

Контрольное домашнее задание № 1

Кобызев Илья

Содержание

Введение	3
Сортировка выбором	4
1.1 Измерение времени	4
1.2 Измерение элементарных операций	5
1.3 Вывод по наблюдениям	5
Сортировка пузырьком	6
2.1 Измерение времени	6
2.2 Измерение элементарных операций	7
2.3 Вывод по наблюдениям	7
Сортировка пузырьком с первой оптимизацией Айверсона	8
3.1 Измерение времени	8
3.2 Измерение элементарных операций	9
3.3 Вывод по наблюдениям	9
Сортировка пузырьком с первой и второй оптимизациями Айверсона	10
4.1 Измерение времени	10
4.2 Измерение элементарных операций	11
4.3 Вывод по наблюдениям	11
Сортировка вставками	12
5.1 Измерение времени	12
5.2 Измерение элементарных операций	13
5.3 Вывод по наблюдениям	13
Сортировка бинарными вставками	14
6.1 Измерение времени	14
6.2 Измерение элементарных операций	15
6.3 Вывод по наблюдениям	15
Сортировка подсчетом	16
7.1 Измерение времени	16
7.2 Измерение элементарных операций	17
7.3 Вывод по наблюдениям	17
Цифровая сортировка	18
8.1 Измерение времени	18
8.2 Измерение элементарных операций	19
8.3 Вывод по наблюдениям	19
Сортировка слиянием	20
9.1 Измерение времени	20
9.2 Измерение элементарных операций	21
9.3 Вывод по наблюдениям	21
Быстрая сортировка	22
10.1 Измерение времени	22
10.2 Измерение элементарных операций	23
10.3 Вывод по наблюдениям	23
Пирамидальная сортировка	24
11.1 Измерение времени	24
11.2 Измерение элементарных операций	25
11.3 Вывод по наблюдениям	25
Сортировка Шелла с последовательностью Циура	26
12.1 Измерение времени	26
12.2 Измерение времени	27
12.3 Вывод по наблюдениям	27

Сортировка Шелла с последовательностью Шелла	28
13.1 Измерение времени	28
13.2 Измерение элементарных операций	29
13.3 Вывод по наблюдениям	29
Случайный массив	30
14.1 Время	30
14.2 Элементарные операции	31
Отсортированный массив	32
15.1 Время	32
15.2 Элементарные операции	33
Почти отсортированный массив	34
16.3 Время	34
16.4 Элементарные операции	35
Обратно отсортированный массив	36
17.5 Время	36
17.6 Элементарные операции	37
Вывод	38

Введение

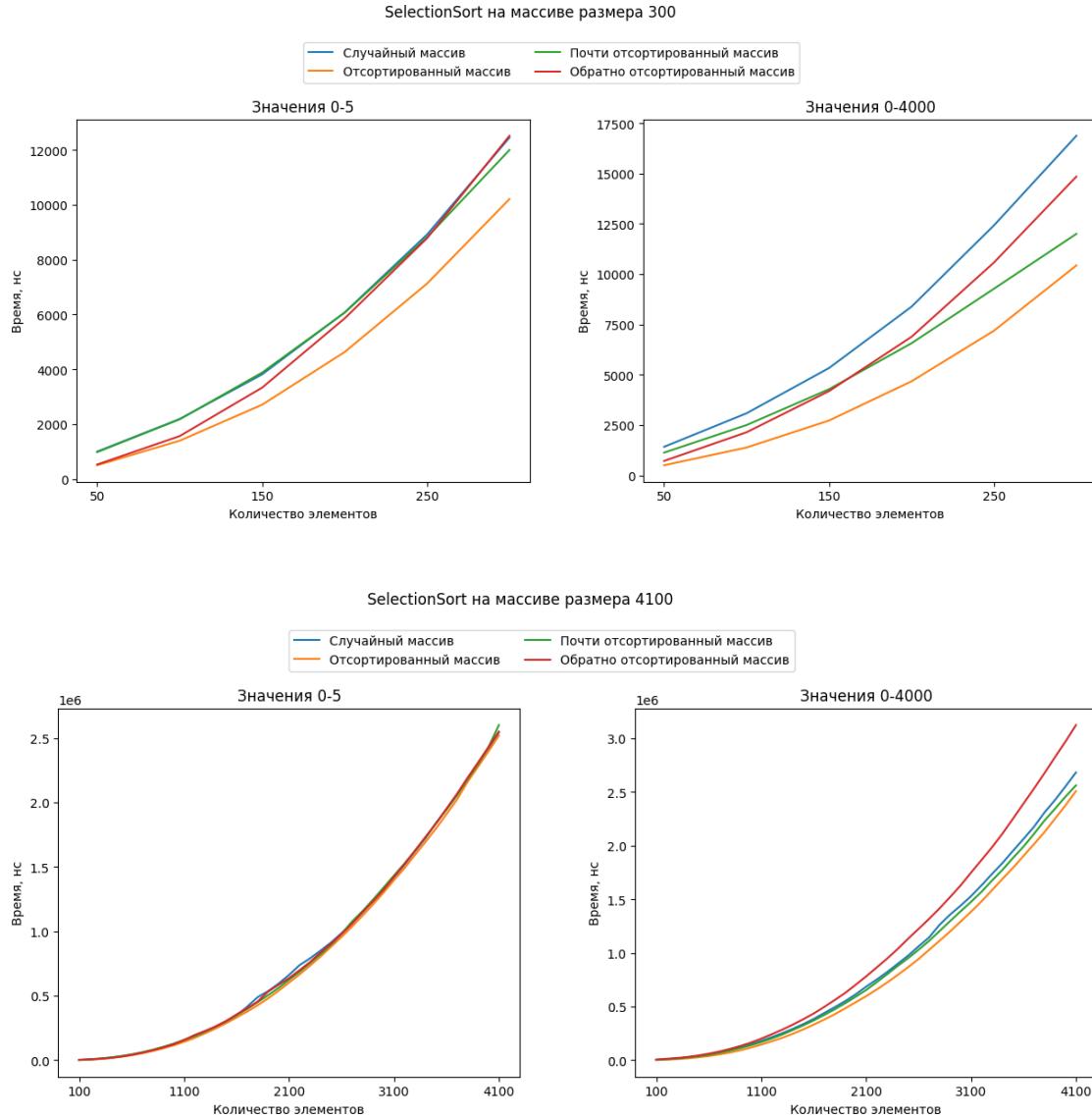
В этом отчете приведены тринадцать сортировок с кратким описанием алгоритма, а также теоретическая асимптотика. Для каждой сортировки измерено время работы и количество выполненных элементарных операций на массивах размера 300 и 4100, заполненных числами в диапазонах $[0; 5]$ и $[0; 4000]$. По этим результатам проведен анализ и сравнение с ожидаемыми данными.

Для экспериментального измерения времени работы генерировалось $k \times 16$ эталонных массивов (4 типа \times 2 размера \times 2 вида значений). Затем для каждой сортировки копировался эталонный массив, измерялось время и элементарные операции и всё это усреднялось k раз.

