# Отчет по лабораторной работе 2

Цель исследования: изучить влияние выбора концепции реализации алгоритма на его затраты по времени и по памяти.

## Объект исследования:

Было исследовано 2 алгоритма сортировки: сортировка слиянием и быстрая сортировка, каждый из которых был реализован как рекурсионный и как итерационный.

## Задачи исследования:

1) реализовать предложенные алгоритмы в виде рекурсионного и итерационного вариантов

2) Произвести отработку каждого алгоритма со списком целых чисел в качестве входных данных.

3) произвести замеры затрат памяти и времени на отработку алгоритма.

4) сделать выводы

## Ход исследования:

Исследование проводится на языке python с помощью стандартных библиотек профилирования time и tracemalloc. Код представлен по [адресу](https://replit.com/@AlieksandrAnan2/lab2-1)

<https://replit.com/@AlieksandrAnan2/lab2-1>

quickSort\_rec – классический алгоритм быстрой сортировки в рекурсионном виде

quickSort\_iter – перестройка алгоритма в итерационный вид

mergesort\_rec – алгоритм сортировки вставками в рекурсионном виде

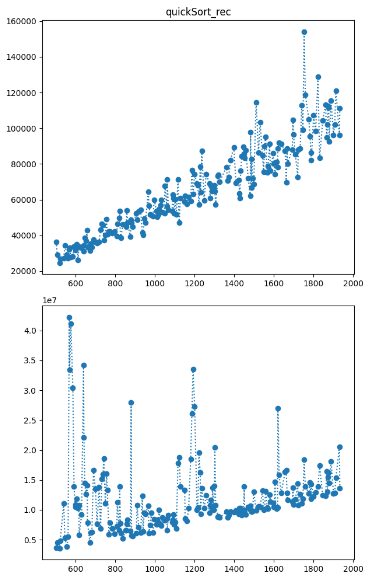
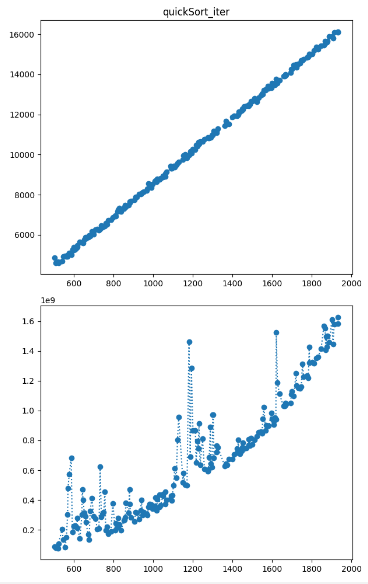
merge\_iter - алгоритм сортировки вставками в итерационном виде

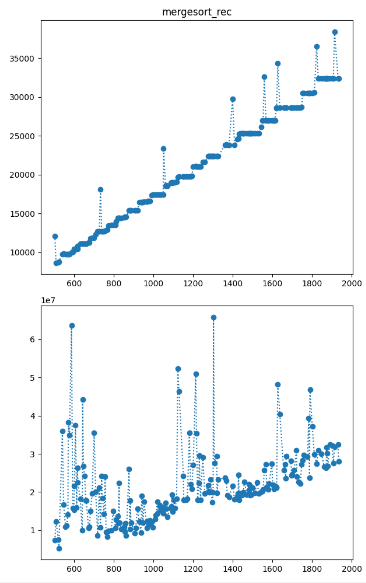
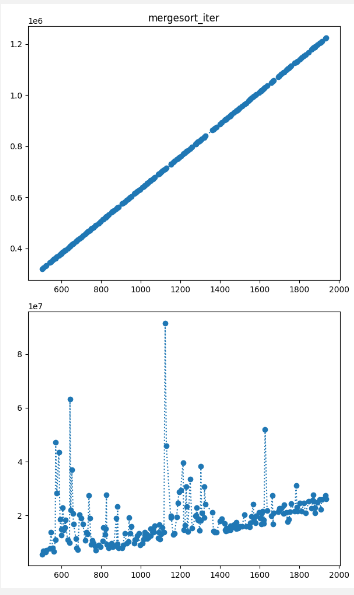
На каждой итерации исследования создается список (массив) простых чисел размером n элементов, распределенных не по порядку. Производится замер затрат памяти и времени исполнения для каждой реализации.

Всего производится 100 итераций с вычислением среднего времени и средних пиковых затрат памяти (с учетом затрат на хранение результатов исследовния).

