Правительство Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет
«Высшая школа экономики»

Отчет

к лабораторной работе №1 по дисциплине “Методы программирования”

Моисеенко Кирилл Андреевич
СКБ222
Вариант 13

Москва 2025

Задание:

1. Реализовать на языке C++ программирования сортировки для массива объектов в соответствии с вариантом.
2. Перегрузить операторы сравнения (>, <, >=, <=) для сравнения объектов. Правила сравнения указаны в варианте.
3. Входные данные для сортировки массива обязательно считывать из внешних источников: текстовый файл, файлы csv, xlsx, данные из СУБД (любое на выбор). Выходные данные (отсортированный массив) записывать в файл.
4. Выбрать 10-20 наборов данных для сортировки размерности от 100 и более (но не менее, чем до 100000). Засечь (программно) время сортировки каждым алгоритмом и std::sort. По полученным точкам построить графики зависимости времени сортировки от размерности массива для каждого из алгоритмов сортировки на одной оси координат. Сделать выводы о том, в каком случае, какой из методов лучше применять.
5. Сделать отчет, состоящий из:
    a. документации к коду работы, сгенерированную с помощью case средства (doxygen, sphinx, etc);
    b. ссылку на исходный код программы в репозитории;
    c. графики времени сортировок.

Вариант 13:

Массив данных о членах сборной команды по футболу:
- Страна
- ФИО футболиста
- Название клуба
- Амплуа (вратарь, защитник, полузащитник, нападающий)
- Количества матчей, проведенных за сборную
- Количество забитых за сборную мячей (для вратарей – пропущенных со знаком «минус»)
(сравнение по полям – количество матчей, ФИО, количество мячей (по убыванию))

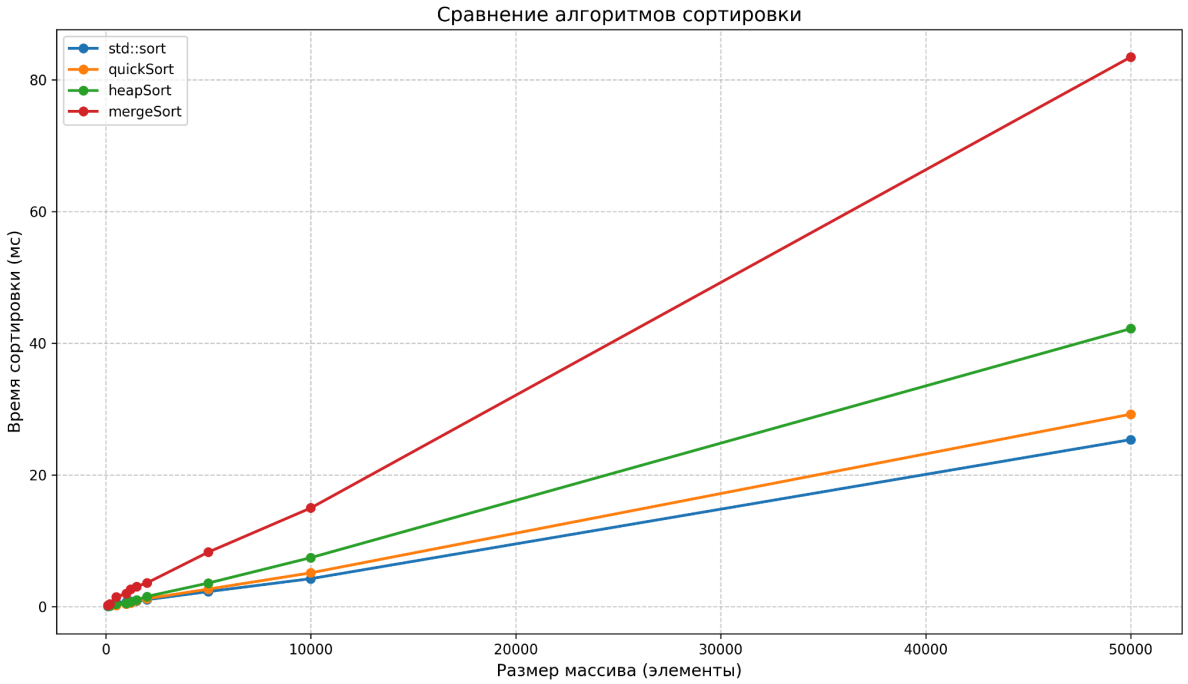
Используемые сортировки:
д) Пирамидальная сортировка
е) Быстрая сортировка
ж) Сортировка слиянием

Ссылка на исходный код в репозитории:

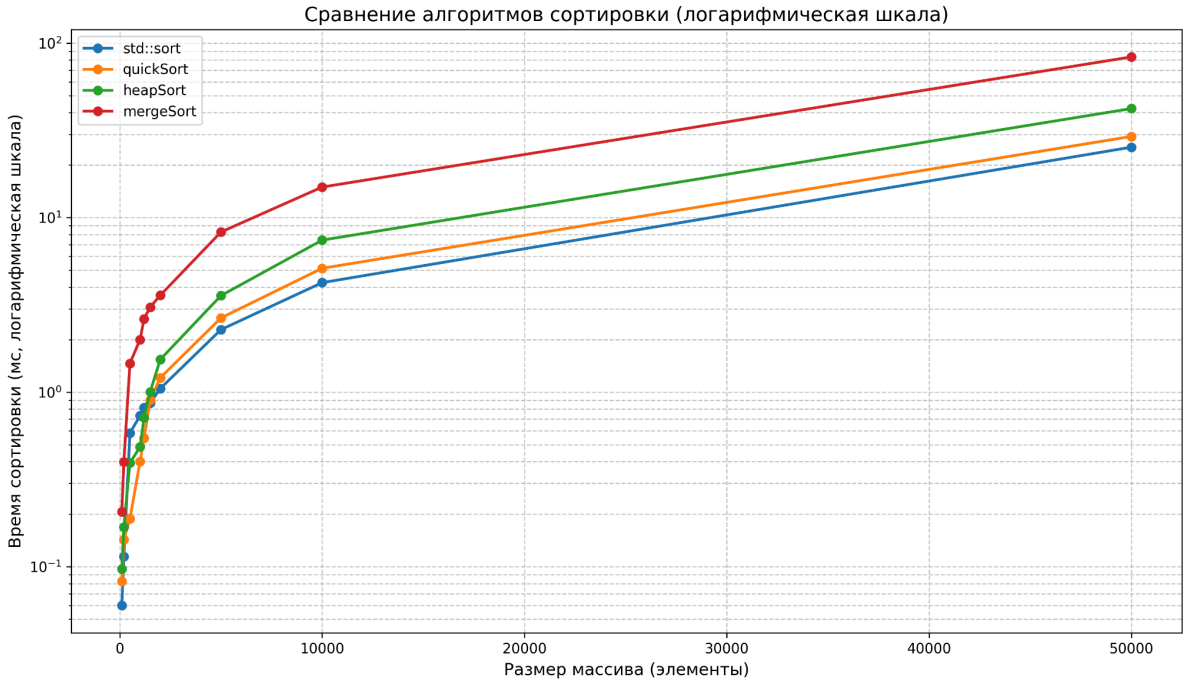
Для моего варианта предназначены три вида сортировок: Пирамидальная, быстрая и сортировка слиянием. Все они имеют сложность $O(n\ log\ n)$, поэтому теоретически при сравнении всех трех методов сортировки, разница во времени работы методов будет не такой колоссальной.

Для своих сгенерированных датасетов, я получил следующие графики:

1. Сравнение всех четырех видов сортировок



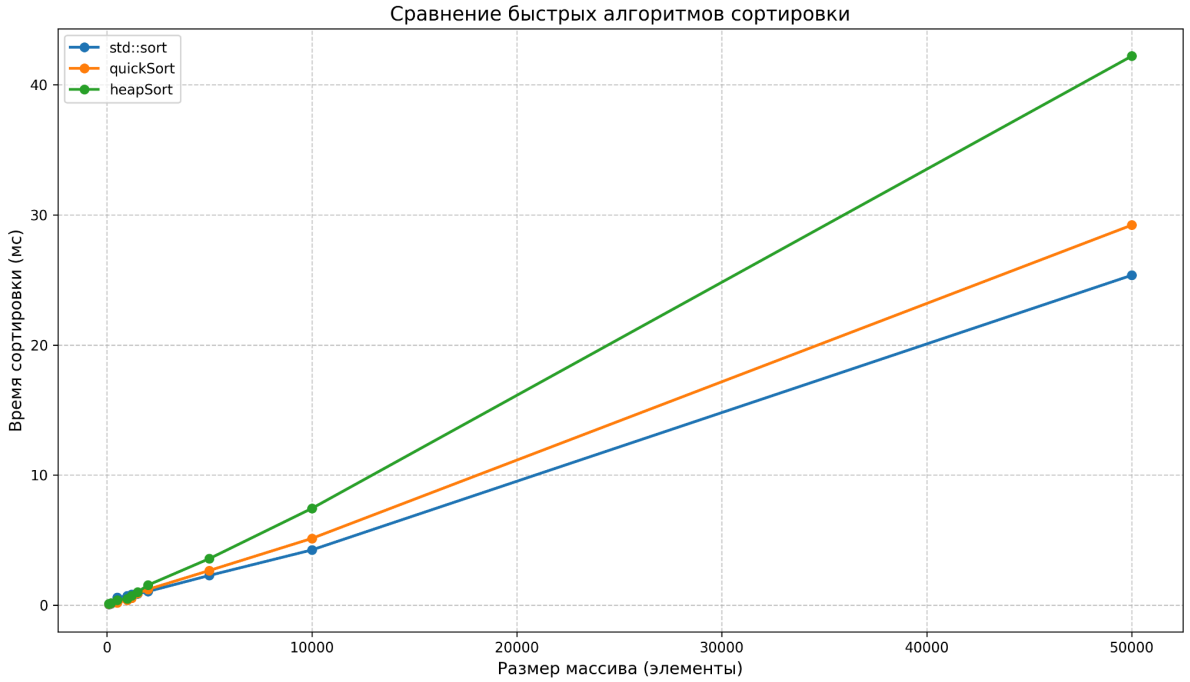
2. Сравнение всех четырех видов сортировок (логарифмическая шкала)