Сортировка выбором

Сортировка выбором — это алгоритм сортировки, который проходит по неотсортированной части массива двумя вложенными циклами и находит наименьший элемент, затем меняет его местами с первым элементом в неотсортированной части.

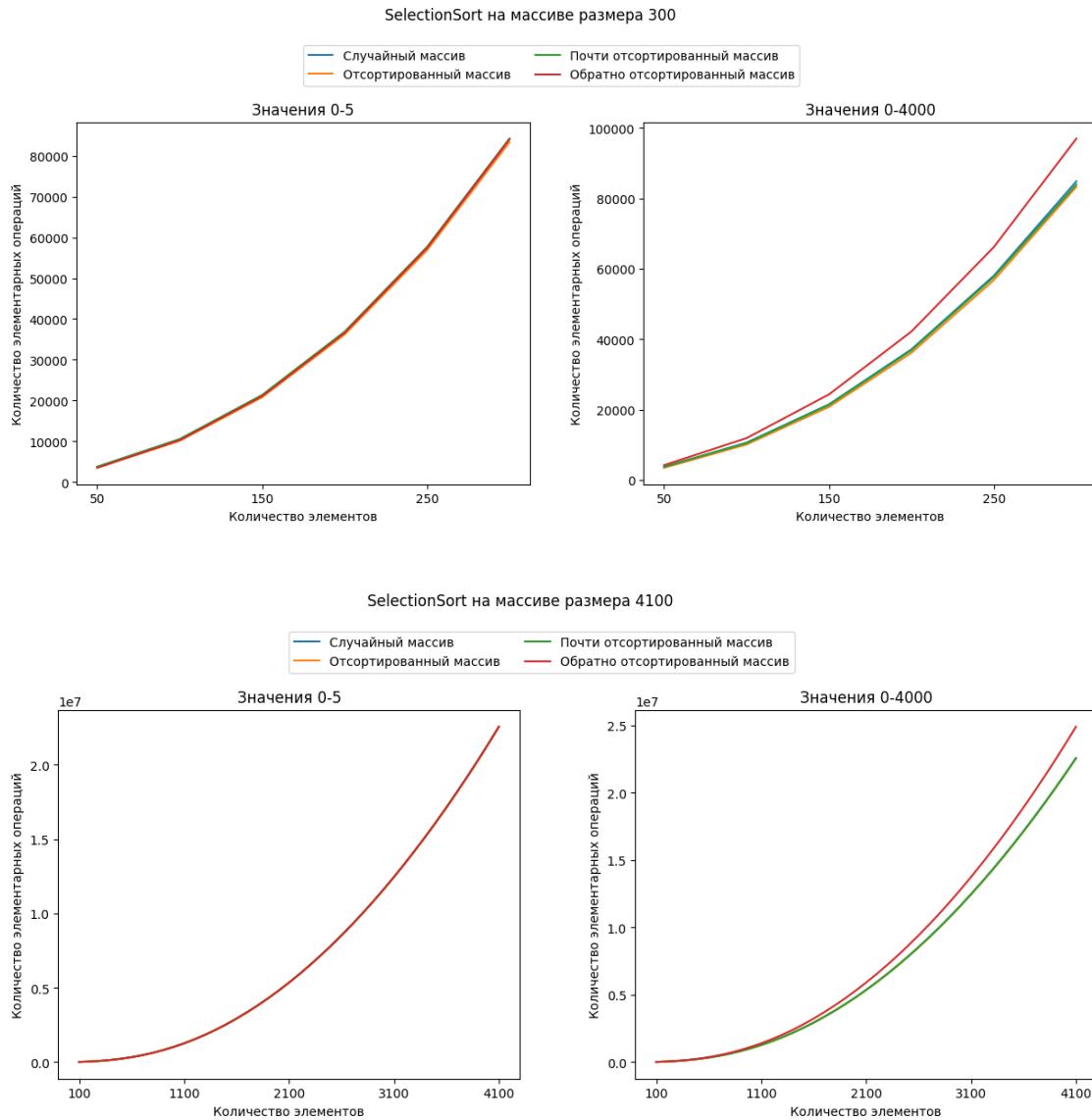
1.1 Измерение времени



Асимптотическая сложность сортировки выбором составляет:

- $O(n^2)$ в худшем случае, когда массив отсортирован в обратном порядке. Пройдет n итераций внешнего цикла и n итераций внутреннего.
- $O(n^2)$ в лучшем случае на отсортированном массиве. Всё ещё выполнится n итераций внешнего и n внутреннего циклов, но во внутреннем не будут выполнены обмены.
- $O(n^2)$ в среднем случае соответственно.

1.2 Измерение элементарных операций



Большой вклад в количество элементарных операций вносит операция обмена двух элементов, что даёт худший случай на массиве, отсортированном в обратном порядке, как и со временем. Лучший и средние случаи так же коррелируют с лучшим и средним случаями для времени.

1.3 Вывод по наблюдениям

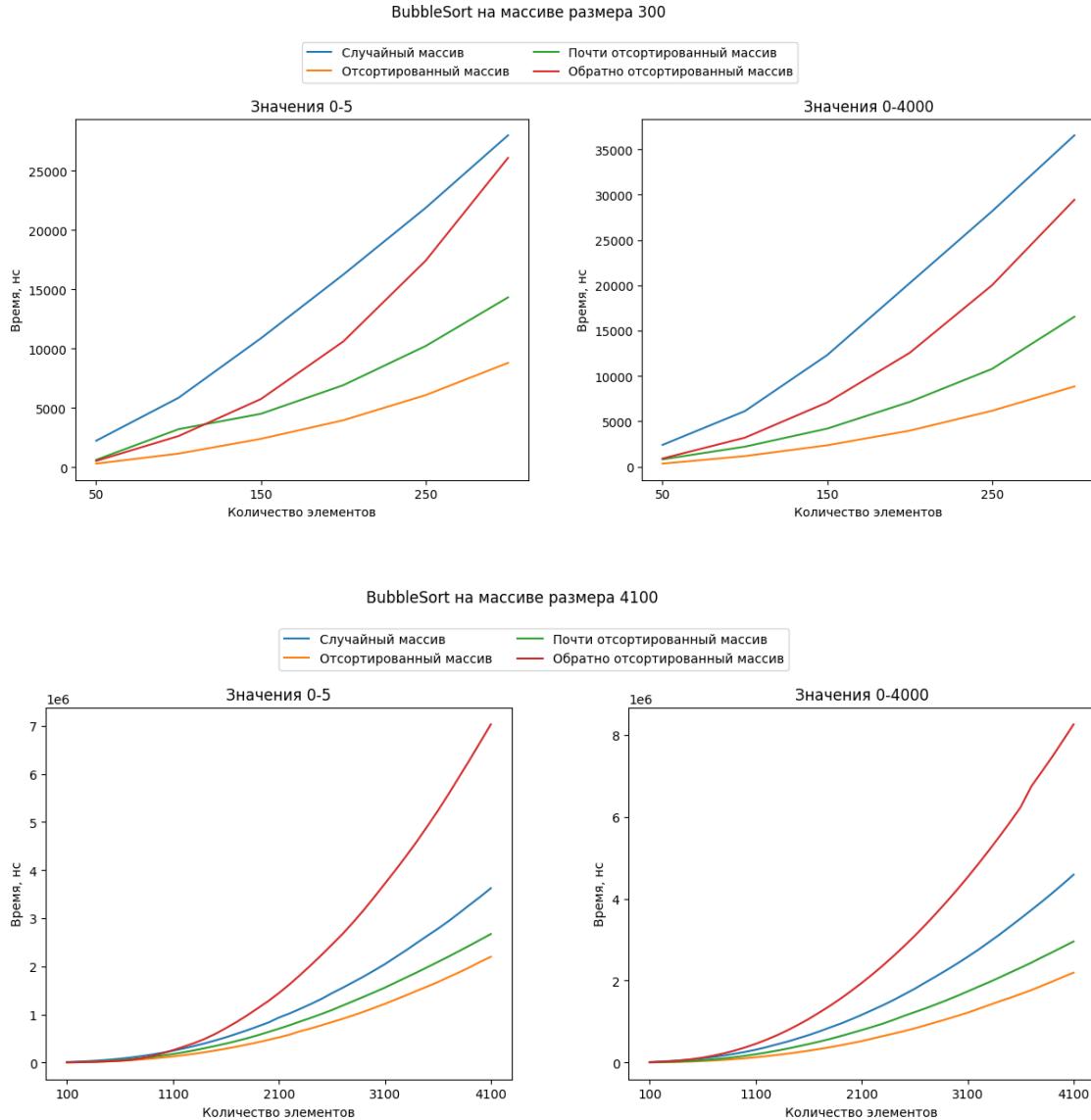
Теоретические данные подтвердились экспериментально, за исключением худшего случая для массива размера 300. Это можно объяснить тем, что количество элементов слишком мало, чтобы разница стала заметна. График элементарных операций подтверждает худший случай на обратно отсортированном массиве.

