МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Физтех-школа аэрокосмических технологий

Лбораторная работа № 1

Эффективность алгоритмов сортировки

Автор:
Леонид Ефремов
Б03-403

Долгопрудный 2024

Содержание

1 Введение

В данной лабораторной работе рассматриваются основные алгоритмы сортировки, их временная сложность и эффективность в зависимости от различных факторов, таких как размер входных данных и их первоначальная упорядоченность. Основное внимание уделяется анализу следующих алгоритмов: сортировка пузырьком, сортировка выбором, сортировка вставками, быстрая сортировка, сортировка кучей и сортировка слиянием.

Целью лабораторной работы является не только понимание теоретических аспектов временной сложности, но и приобретение практических навыков в реализации и тестировании алгоритмов сортировки. Результаты работы помогут лучше осознать, как выбор алгоритма может влиять на эффективность обработки данных в реальных приложениях.

1.1 Эксперимент

Для проведения эксперимента использовался компьютер на базе процессора "Ryzen 5 3600"в однопоточном режиме с фиксацией тактовой частоты на 3.6 ГГц при постоянном температурном режиме, двухканальная память ddr4 с фиксированной частотой на 2.4 ГГц. Замер времени производится через каждые 100 добавленных элементов. Управляющий код в приложении.

2 Алгоритмы асимптротики $O(n^2)$

2.1 Демонстрация

Для начала рассмотрим алгоритмы, простые в написании.

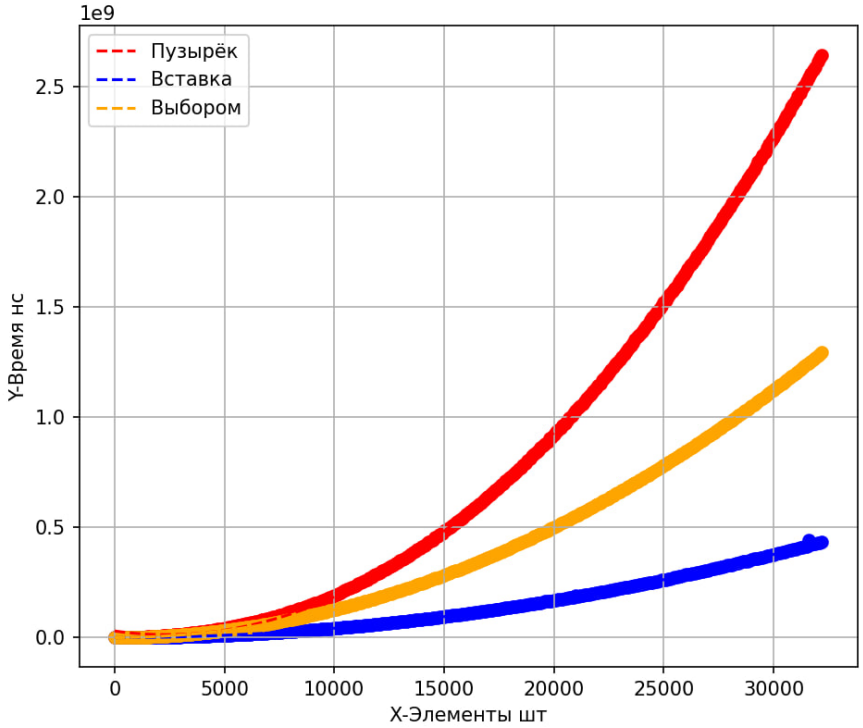


Рис. 1: Сложность $O(n^2)$

Для доказательства такой сложности используем логарифмический масштаб: $ln(t) = ln(C) + 2lnN$ Имеем линейный вид графиков, квадратичная зависимость подтверждена.

