ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 5

«Алгоритмы сортировки»

Выполнил работу

Трегубов Вадим

Академическая группа C3100

Принято

Ментор, Вершинин Владислав Константинович

Санкт-Петербург

2024

1. Введение

Цель работы: разработка и изучение алгоритмов сортировки массивов.

Задачи

1.Изучить основные принципы работы алгоритмов сортировки.

2.Реализовать три алгоритма сортировки: Bucket Sort, Heap Sort и Shaker Sort.

3.Проанализировать теоретическую сложность алгоритмов.

2. Теоретическая подготовка

Сортировка — это процесс упорядочивания массива элементов в определенном порядке (возрастания или убывания).

Используемые алгоритмы

1.Bucket Sort

Описание: делит массив на корзины, сортирует каждую и объединяет результат.

Сложность:

Лучший случай: O (n + k), где k — количество корзин.

Худший случай: O(n²), если все элементы попадают в одну корзину.

2.Heap Sort

Описание: использует кучу для построения отсортированного массива.

Сложность: O (n log n) во всех случаях.

3.Shaker Sort

Описание: Модифицированная пузырьковая сортировка с движением элементов в обе стороны.

Сложность:

Лучший случай: O(n), если массив уже отсортирован.

Худший случай: O(n²).

3. Реализация

Этапы реализации

1.Bucket Sort

Генерация корзин: Элементы делятся на k корзин в зависимости от их значений.

Сортировка корзин: Каждая корзина сортируется отдельно.

Слияние корзин: Отсортированные корзины объединяются в общий массив.

2.Heap Sort

Построение кучи: Массив преобразуется в кучу (свойство: родительский элемент больше дочерних).

Сортировка: Максимальный элемент удаляется из кучи, помещается в конец массива, а куча перестраивается.

3.Shaker Sort

Проход слева направо: Самый большой элемент перемещается в конец массива.

Проход справа налево: Самый маленький элемент перемещается в начало массива.

Сокращение диапазона: после каждого прохода уменьшается область неотсортированных элементов.

Использование библиотек:

#include <iostream> — Для ввода и вывода данных.

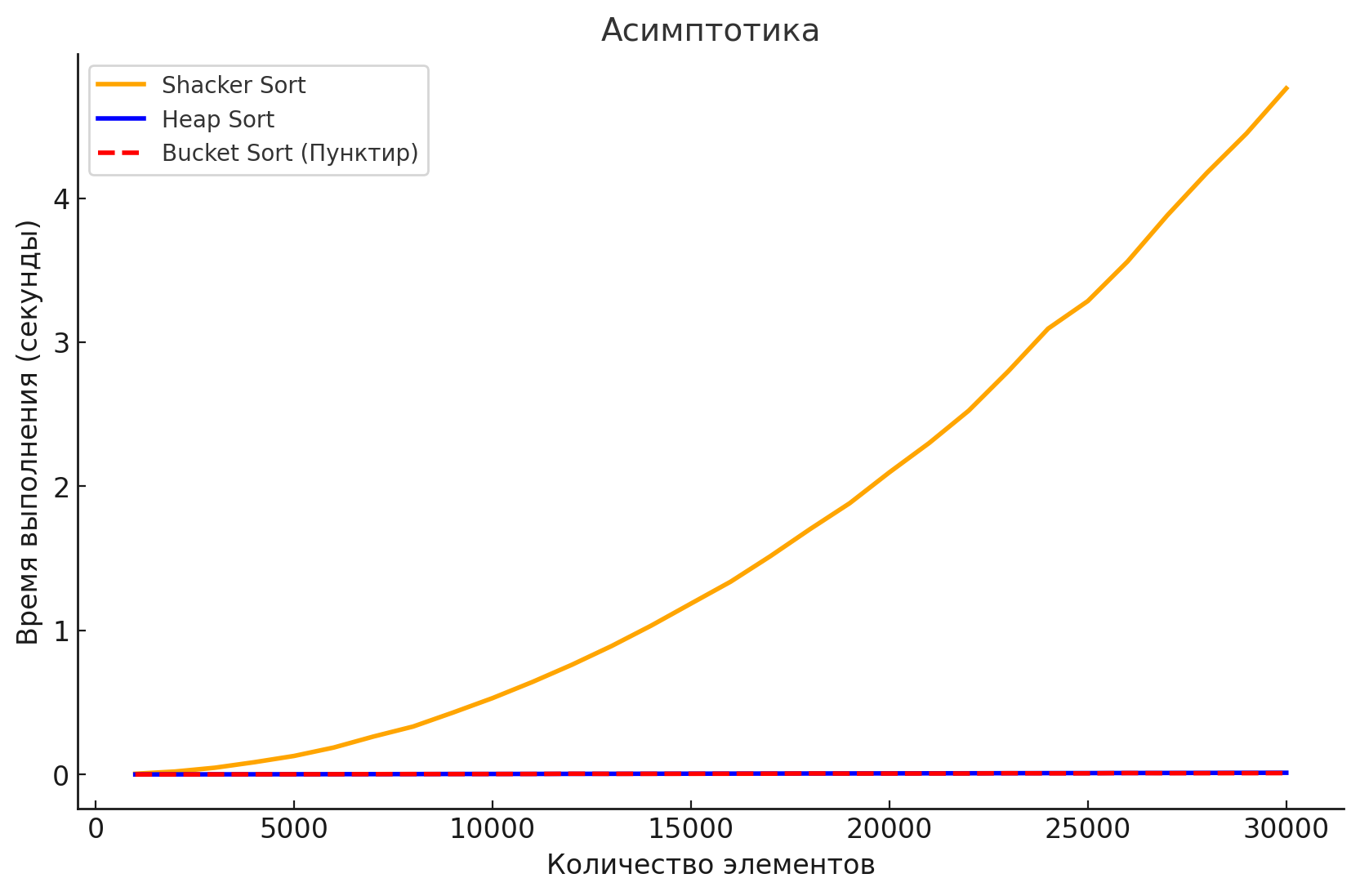
#include <vector> — Для работы с динамическими массивами.