Выбросов или значительных отклонений не произошло.

Сортировка пузырьком

Сортировка пузырьком — это алгоритм сортировки, который проходит через массив несколько раз, сравнивая каждую пару соседних элементов и меняя их местами, если они находятся в неправильном порядке. На каждом проходе самый большой элемент из оставшихся неотсортированных перемещается в конец массива.

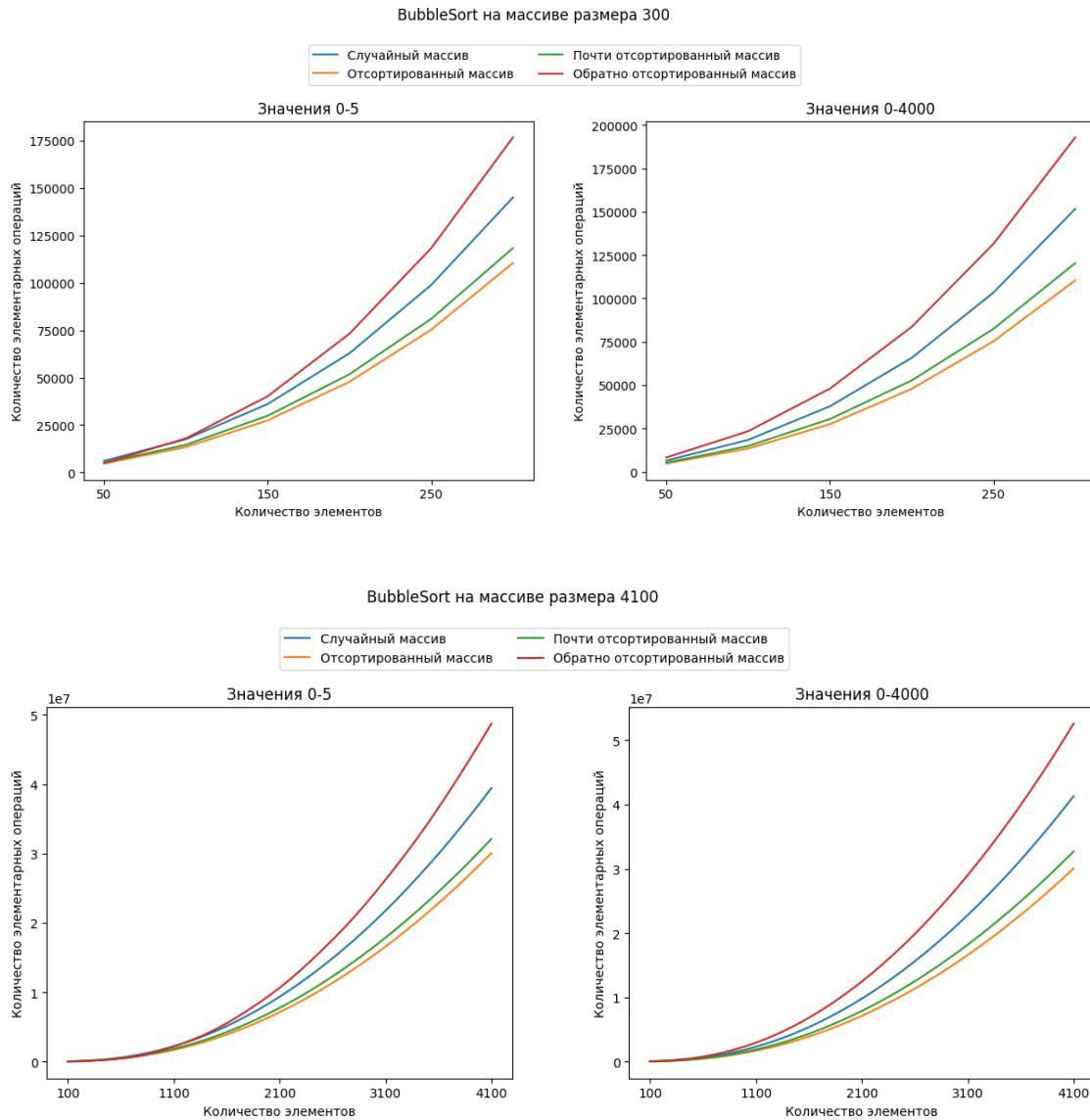
2.1 Измерение времени



Асимптотическая сложность сортировки пузырьком похожа на сложность сортировки выбором и составляет:

- $O(n^2)$ в худшем случае, когда массив отсортирован в обратном порядке. В таком случае на каждом проходе алгоритма каждый элемент сравняется со своим соседним.
- $O(n^2)$ в лучшем случае на отсортированном массиве. Всё ещё выполнится n итераций внешнего и n внутреннего циклов, но во внутреннем не будут выполнены обмены.
- $O(n^2)$ в среднем случае соответственно.

2.2 Измерение элементарных операций



Как и в случае со временем, худший случай для элементарных операций — обратно отсортированный массив, на котором произойдёт n^2 обменов, тогда как в отсортированном массиве внутренний цикл не станет менять соседние элементы.