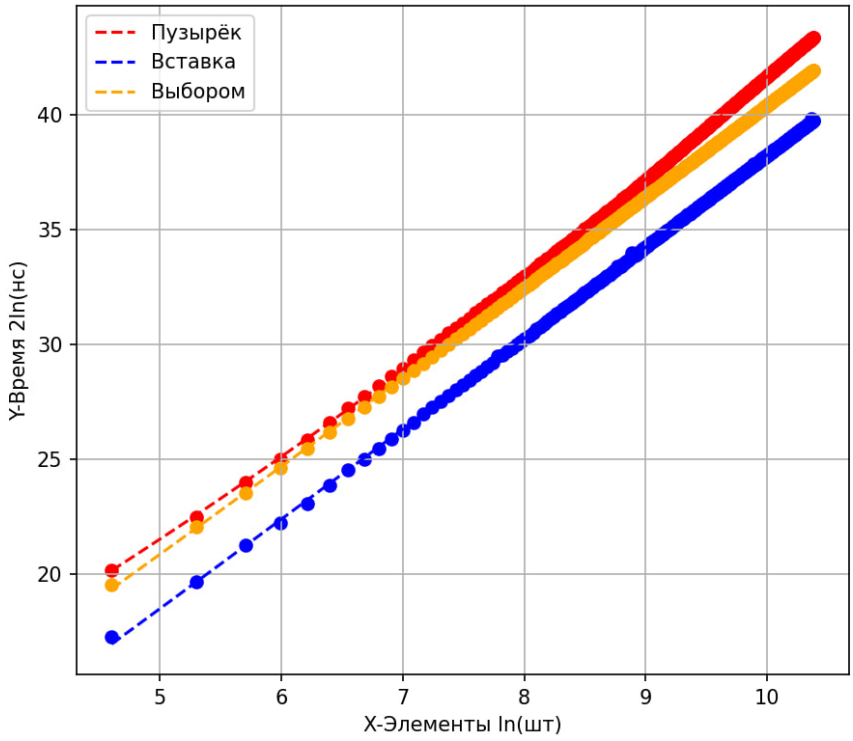


Рис. 2: Доказательство сложности $O(n^2)$

2.2 Сравнение оптимизаций

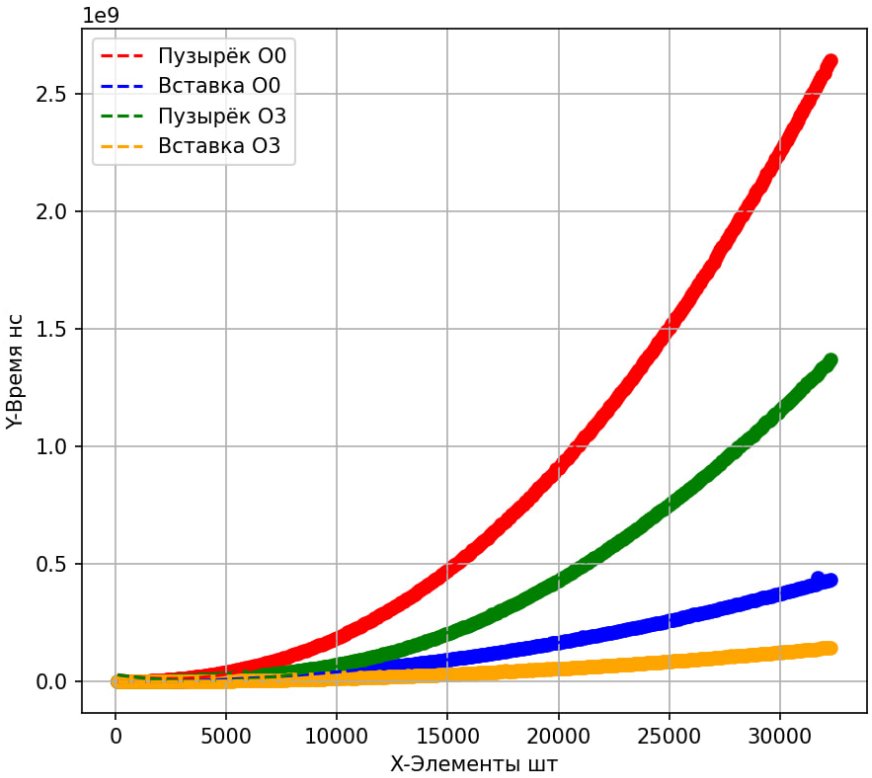


Рис. 3: Демонстрация эффективности оптимизации для квадратичных алгоритмов

3 Алгоритмы асимптротики $O(NlogN)$

3.1 Демонстрация

Сравним эффективные алгоритмы сортировки между собой.

