Лабораторная работа
Исследование сортировок

10 сентября 2024 г.

Гальперин Тимофей
Б03-404

---

1 Цель

Цель работы - оценить время работы алгоритмов сортировки, сравнить с их асимптотикой.

2 Арихитектура кода

2.1 sortes.hpp

В этом файле собрано 6 сортировок: bubble, insertion, selection, quick, heap, merge. У каждой функции есть шаблон элемента T у которого 2 условия:

1. T должно иметь operator>
2. T должно иметь конструктор для int

2.2 helper.hpp

Содержит вспомогательные функции для подсчета времени обработки массива (отсортированного по возрастание, рандомного и отсортированного по убыванию) при заданном диапозоне количества элементов, наборе используемых сортировок, а также количестве повторений для точности.

2.3 main.cpp

Из-за особенности C++ для каждого типа элемента вызывает подсчет времени работы согласно config.json .

2.4 config.json

Конфигурационный файл для создания базы даннаых времени обработки каждого варианта (output.json).

2.5 Graphs.ipynb

Обработка полученной базы данных, создание графиков

3 Доказательство асимптотики $O(N^2)$

Рассмотрим сортировки bubble, insertion, selection рандомного массива типа int на значениях от 1000 до 20000 с шагом 1000 и точностью 3:

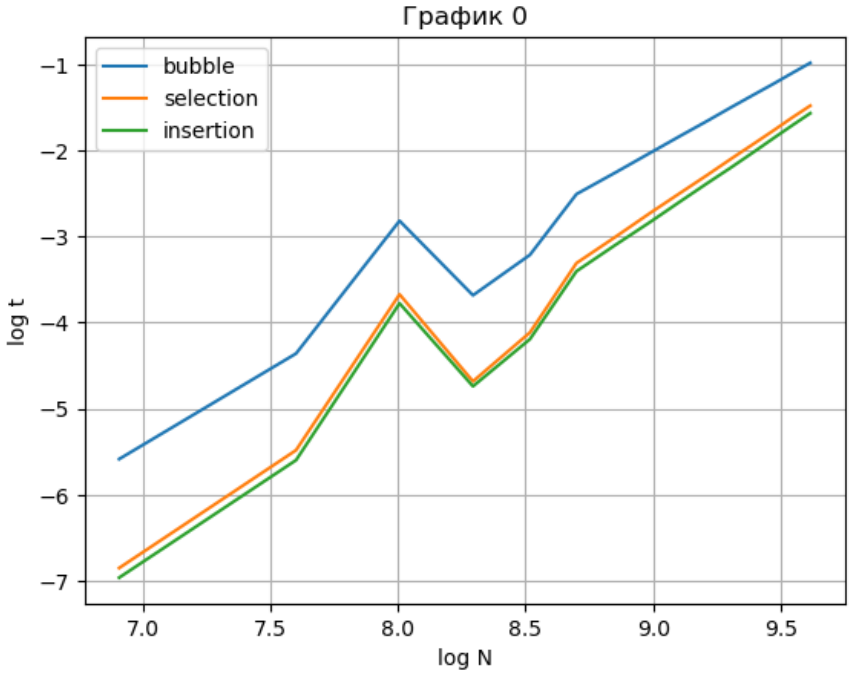


Рис. 1: log - натуральный логарифм, t - время работы в наносекундах, N - количество элементов

Если действительно $t = C * N^2$, то $logt = logC + 2 * logN$. Заметим, что не считая области с $logN = 8$ ($N \approx 2000$) график представляет собой прямую. Посчитаем угловой коэффицент:

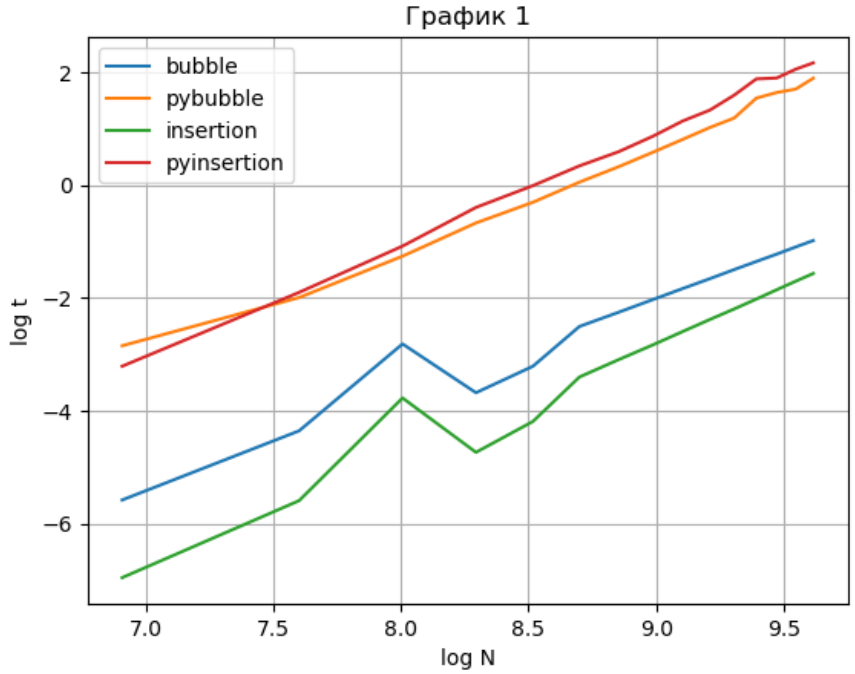
$$x = logN \qquad (x1, y1) = (7, -7)$$
$$y = logt \qquad (x2, y2) = (9.5, -2)$$

$$a = \frac{dy}{dx} = \frac{y2 - y1}{x2 - x1} = \frac{5}{2.5} = 2$$

Ч.Т.Д, также из графика можно заметить, что selection и insertion имеют гораздо меньший коэффицент C, а значит и работают быстрее

4 Сравнение Python и C++

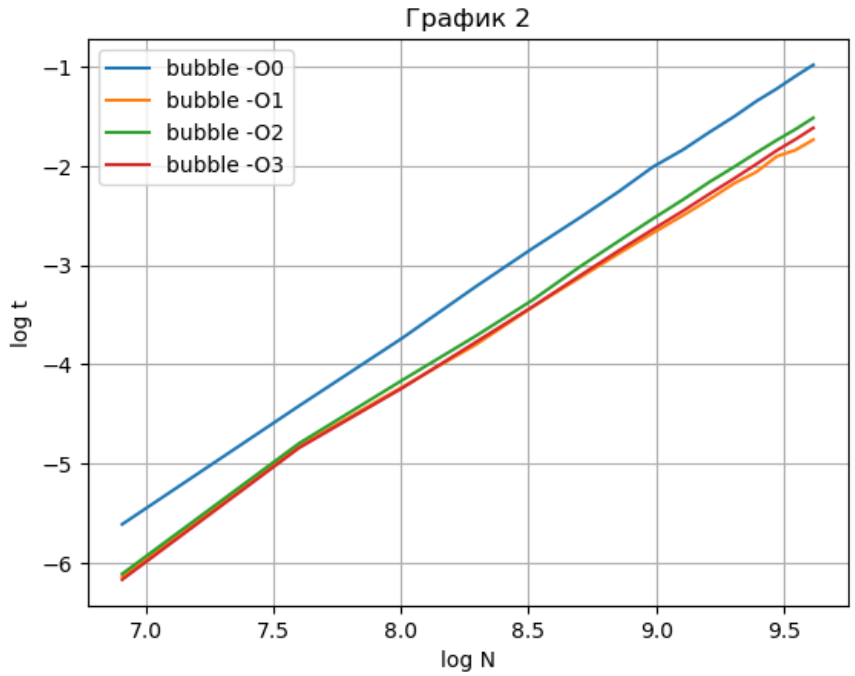
Напишем копию сортировок bubble и insertion на python и сравним в тех же координатах графики:



Время работы отличается почти в e раз, а пик "проблем"приходится на $logN = 9.4$ ($N \approx 12000$)

5 Оптимизация компилятора

Предыдущие значения были получены при -O0, добавим другие флаги для сравнения влияния оптимизации компилятора на примере bubble:



Переход на -O1 значительно ускорил сортировку, а дальнейшие почти не влияют при выбранном диапазоне (1000 - 20000)

6 Быстрые сортировки

Докажим асимптотику $O(NlogN)$ для быстрых сортировок:

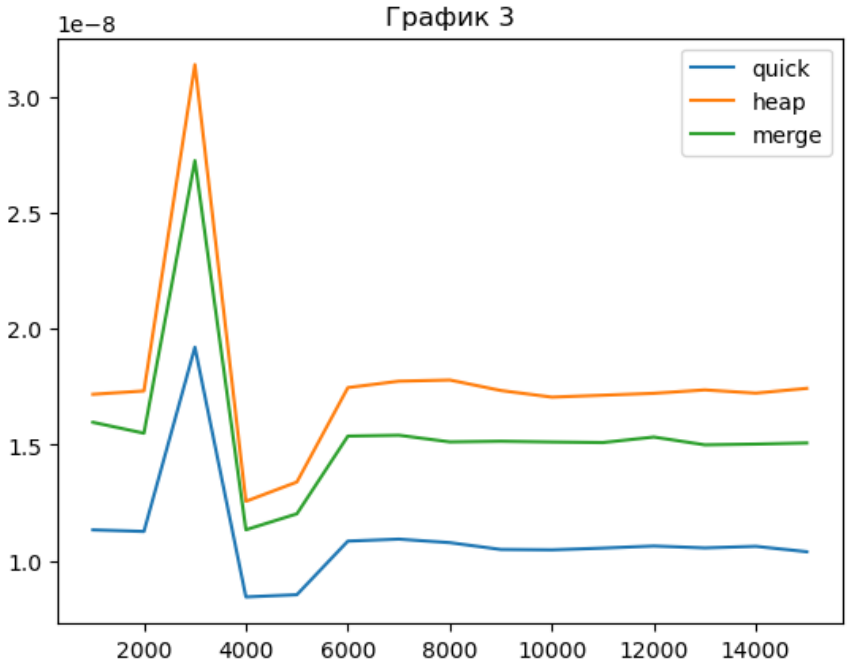
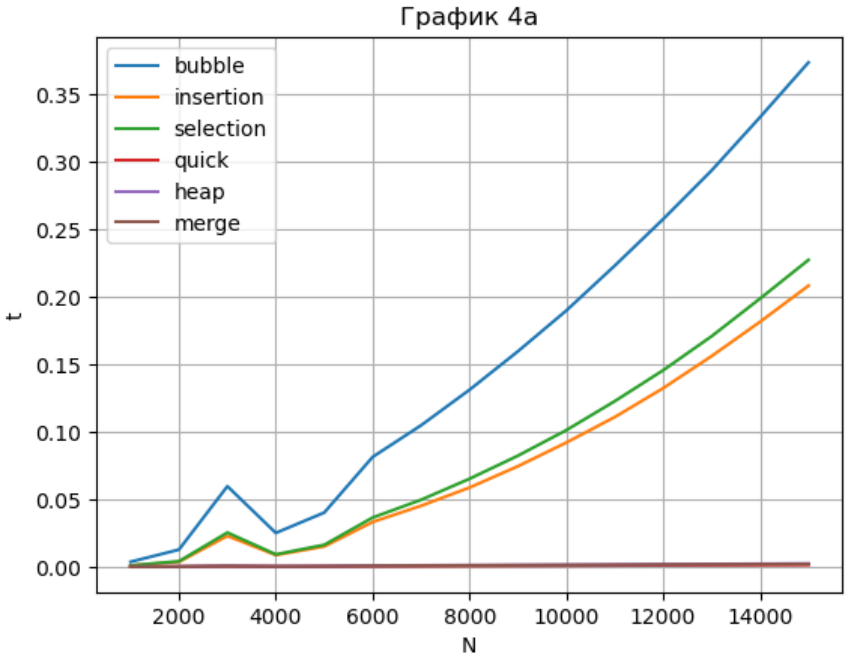