#include <algorithm> — Для сортировки внутри корзин.

#include <chrono> — Для контроля времени выполнения.

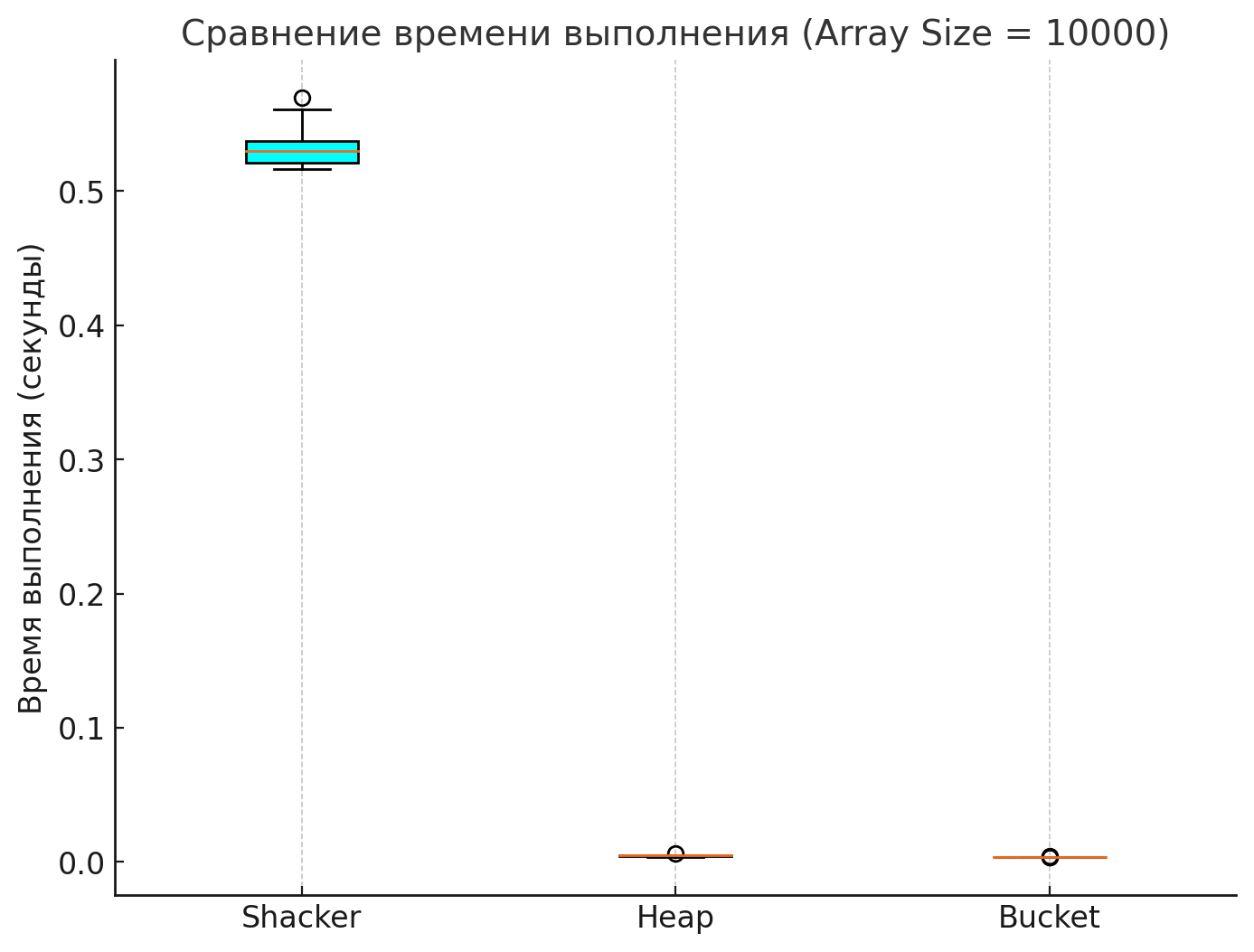
1. Экспериментальная часть

Для оценки эффективности алгоритма было рассчитано время для работы алгоритмов, для массивов от 1000 до 30000 с шагом в 1000. Результат представлен на изображении №1.