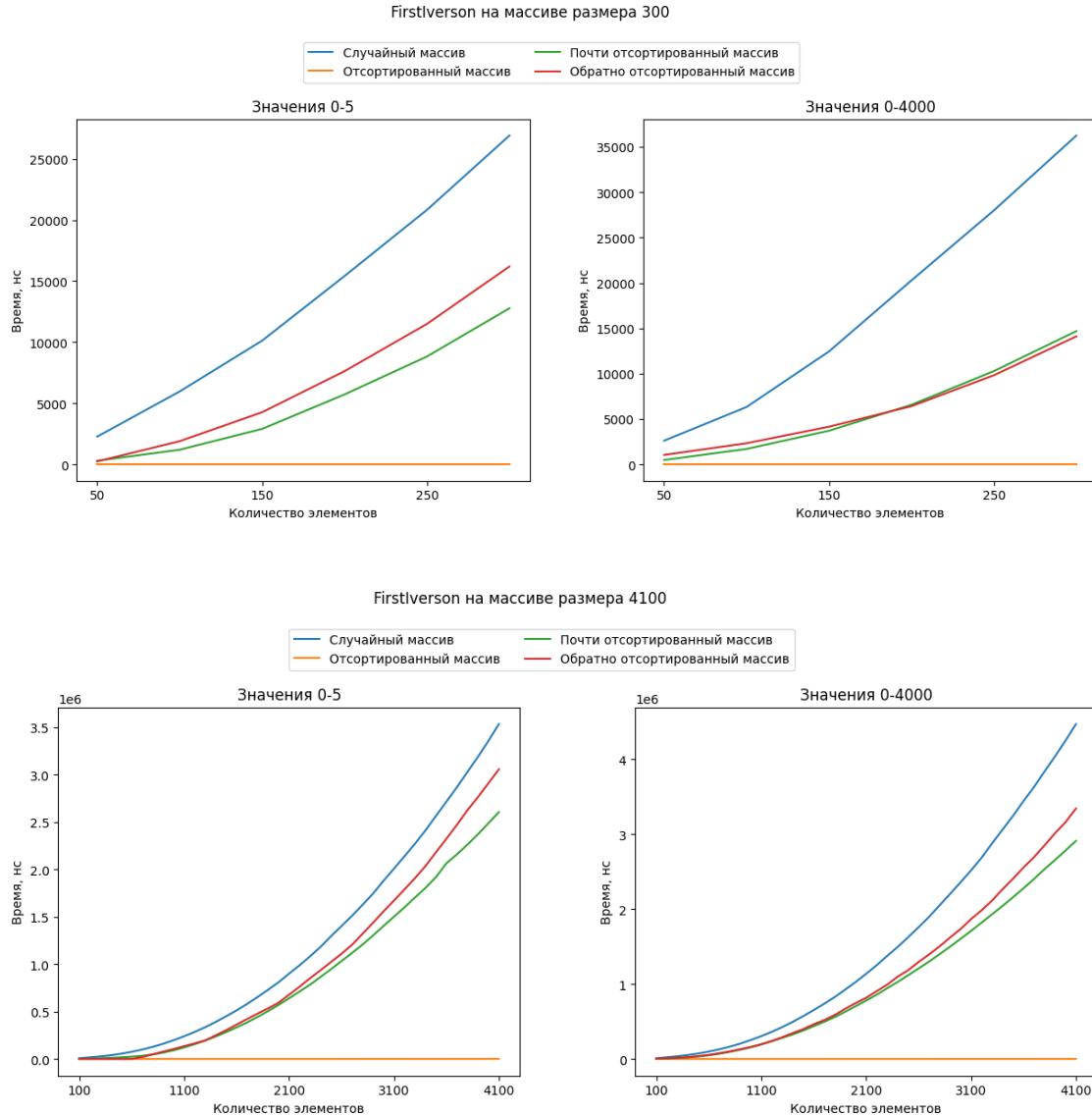
2.3 Вывод по наблюдениям

Может показаться, что на массиве размера 300 не подтверждается худший случай в виде обратной сортировки, но можно заметить, что к концу графика красная линия начинает расти быстрее. Это объясняется небольшим размером массива. Графики элементарных операций подтверждают худший случай сортировки пузырьком.

Сортировка пузырьком с первой оптимизацией Айверсона

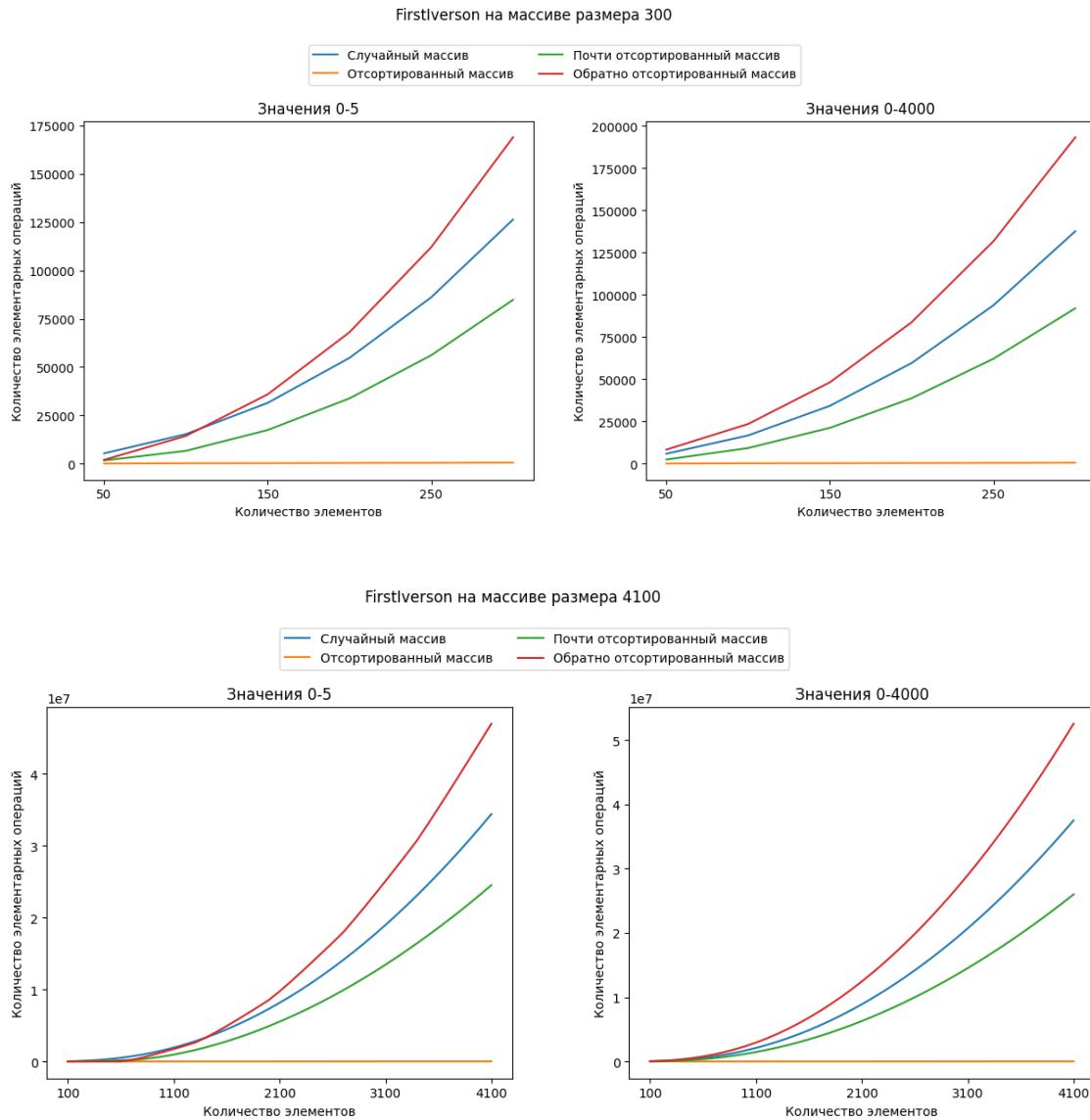
Первая оптимизация Айверсона заключается в использовании флага, который позволяет определить, отсортирован ли массив или нет. Если на какой-то итерации внутреннего цикла не было произведено ни одного обмена, значит массив уже отсортирован, и сортировка может быть завершена раньше.