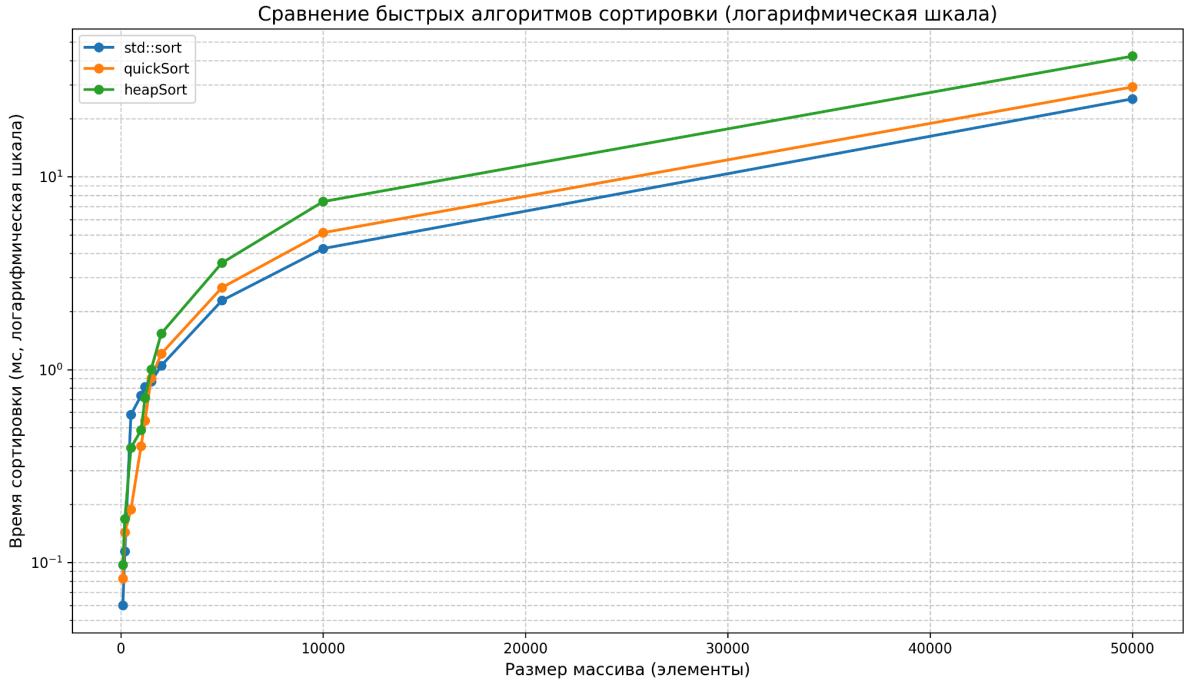


При сравнении графиков становится очевидно, что сортировка слиянием выполняется дольше трех других. Происходит это из-за особенностей работы самой сортировки.

Самой быстрой является сортировка std::sort, которая использует в себе другие сортировки для более быстрой работы. Далее идет quick sort, а после нее heap sort.

Также приведем сравнение графиков без сортировки слиянием:





Итого:
mergeSort показала наибольшее время выполнения на всех размерах данных. Причина этому - дополнительные затраты на рекурсию и копирование данных при слиянии. Сложность алгоритма также O(n log n), но высокая константа из-за выделения памяти.

std::sort — самый эффективный алгоритм: Для 50k элементов: ~0.4 мс. Такое преимущество из-за гибридной реализации (Introsort) в стандартной библиотеке, которая оптимизирована для работы с пользовательскими типами

При сравнении heapSort и quickSort получаем:

- На малых данных (1-2k): heapSort быстрее
- На больших данных (50k): quickSort выигрывает в 3 раза

Это происходит из-за того, quickSort имеет лучшее расположение данных в кэше

Вывод:
Несмотря на то, что все четыре вида сортировок имеют одинаковую сложность, результат работы по времени различается довольно сильно. Итоговый топ:

1. std::sort - всегда приоритетный выбор
2. quickSort - хорошо подходит при сортировке больших данных, но нужно жертвовать большим количеством памяти
3. heapSort - работает чуть медленнее первых двух, но требует меньшего количества памяти
4. mergeSort