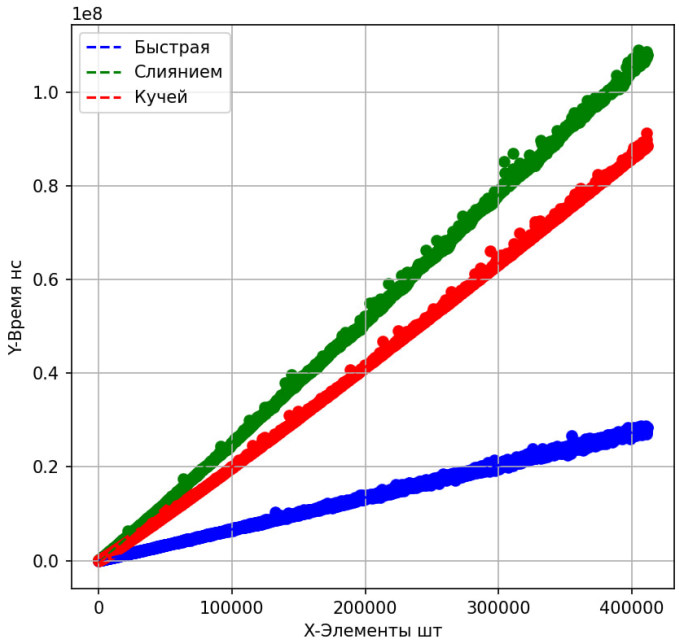


Рис. 4: Сложность $O(NlogN)$

Для доказательства такой сложности используем логарифмический масштаб: $t = t/NlnN$

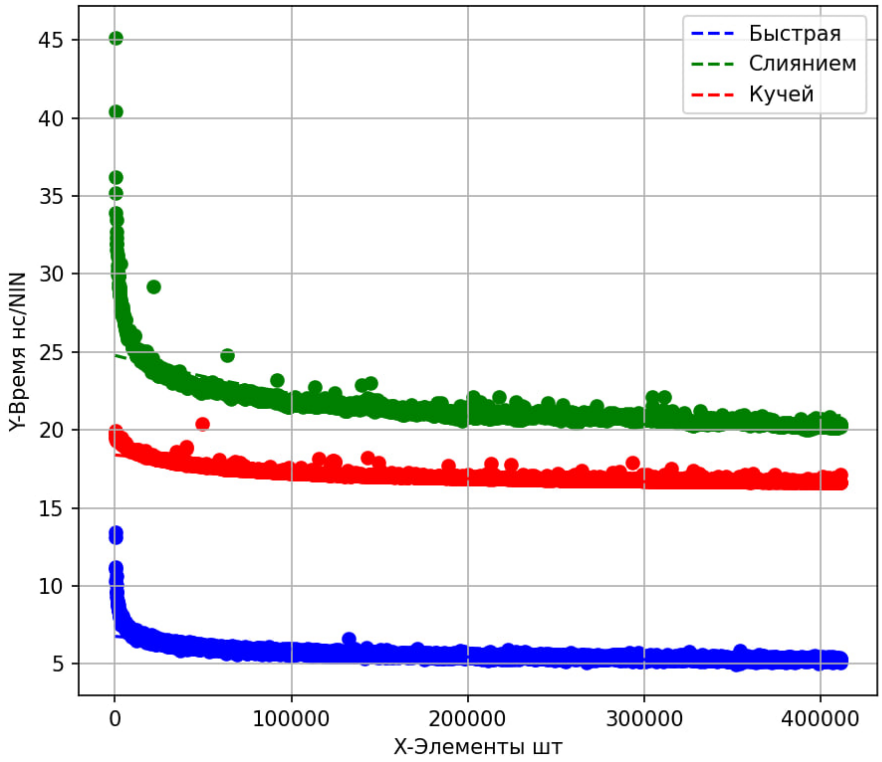


Рис. 5: Доказательство сложности $O(NlogN)$

Имеем линейный вид графиков, логарифмическая зависимость подтверждена.