3.1 Измерение времени



Асимптотика худшего и среднего случаев аналогична сортировке пузырьком, но в лучшем случае на отсортированном массиве она составит $O(n)$

3.2 Измерение элементарных операций



Количество элементарных операций пропорционально времени выполнения, лучший случай также $O(n)$ на отсортированном массиве.

3.3 Вывод по наблюдениям

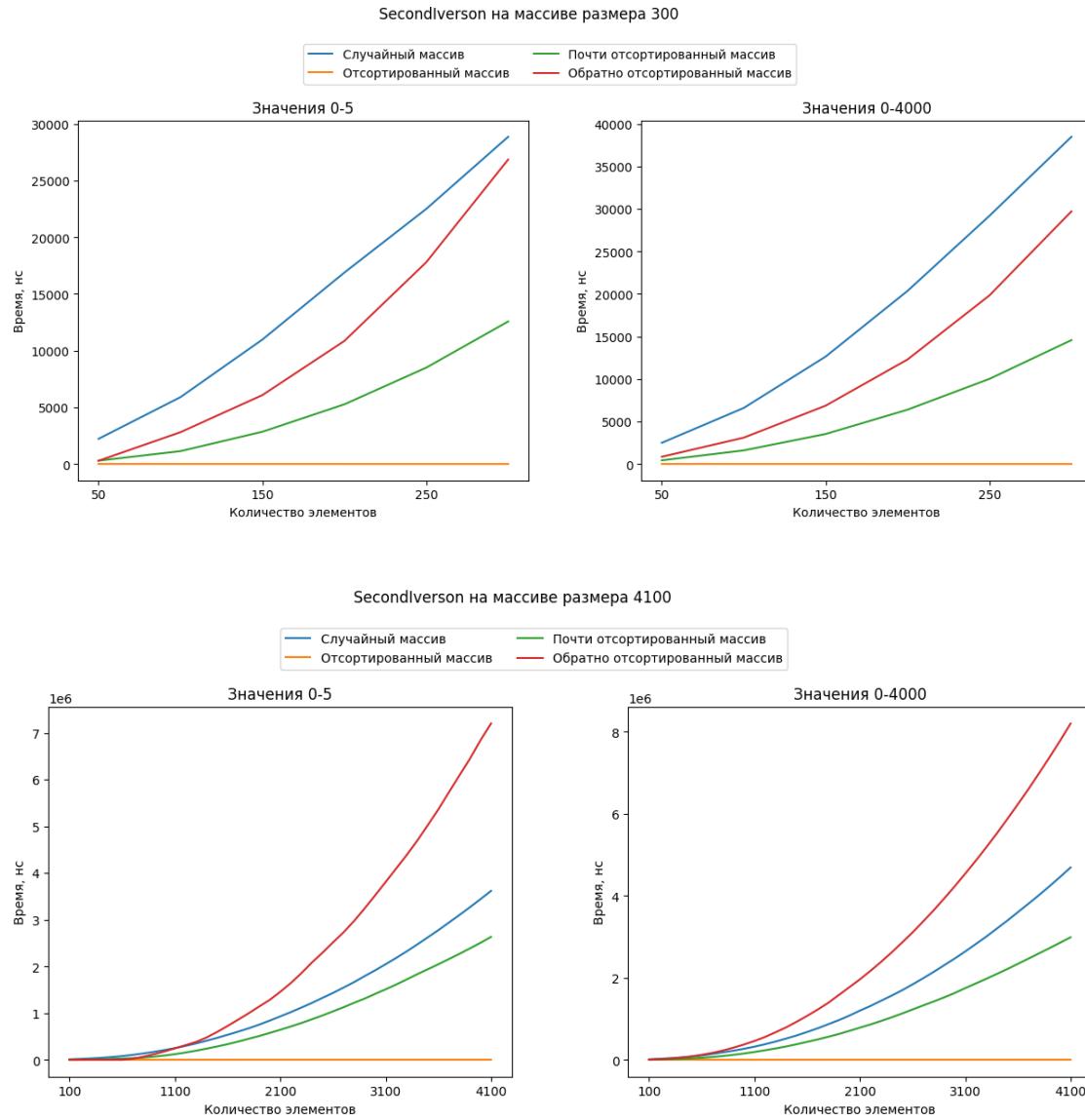
Теоретические данные частично подтвердились. На графиках обратно отсортированный массив не является худшим случаем. Это может быть связано с оптимизациями компилятора или недостаточным размером данных.

Выбросов или значительных отклонений не обнаружено. Из-за масштаба графика отсортированный массив может выглядеть как $O(1)$, однако это $O(n)$ в приближении.

