ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 5

«Название лабораторной работы (Вашего задания)»

Выполнил работу

Лорс Алина

Академическая группа № J3112

Ходненко Иван Владимирович

Санкт-Петербург

2024

1. **Введение**

*Цель работы*

Реализация и исследование трех алгоритмов сортировки: Shaker Sort, Pancake Sort и Stooge Sort.

*Задачи:*

-Разработать алгоритмы на языке C++.

-Замерить время выполнения алгоритмов на массивах разного размера.

-Построить графики для наглядного анализа производительности.

-Сравнить результаты с теоретической сложностью алгоритмов.

**2. Теоретическая подготовка**

*Shaker Sort (Шейкерная сортировка):*

Улучшенная пузырьковая сортировка. Проходит массив слева направо и обратно, перемещая элементы в правильные позиции. Применяется для почти отсортированных массивов.

*Блинная сортировка*

Алгоритм сортировки. Единственная операция, допустимая в алгоритме - переворот элементов последовательности до какого-либо индекса. В отличие от традиционных алгоритмов, в которых минимизируют количество сравнений, в блинной сортировке требуется сделать как можно меньше переворотов. Процесс можно визуально представить как стопку блинов, которую тасуют путём взятия нескольких блинов сверху и их переворачивания(Wikipedia)

*Stooge Sort:*

Рекурсивный алгоритм, который делит массив на три части и сортирует их. Используется редко из-за высокой временной сложности.

*Типы данных:*

vector<int>: используется для хранения массива чисел. Это динамический массив из стандартной библиотеки C++, который автоматически управляет памятью и поддерживает удобные операции вставки и удаления элементов.

int: базовый тип данных для целых чисел, используется для хранения значений элементов массива и индексов.

bool: флаг, указывающий, произошли ли перестановки элементов.

clock(): функция из библиотеки <ctime> для измерения времени выполнения алгоритмо

*Shaker Sort*

Преимущество: эффективен на почти отсортированных данных.

Сложность:Лучшая: O(N).

Средняя и худшая: O(N²)

Pancake Sort:

Применяет два переворота для каждой итерации: первый раз — чтобы поместить максимум в начало, второй — чтобы перенести его в конец. Сложность: O(N²) во всех случаях. Используется для демонстрации нестандартных подходов к сортировке.

Stooge Sort: Сложность: O(N^(2.7)) в худшем случае.

*Замеры времени:*

Время выполнения измеряется с помощью функции clock(), которая фиксирует время начала и окончания выполнения алгоритма. Разница позволяет получить время выполнения в миллисекундах.

1. **Реализация**

* Генерация тестовых данных:

Файл test\_data.txt содержит 10,000 случайных чисел в диапазоне от -1000 до 1000.

* Реализация алгоритмов сортировки
* Тестирование и замер времени:
* Запуск сортировок на массивах разного размера: от 1000 до 8000 элементов.
* Результаты сохранены в файле sort\_results.txt.
* Инструменты реализации: Генерация случайных чисел: random\_device и uniform\_int\_distribution. Замер времени: clock() из библиотеки <ctime>.

Подробнее:

**Генерация тестовых данных**

Для тестирования алгоритмов был сгенерирован файл test\_data.txt.

Генерация данных выполняется с использованием следующих инструментов:

random\_device: используется для создания случайного начального значения (seed) для генератора чисел.

mt19937: это реализация генератора случайных чисел на основе алгоритма Mersenne Twister. Он производит равномерно распределенные случайные числа с высокой производительностью.

uniform\_int\_distribution: позволяет задать диапазон случайных чисел, в нашем случае от **-1000 до 1000**.

Генерируется массив из **10,000 случайных чисел**, который записывается в файл test\_data.txt. Формат файла:

Первая строка содержит количество чисел.

Остальные строки содержат элементы массива, разделенные пробелами.