3.2 Сравнение оптимизаций

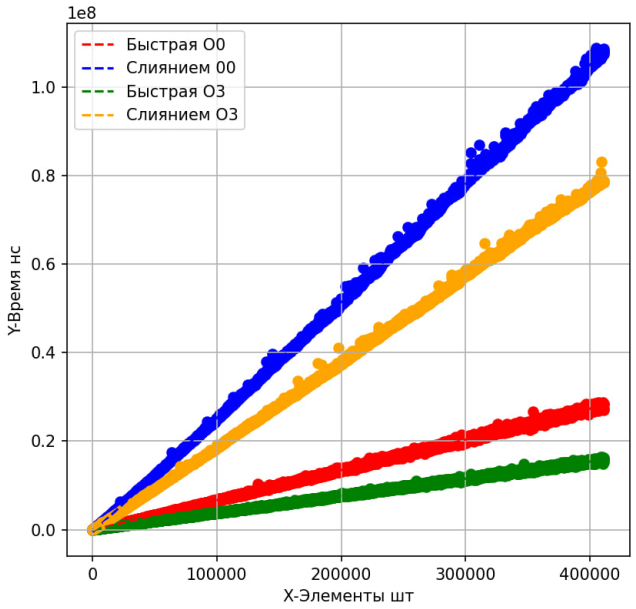


Рис. 6: Демонстрация эффективности оптимизации для логарифмических алгоритмов

Быстрая сортировка оправдывает своё имя, даже её неоптимизированная версия превосходит конкурентов.

4 $O(NlogN)$ и $O(N^2)$

Для наглядности сравним алгоритмы сортировки разных рангов. Для более объективной картины, отключим оптимизации:

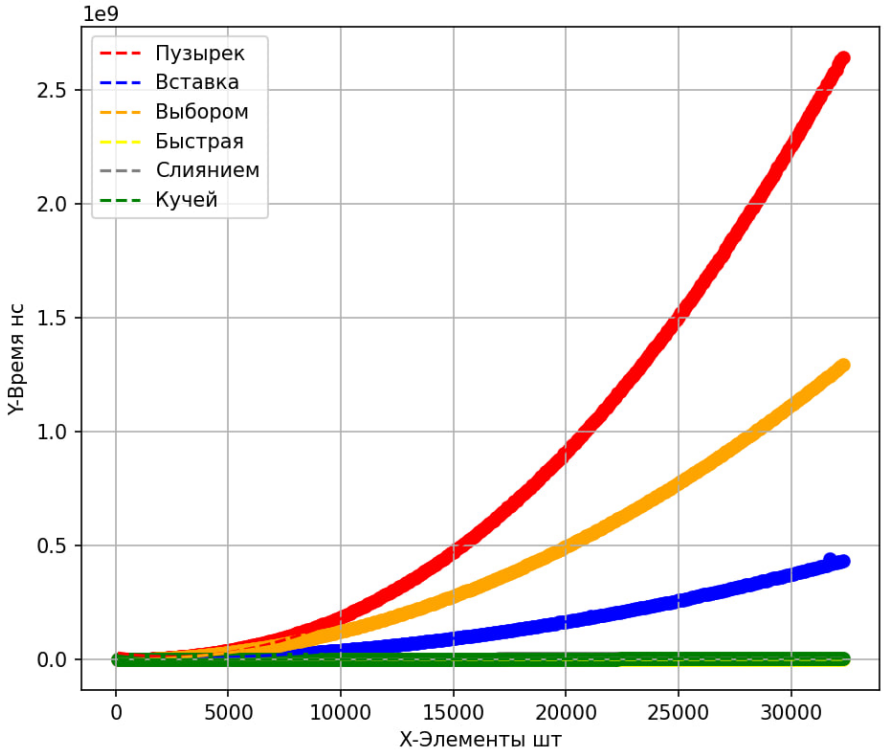
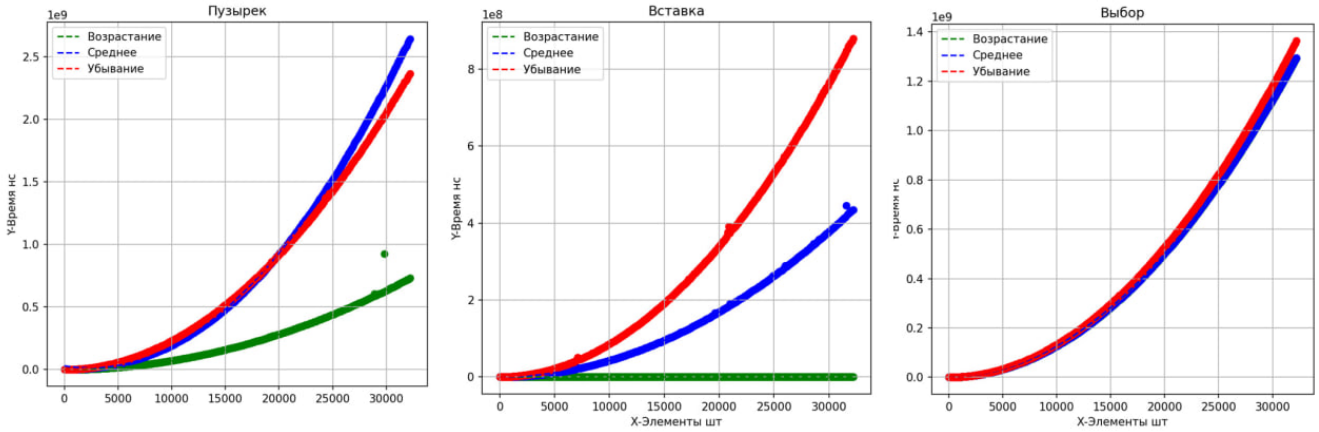
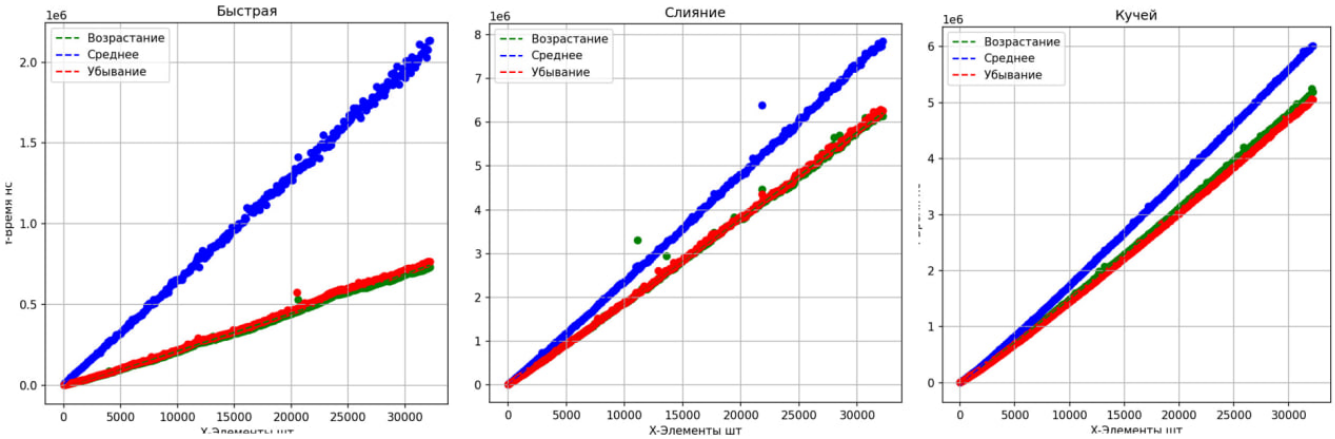