Сортировка пузырьком с первой и второй оптимизациями Айверсона

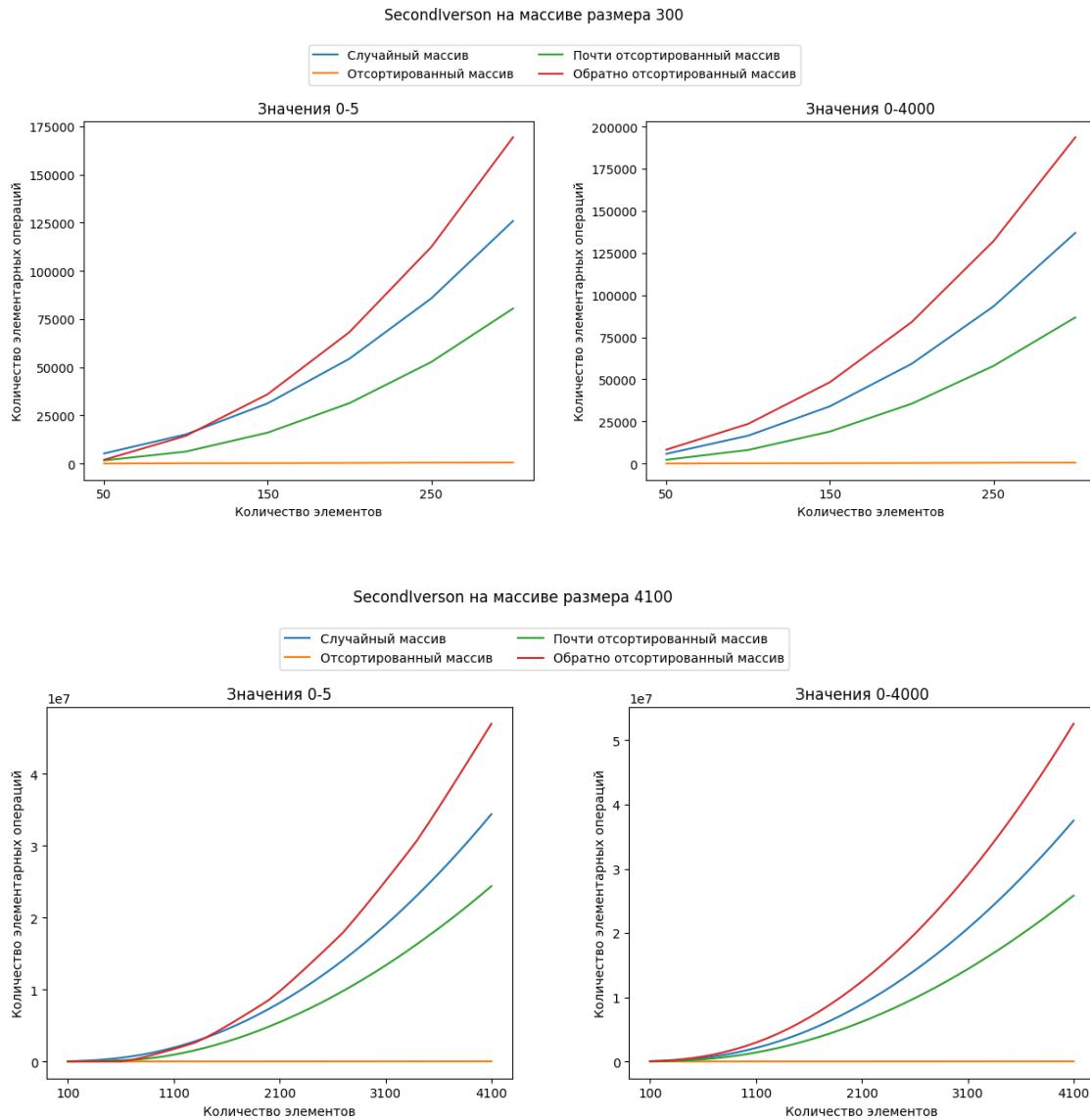
Вторая оптимизация Айверсона заключается в том, чтобы запоминать индекс последнего обмена на каждой итерации внешнего цикла и использовать его в качестве границы для следующей итерации.

4.1 Измерение времени



Асимптотика аналогична первой оптимизации Айверсона, но количество обменов снижено, что улучшает константу квадрата.

4.2 Измерение элементарных операций



За счет меньшего количества сравнений и обменов, количество элементарных операций должно быть меньшим, чем в первой оптимизации Айверсона.

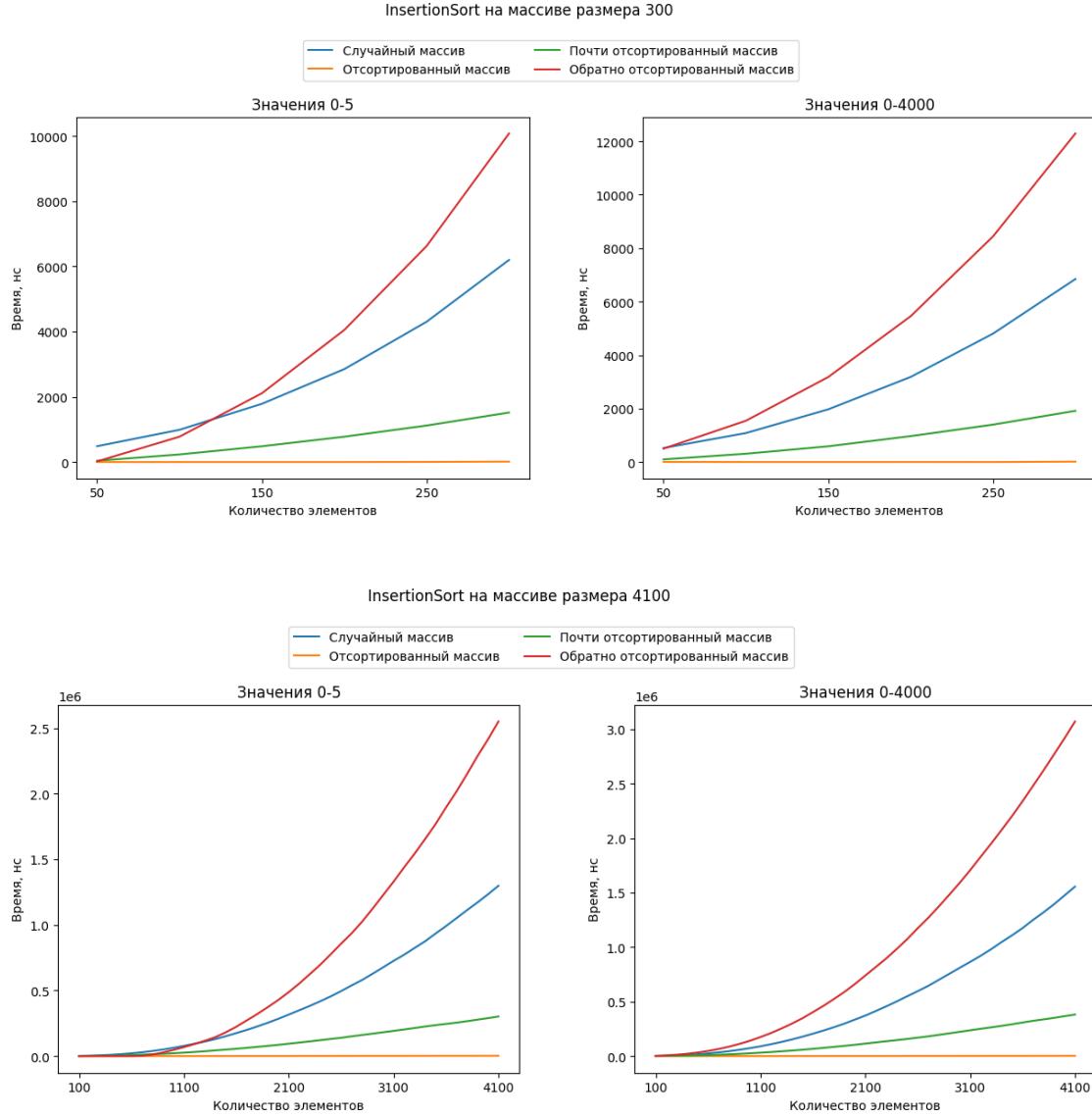
4.3 Вывод по наблюдениям

Лучший и худший случаи подтвердились экспериментально аналогично первому Айверсону. Однако данный алгоритм работает дольше, чем первый Айверсон, хотя в теории должно быть наоборот. Это может быть связано с различными реализациями этих алгоритмов.