Рис. 2: По оси ординат $\frac{t}{NlogN}$, по оси абцисс N

Так как $\frac{t}{NlogN}$ примерно равно 1, то для быстрых сортировок действительно асимптотика $O(NlogN)$. Заметим также, что в наличии "проблемный"пик около 3000

7 $O(N^2)$ vs $O(N * logN)$

Сравним сортировки с разной асимптотикой:



$O(N * logN)$ во много раз быстрее на больших размерах массива

8 Начальные значения

Теперь исследуем как меняется асимптотика в зависимости от начального массива у разных сортировок

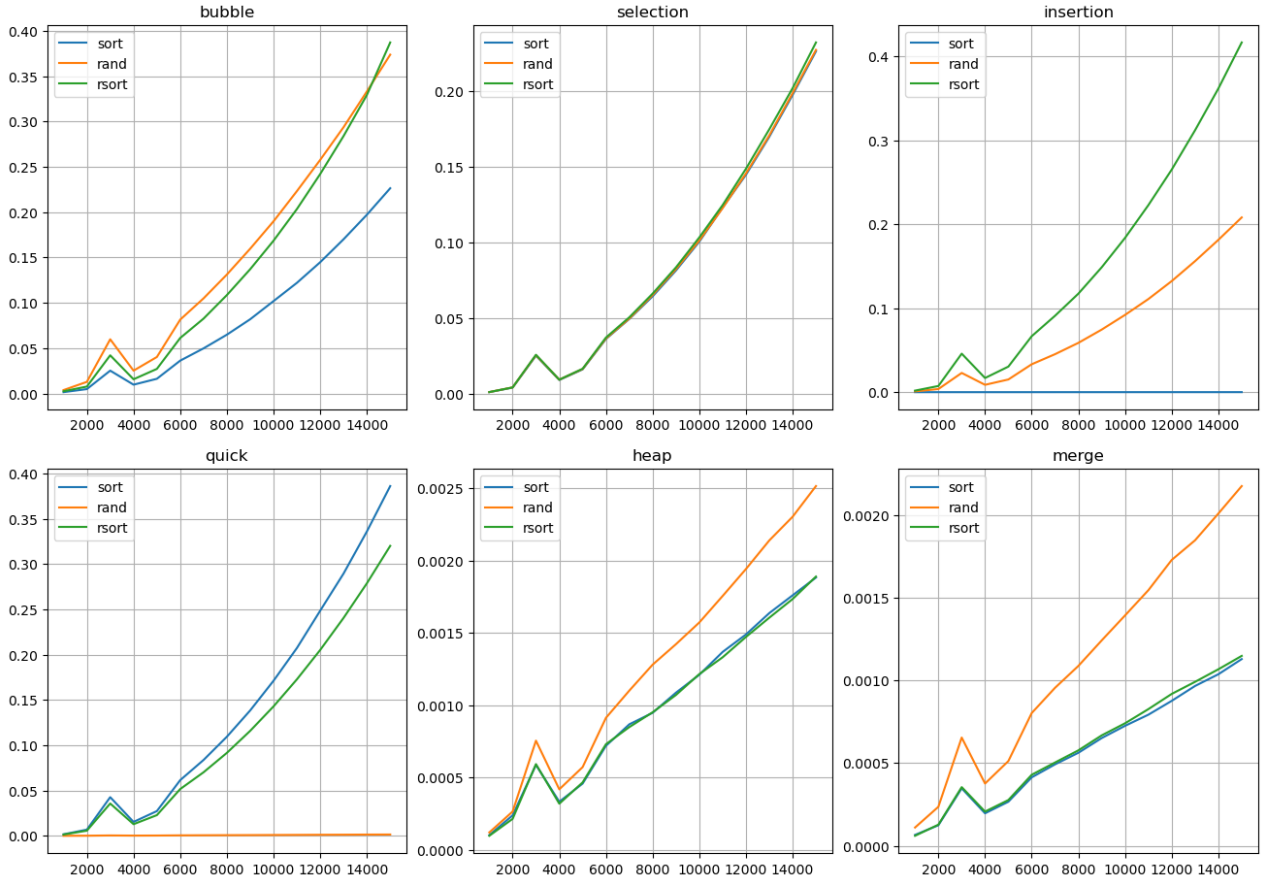
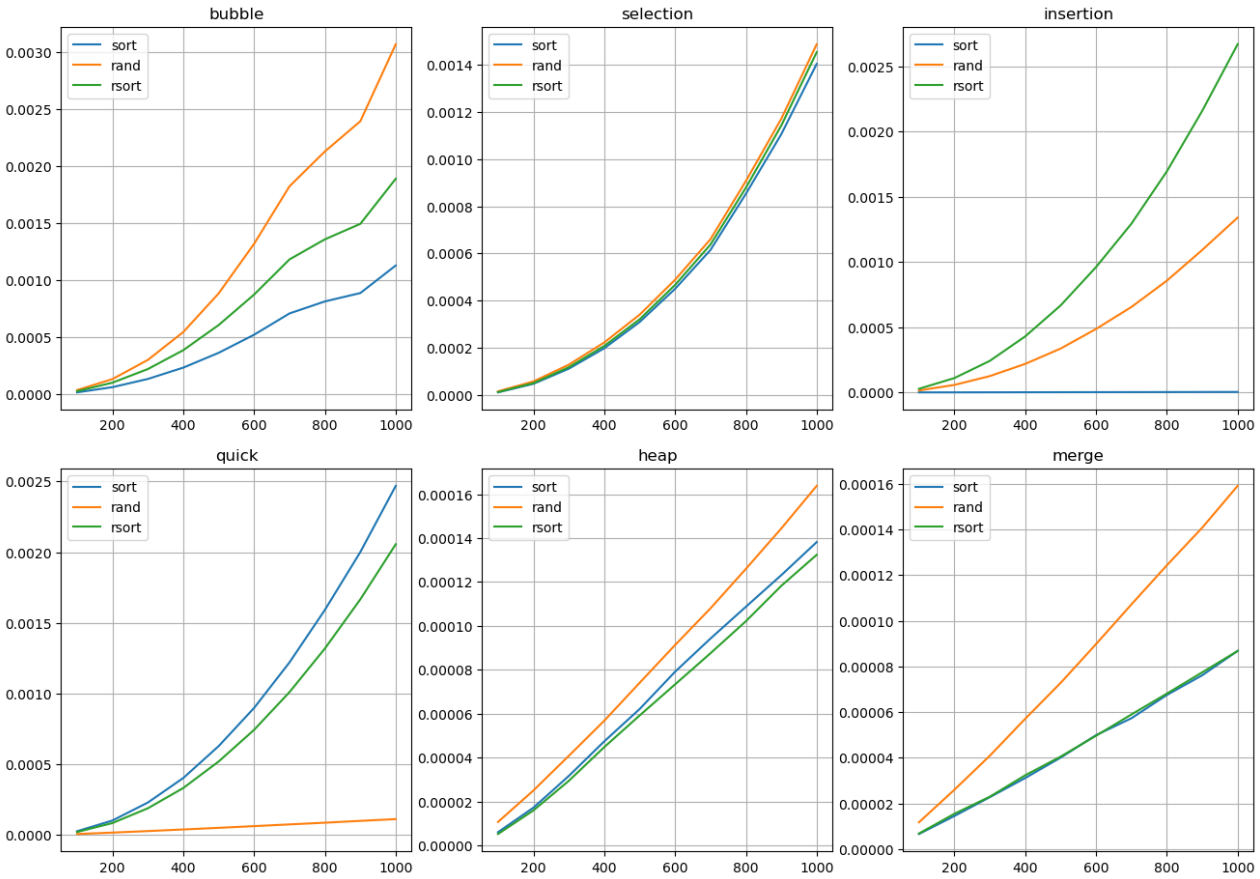