Рис. 7: Все алгоритмы сортировки

5 Зависимость от начальных данных

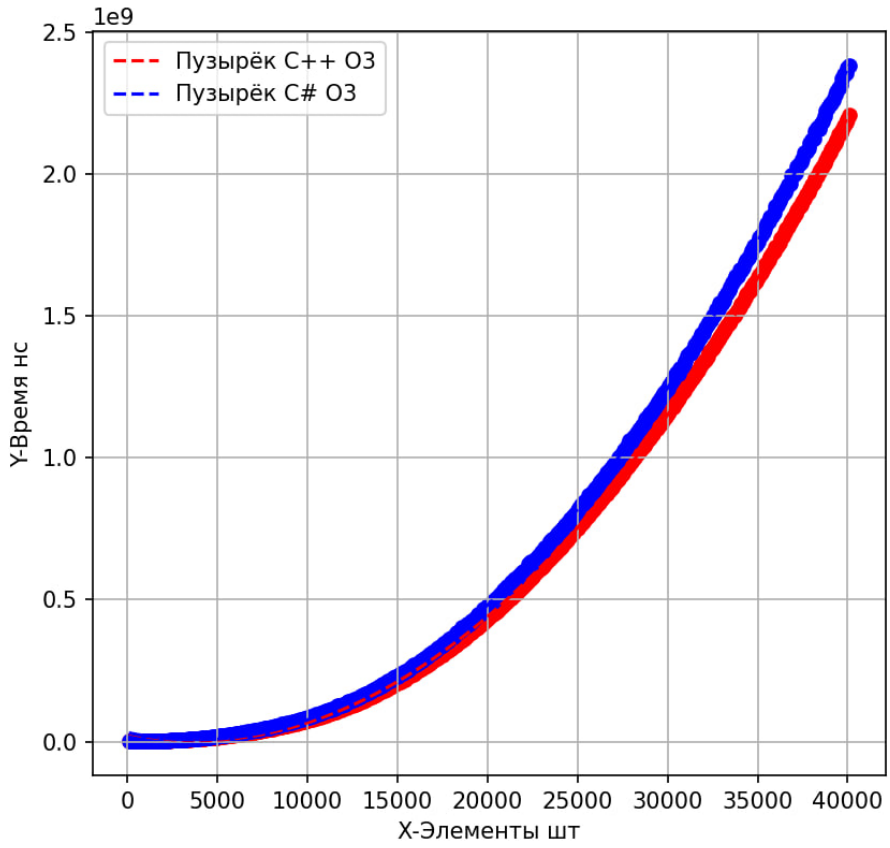
Используем случайный, возрастающий и убывающий массив чисел:



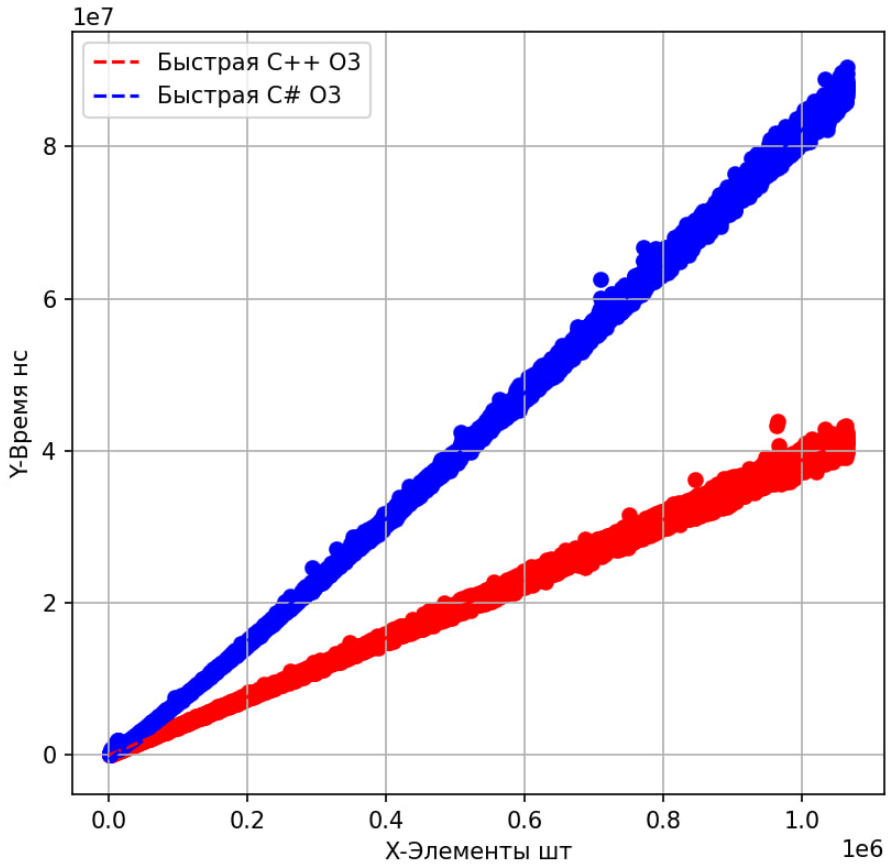


6 C++ VS C#

Так как Python очевидно медленнее C++, сравним отца и его блудного объектно-ориентированного сына. Здесь уже схватка насмерть - максимальная оптимизация на обоих языках:

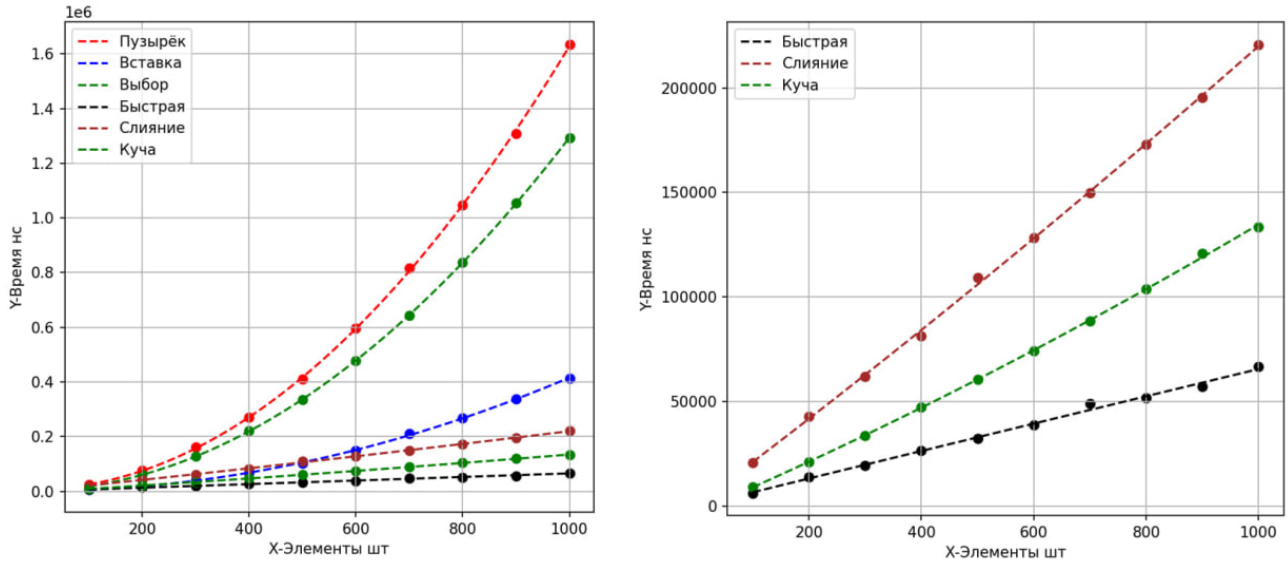


В первом раунде разница в производительности по времени неочевидна, однако C# убивает C++, если речь заходит о удобстве написания кода и количестве подводных камней. Устроим бой на более совершенных алгоритмах:



В быстрых алгоритмах ситуация печальнее: C++ значительно выигрывает.

7 Сравнение мощности на малых массивах



Благодаря грамотной конфигукации эксперимента, всякие артефакты при замере времени отсутствуют даже при малых массивах. Быстрая сортировка - фаворит даже при малых массивах. Квадратичная сортировка вставкой эффективнее логарифмической сортировки слиянием при входных массивах длиной меньше 500 элементов.

8 Вывод

В ходе выполнения лабораторной работы была проведена сравнительная оценка временной эффективности различных алгоритмов сортировки, таких как пузырьковая сортировка, сортировка вставками, сортировка выбором, быстрая сортировка, сортировка кучей и сортировка слиянием.

Мы проанализировали время выполнения каждого алгоритма на различных наборах данных, включая уже отсортированные, случайные и обратные последовательности. Результаты эксперимента показали, что временная сложность алгоритмов существенно зависит от структуры входных данных.

1. Пузырьковая сортировка и сортировка выбором продемонстрировали наименьшую эффективность, особенно на больших объемах данных, с временной сложностью $O(n^2)$. Эти алгоритмы не рекомендуются для использования в реальных задачах, требующих быстрой обработки.

2. Быстрая сортировка, сортировка кучей и сортировка слиянием оказались наиболее эффективными для больших массивов, с временной сложностью $O(NlogN)$ в среднем случае. Быстрая сортировка оказалась быстрее во всех тестах. В реальных приложениях предпочтение следует отдавать более эффективным алгоритмам, таким как быстрая сортировка или сортировка слиянием, особенно при работе с большими объемами данных.

Полученные результаты подчеркивают важность анализа временной эффективности алгоритмов и выбора языка программирования при разработке программного обеспечения и оптимизации вычислительных процессов.