/tree/main/algorithms>

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Лобанов Алексей Владимирович

Крашенинников Роман Сергеевич

Дата сдачи: 17.02.2025

Оглавление

[Описание задачи. 2](#_Toc63548272)

[Описание метода/модели. 2](#_Toc63548273)

[Выполнение задачи. 2](#_Toc63548274)

[Заключение. 2](#_Toc63548275)

# Описание задачи.

В рамках лабораторной работы №1 поставлена задача изучения и анализа временной сложности алгоритмов сортировки. Основная цель работы заключается в исследовании наихудшего, наилучшего и среднего времени выполнения выбранного алгоритма сортировки, а также в сопоставлении полученных результатов с теоретической оценкой сложности алгоритма, выраженной в нотации O-большое.

# Описание метода/модели.

Сортировка пузырьком (Bubble Sort) — это простой алгоритм сортировки, который последовательно сравнивает соседние элементы массива и меняет их местами, если они находятся в неправильном порядке. Этот процесс повторяется до тех пор, пока массив не будет полностью отсортирован. Название алгоритма происходит от аналогии с "всплывающими" элементами, как пузырьки в жидкости, где более легкие (меньшие) элементы постепенно перемещаются в начало массива.

**Основные шаги алгоритма:**

1. **Итерация по массиву**:
   * Алгоритм проходит по массиву от начала до конца, сравнивая каждую пару соседних элементов.
   * Если текущий элемент больше следующего, они меняются местами.
   * После первого прохода наибольший элемент "всплывает" в конец массива.
2. **Повторение проходов**:
   * Процесс повторяется для оставшейся части массива (исключая уже отсортированные элементы в конце).
   * На каждом следующем проходе количество сравниваемых элементов уменьшается на один.
3. **Завершение сортировки**:
   * Алгоритм завершает работу, когда за весь проход не происходит ни одного обмена (массив отсортирован).

**Временная сложность:**

* **Наихудший случай**: O(n2). Возникает, когда массив отсортирован в обратном порядке. В этом случае требуется максимальное количество сравнений и обменов.
* **Средний случай**: Ɵ(n2). В среднем, для случайного массива, количество операций также квадратично зависит от размера массива.
* **Наилучший случай**: Ω(n). Если массив уже отсортирован, алгоритм завершится после первого прохода, сделав n−1 сравнений и ни одного обмена.

# Выполнение задачи.

Для выполнения поставленной задачи был использован язык программирования C++. Программа предназначена для анализа времени работы алгоритма сортировки пузырьком (bubble sort) при различных размерах входных массивов. Входные данные генерируются случайным образом, а затем производится их сортировка с подсчетом количества обменов элементов и числа повторений прохода по массиву до завершения сортировки.

**Организация программы:**

Программа состоит из нескольких ключевых функций:

1. generateRandomArray – генерация массива заданного размера со случайными значениями в диапазоне от -1.0 до 1.0.
2. bubble\_sort – реализация алгоритма пузырьковой сортировки с подсчётом количества обменов и числа проходов по массиву.
3. saveDataToFile – сохранение результатов тестирования (время выполнения, количество обменов, число проходов) в файлы формата CSV для последующего построения графиков.
4. main – главная функция, которая управляет процессом тестирования. Она последовательно увеличивает размер массива и запускает серию тестов для каждого размера, собирая статистику о времени выполнения, количестве обменов и числе проходов.

**Проведённые тесты:**

Тестирование проводилось для размеров массива от 1000 до 128000 элементов с шагом удвоения. Для каждого размера массива выполнялось 20 попыток сортировки, чтобы получить усреднённое значение времени выполнения и других показателей.

**Результаты:**

В результате были собраны следующие данные:

* Время выполнения алгоритма для каждого размера массива.
* Количество обменов элементов при каждой попытке сортировки.
* Число проходов по массиву до завершения сортировки.

Эти данные сохранены в файлах times.csv, exchanges.csv и repeats.csv. Построение графиков на основе этих данных позволяет наглядно оценить зависимость времени выполнения алгоритма от размера массива, а также понять динамику количества обменов и числа проходов.

Изображение выглядит как текст, снимок экрана, линия, График

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, График, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, линия, График

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как текст, снимок экрана, линия, График

Контент, сгенерированный ИИ, может содержать ошибки.

# Заключение.

Алгоритм пузырьковой сортировки прост в реализации, но имеет высокую временную сложность O(n2). Тестирование подтвердило экспоненциальный рост времени выполнения с увеличением размера массива. Генерация случайных чисел обеспечила разнообразие наборов данных для тестирования. Сбор статистики и построение графиков позволили проанализировать зависимость времени выполнения от размера массива. Организация программы была проста благодаря стандартным библиотекам C++. Опыт реализации подчеркнул важность тщательного планирования экспериментов и использования подходящих инструментов для анализа данных.