Рис. 3: sort - отсортированный массив, rand - рандомные значения, rsort - массив отсортированный в обратном порядке

Быстрые сортировки имеют схожую асимптотику для сортированных массивов. Selection неважно какой массив, асимптотика почти одинаковая

9 Малые массивы

Асимптотика для малых значений, точность составляет 70 измерений на одно количество элементов



Благодаря точности графики получились очень гладкими, заметные изменения только у bubble

10 Тип данных

Для исследования зависимости асимпототики от типа значений массива в helper.hpp создана "тяжелая"структура, содержащая 100 значений int.

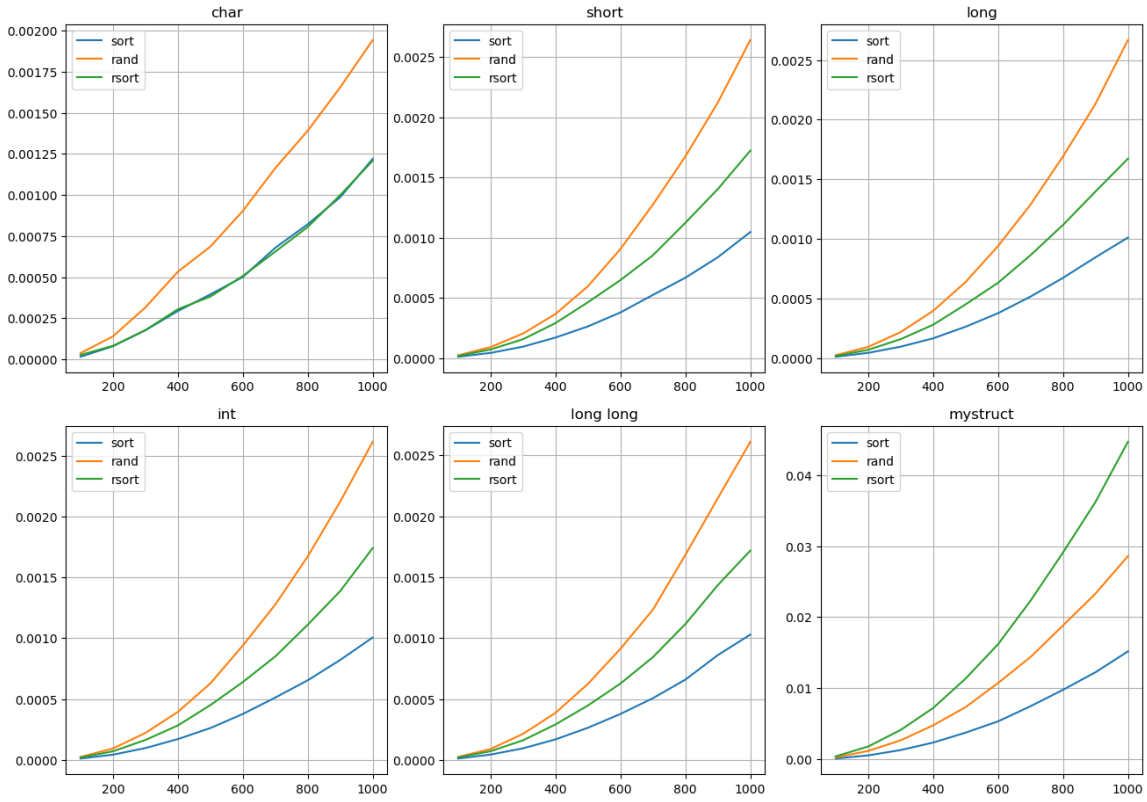


Рис. 4: Используется сортировка bubble для малых размеров массива

Чем меньше "вес"структуры, тем на порядок быстрее происходит сортировка, это связано с процессом копирования её значений, в теории, если хранить только указатели на значения, то асимптотика будет примерно одинаковой.