Сортировка вставками

Сортировка вставками — это алгоритм сортировки, который проходит по массиву и на каждой итерации берет очередной элемент и вставляет его в отсортированную часть массива.

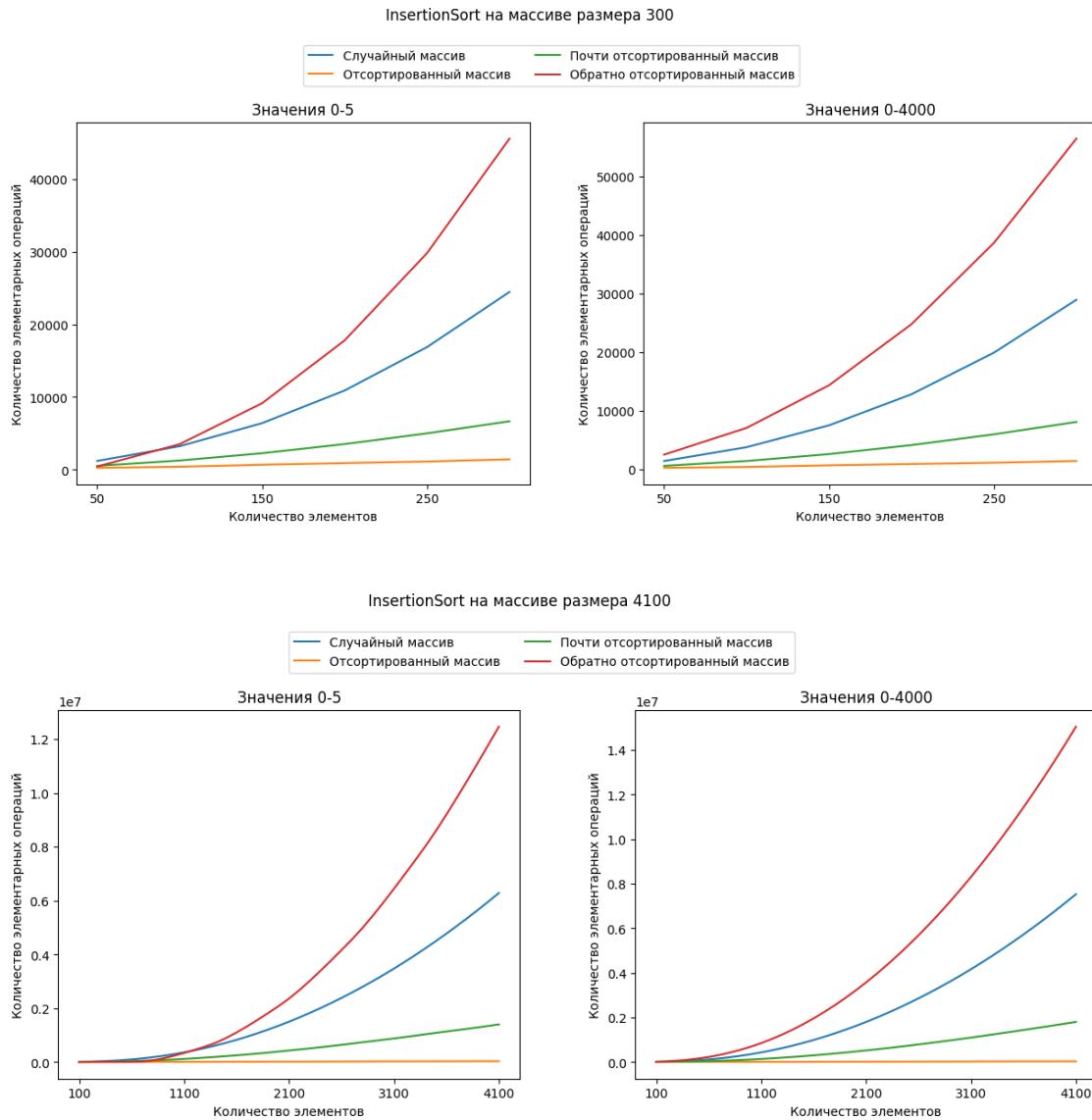
5.1 Измерение времени



Асимптотическая сложность сортировки вставками составляет:

- $O(n^2)$ в худшем случае, когда массив отсортирован в обратном порядке. В таком случае, каждый новый элемент нужно вставить в начало массива, а для этого нужно выполнить максимальное количество операций перестановки элементов.
- $O(n)$ в лучшем случае на отсортированном массиве. В этом случае каждый элемент массива будет сравниваться только с предыдущим элементом и ни один элемент не будет перемещен в процессе сортировки.
- $O(n^2)$ в среднем случае соответственно.

5.2 Измерение элементарных операций



Худший, лучший и средний случаи для элементарных операций аналогичен таковым для времени.