## Результаты исследования.

При n0=500, nmax = 1900, получены следующие рузультаты:

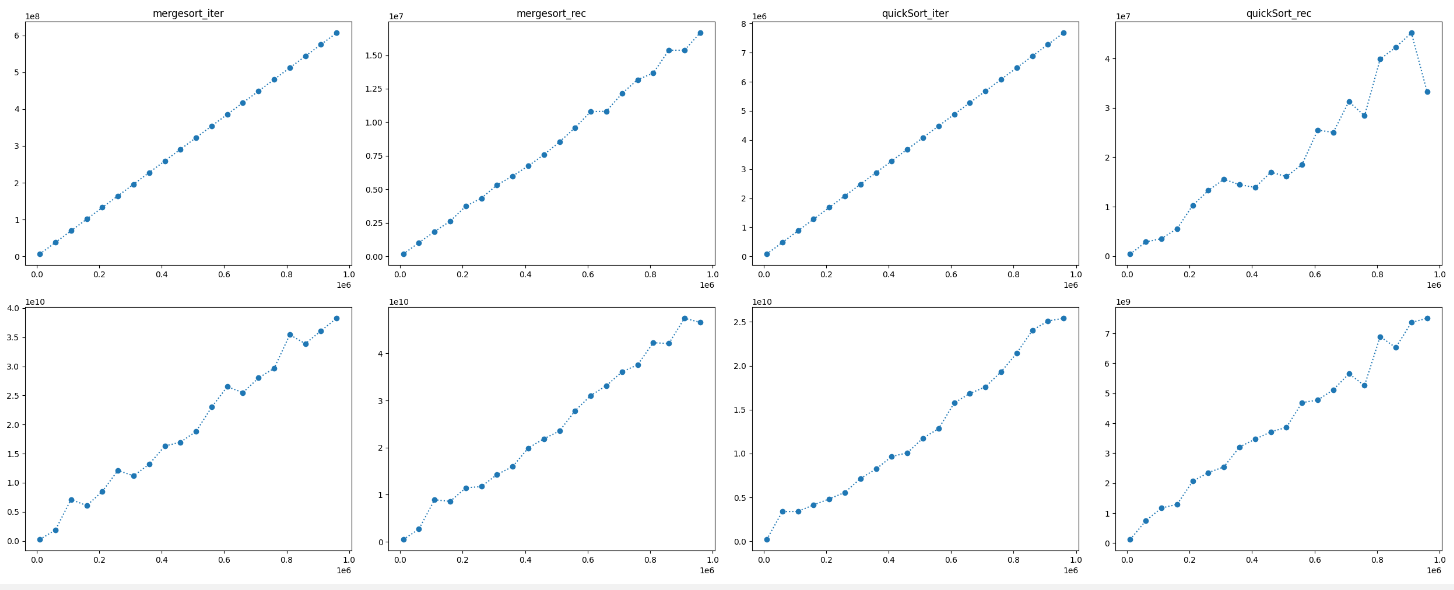
 

Видно, что по затратам времени быстрая сортировка рекурсией на два порядка опережает показатели быстрой сортировки итерацией. В то же время, итерационный вариант на порядок экономней по памяти.

Сортировка вставками с помощью рекурсии на 2 порядка выгодней по памяти, чем итерационный вариант. Достоверных различий по времени двух вариантов исполнения выявлено не было.

При n0=10000, nmax = 1000000 (шаг 10000), картина практически не меняется.



## Выводы:

Рекурсионные варианты обоих алгоритмов показали более эффективное использование машинного времени на исполнение. Алгоритм быстрой сортировки ожидаемо выгодней по используемой памяти в виде итерационного варианта. Однако, рекурсионный алгоритм сортировки вставками оказался выгодней по памяти, чем итерационный вариант. Думаю, это связано с тем, что сам алгоритм сортировки вставками изначально имеет свою «идеальную» реализацию с помощью рекурсии и его итерационная версия не эфективна.

Кроме того, итерационный алгорим быстрой сортировки оказался более эффективным по паямяти, чем встроенный оптимизированный сложный алгоритм сортировки языка (sorted()) , а рекурсионный алгоритм сортировки вставками лишь смог приблизиться по порядку к показателям встроенного алгоритма.

По скорости же ни один из представленных для исследования алгоритмов не смог приблизиться к встроенному, а рекурсионная быстрая сортировка – на целый порядок отставала от него, хоть и была лидером среди других алгоритмов.

Это связан с тем, что встроенная функция определяет наилучшее решение для осуществления сортировки в зависимости от размера списка и типа его элементов и оптимизирует скорость сортировки сложными механизмами.