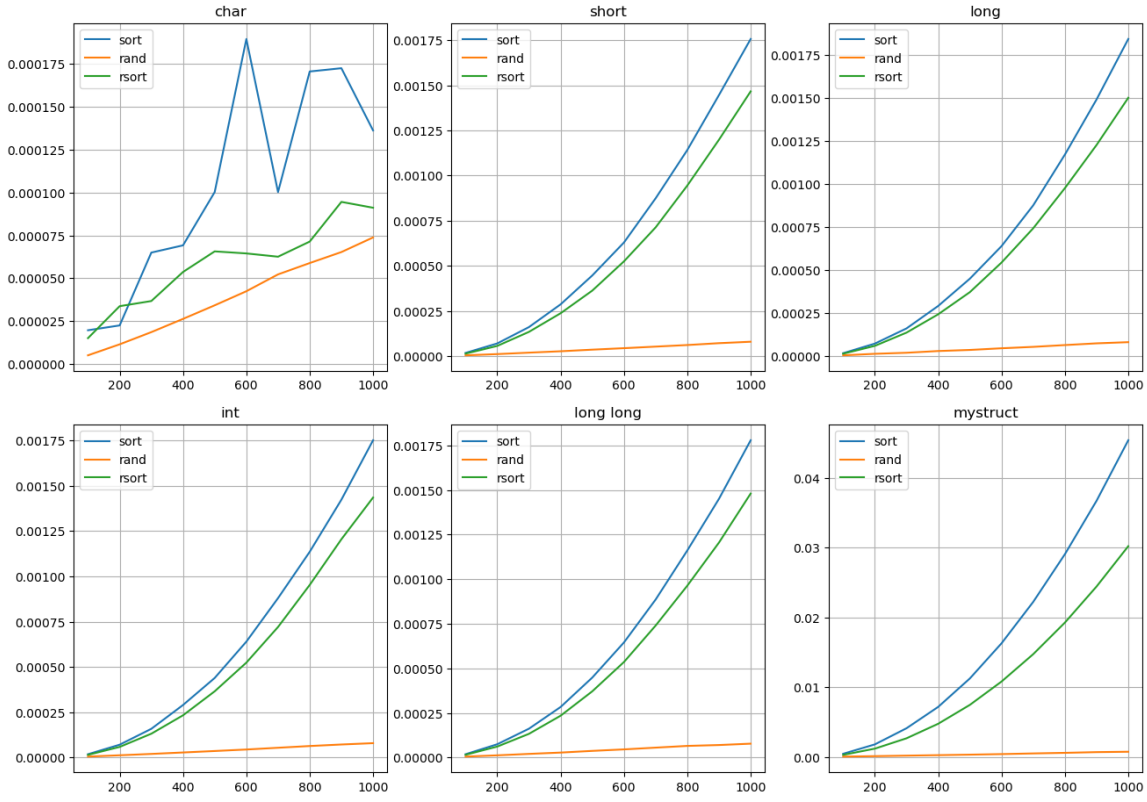


Рис. 5: Используется сортировка quick для малых размеров массива

Char имеет негладкую асимптотику, что связано со сверхнизким временем работы, несмотря на точность. В целом график асимптотики схожий, отличается только порядок времени работы.

11 Выводы

1. Python медленее C++
2. Временная сложность соответствует асимптотике почти на всем диапазоне размеров массивов
3. Оптимизация компилятора ускоряет работу при -O1, дальнейшим ускорением можно пренебречь
4. Разные сортировки имеют разную асимптотику при отсортированном, рандомным и отсортированном в обратной последовательности массивах. Кроме selection.
5. "Вес" типа элементов существенно влияет на время сортировки