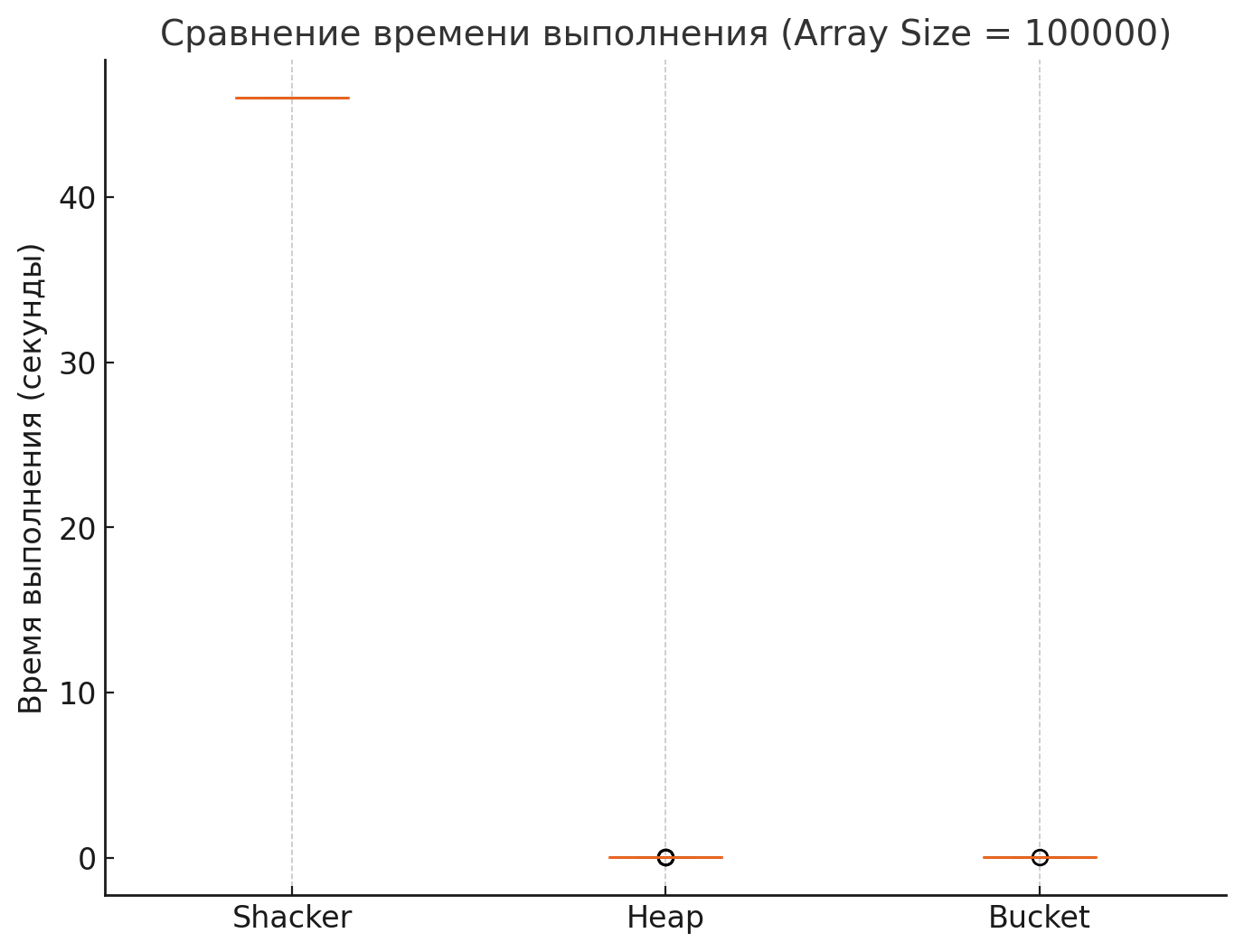


Изображение №1 - График работы алгоритмов

Для анализа времени работы алгоритмов были созданы box plot для 1e4 и 1e5 элементов. Результат представлен на изображении №2 и №3.



Изображение №2 - Box plot (10000 size)



Изображение №3 – box plot (100000 size)

5. Заключение

1. Производительность: Shacker Sort показывает низкую эффективность на больших массивах, тогда как Heap и Bucket Sort работают быстрее и стабильнее.
2. Выбросы: у Heap и Bucket есть небольшие выбросы из-за системных факторов или распределения данных. У Shacker они незначительны.
3. Применение: Shacker подходит для маленьких массивов или учебных целей, Heap - универсален, Bucket эффективен для равномерных данных.
4. Заключение: Heap и Bucket лучше подходят для работы с большими массивами. Shacker стоит использовать только на небольших данных.