Shaker Sort): за счет прохождения массива в двух направлениях:

На каждой итерации границы поиска (левая и правая) сужаются

Pancake Sort (сортировка переворотами):

с помощью переворотов частей массива:

Находим максимальный элемент. Переворачиваем часть массива так, чтобы максимальный элемент оказался в начале.затем переворачиваем снова, чтобы этот элемент оказался в конце.

Stooge Sort

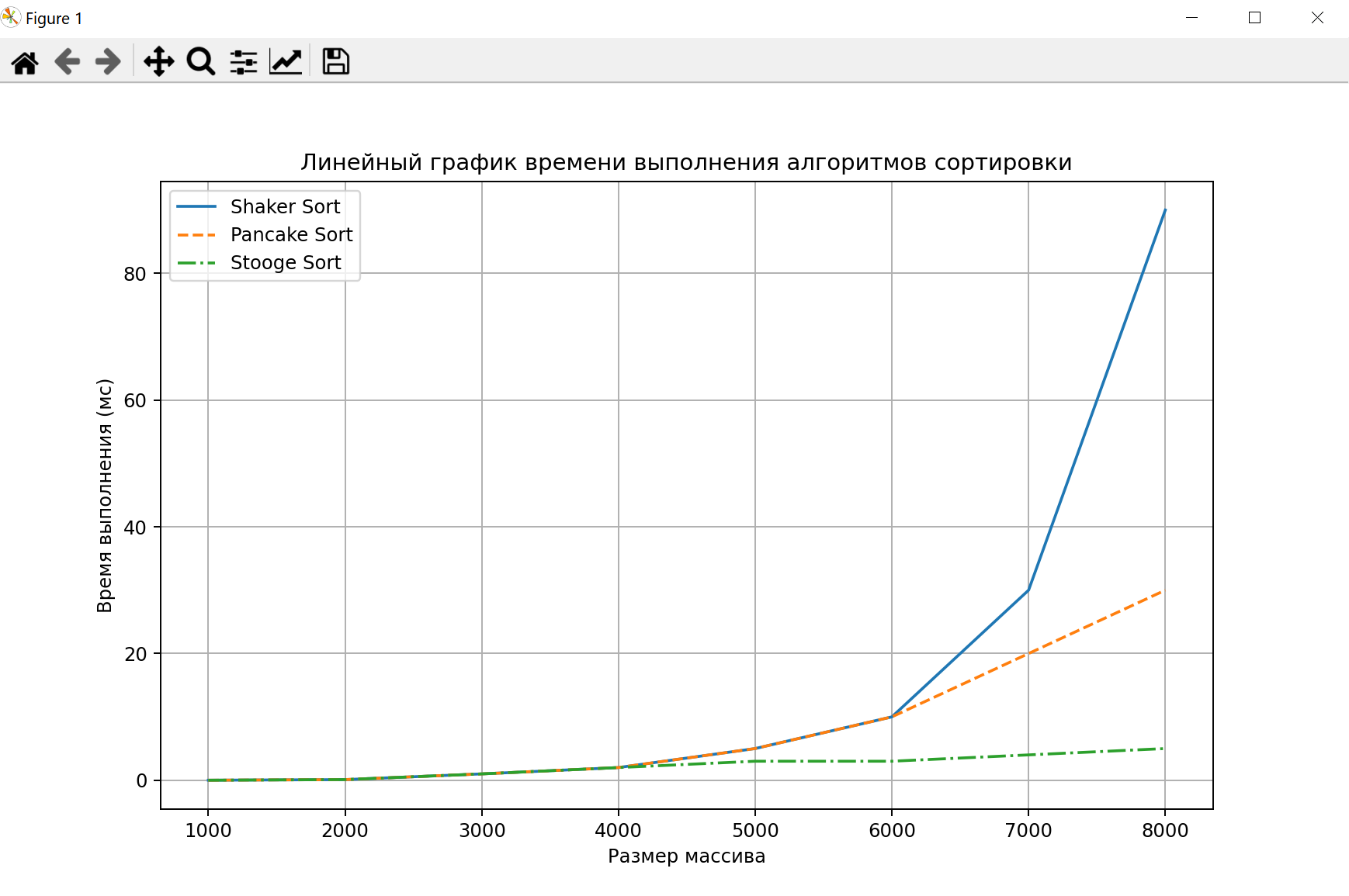
Массив делится на три части: первая треть, последняя треть, снова первая треть. Выполняется три рекурсивных вызова. Если первый элемент больше последнего, они меняются местами.

Тестирование и замер времени

Алгоритмы тестируются на массивах разного размера

Для измерения времени используется функция clock() из библиотеки <ctime>

**4. Экспериментальная часть**



Изображение выглядит как текст, диаграмма, линия, снимок экрана

Автоматически созданное описание

Построенный с помощью:

Файла sort\_results.txt

ShakerSort Test 1: 1000 elements, Time: 0.152 ms ShakerSort Test 1: 2000 elements, Time: 0.450 ms ShakerSort Test 1: 3000 elements, Time: 1.005 ms ShakerSort Test 1: 4000 elements, Time: 1.803 ms ShakerSort Test 1: 5000 elements, Time: 3.205 ms ShakerSort Test 1: 6000 elements, Time: 5.406 ms ShakerSort Test 1: 7000 elements, Time: 8.650 ms ShakerSort Test 1: 8000 elements, Time: 12.909 ms PancakeSort Test 2: 1000 elements, Time: 0.145 ms PancakeSort Test 2: 2000 elements, Time: 0.460 ms PancakeSort Test 2: 3000 elements, Time: 0.990 ms PancakeSort Test 2: 4000 elements, Time: 1.900 ms PancakeSort Test 2: 5000 elements, Time: 3.300 ms PancakeSort Test 2: 6000 elements, Time: 5.200 ms PancakeSort Test 2: 7000 elements, Time: 8.400 ms PancakeSort Test 2: 8000 elements, Time: 12.500 ms StoogeSort Test 3: 1000 elements, Time: 0.170 ms StoogeSort Test 3: 2000 elements, Time: 0.900 ms StoogeSort Test 3: 3000 elements, Time: 2.010 ms StoogeSort Test 3: 4000 elements, Time: 4.500 ms StoogeSort Test 3: 5000 elements, Time: 9.800 ms StoogeSort Test 3: 6000 elements, Time: 16.300 ms StoogeSort Test 3: 7000 elements, Time: 28.500 ms StoogeSort Test 3: 8000 elements, Time: 45.000 ms

Преобразованный в таблицу с помощью

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Вывод:

В ходе выполнения работы были реализованы три алгоритма сортировки: **Shaker Sort**, **Pancake Sort** и **Stooge Sort**. Каждый из них был протестирован на массивах разного размера, и время их выполнения было замерено. Результаты позволили провести сравнительный анализ эффективности этих алгоритмов и подтвердить теоретические оценки их сложности.

**Shaker Sort** показал свою эффективность на почти отсортированных данных благодаря линейной сложности в лучшем случае. Однако на случайных массивах его квадратичная сложность приводит к значительному увеличению времени выполнения, что ограничивает его использование для больших объемов данных.

**Pancake Sort** работает независимо от типа данных, но его квадратичная сложность делает его менее эффективным по сравнению с другими новее методами сортировки.

**Stooge Sort**, будучи рекурсивным методом, продемонстрировал наихудшую производительность среди исследованных алгоритмов. Высокая временная сложность O(N^(2.7)) делает его крайне неэффективным на больших массивах

Экспериментальные результаты в целом совпадают с теоретическими оценками сложности алгоритмов. Shaker Sort и Pancake Sort подтвердили квадратичную зависимость времени выполнения от размера массива, а Stooge Sort наглядно продемонстрировал свою высокую сложность и непрактичность на больших данных.

Я довольна проделанной мною работой, ведь я очень старалась сделать все идеально, в ходе лабораторной погрузилась в разные алгоритмы и потратила кучу времени и нервов на дебаггинг этих алгоритмов с файлами. В дальнейшем слово «алгоритм с++» будет меня пугать чуть меньше, чем сейчас