5.3 Вывод по наблюдениям

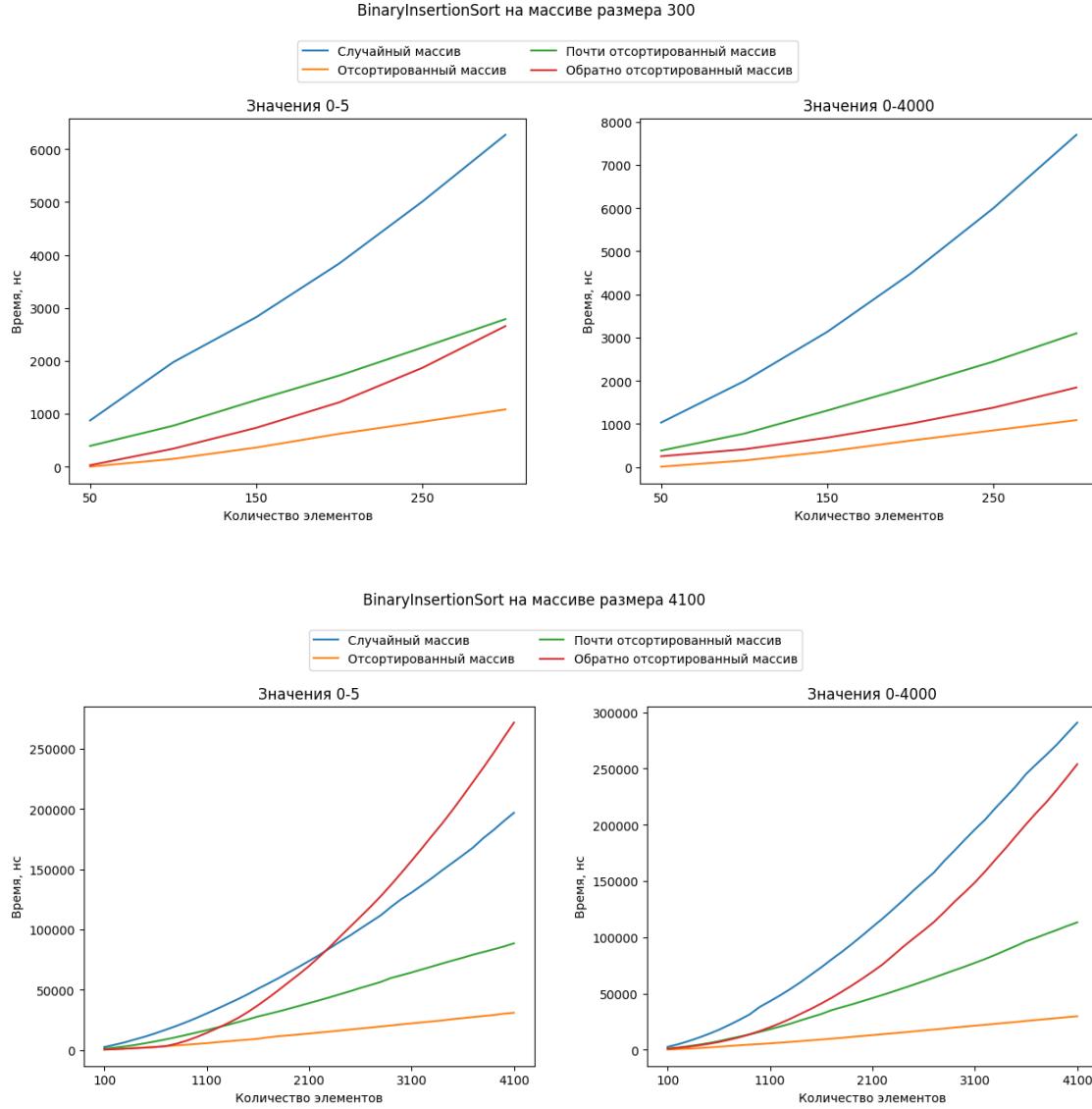
Теоретические данные подтвердились экспериментально, значительных отклонений или выбросов не обнаружено.

На графике массива размера 300 со значениями $[0; 5]$ случайный массив сначала имеет большее, время сортировки, чем обратно отсортированный, однако затем они меняются местами и у обратно отсортированного массива заметна большая константа квадрата. Это произошло из-за небольшого размера массива и диапазона значений.

Сортировка бинарными вставками

Сортировка бинарными вставками является модификацией сортировки вставками, в которой поиск правильной позиции для вставки нового элемента осуществляется бинарным поиском в отсортированной части массива.

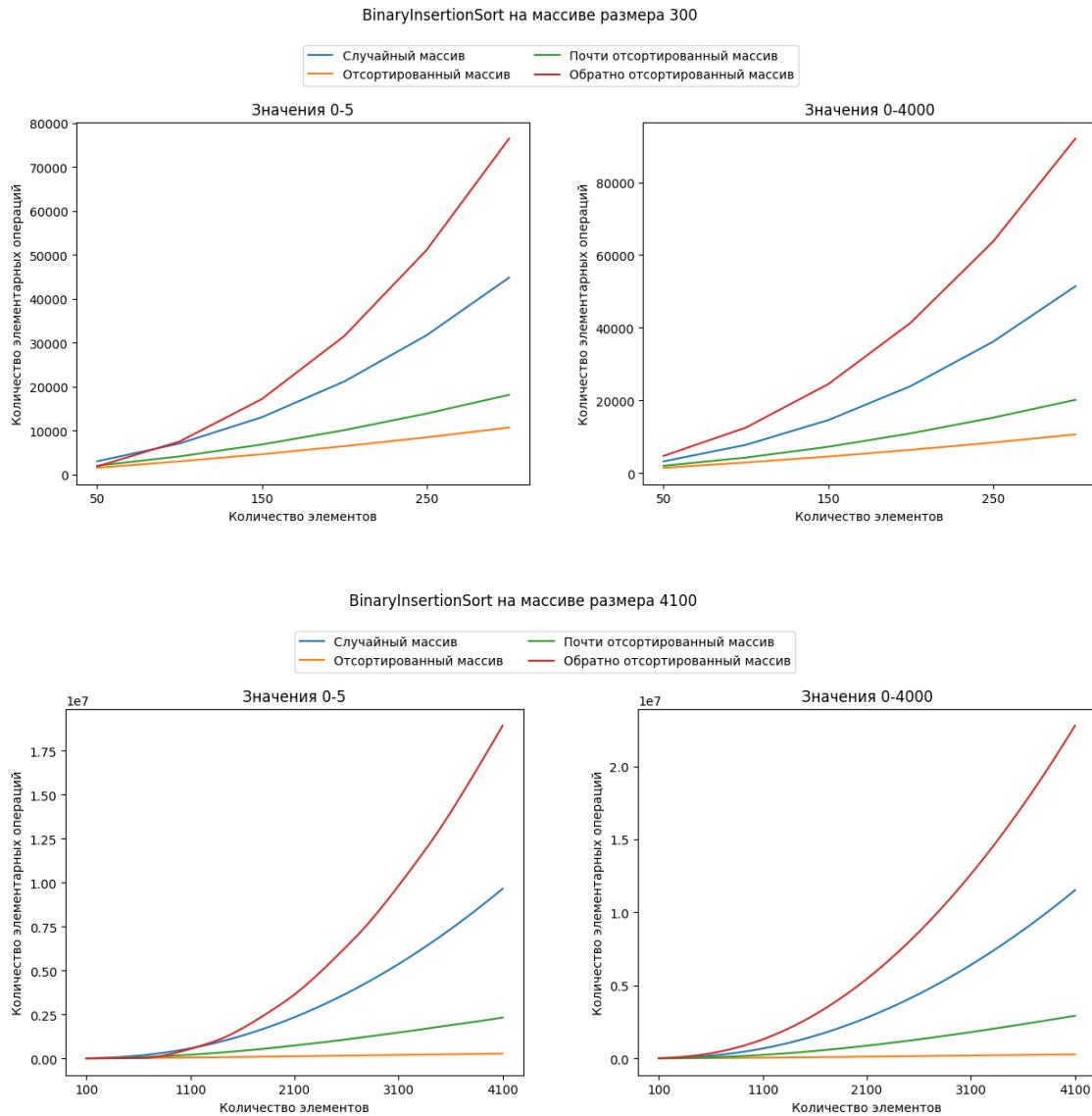
6.1 Измерение времени



Асимптотическая сложность сортировки бинарными вставками составляет:

- $O(n^2)$ в худшем случае, когда массив отсортирован в обратном порядке. Бинарный поиск выполняется на каждой итерации для всех элементов отсортированной части массива.
- $O(n)$ в лучшем случае на отсортированном массиве. Бинарный поиск в отсортированной части массива не выполняется, только один проход по массиву.
- $O(n^2)$ в среднем случае соответственно.

6.2 Измерение элементарных операций



Количество элементарных операций должно быть пропорционально времени выполнения.

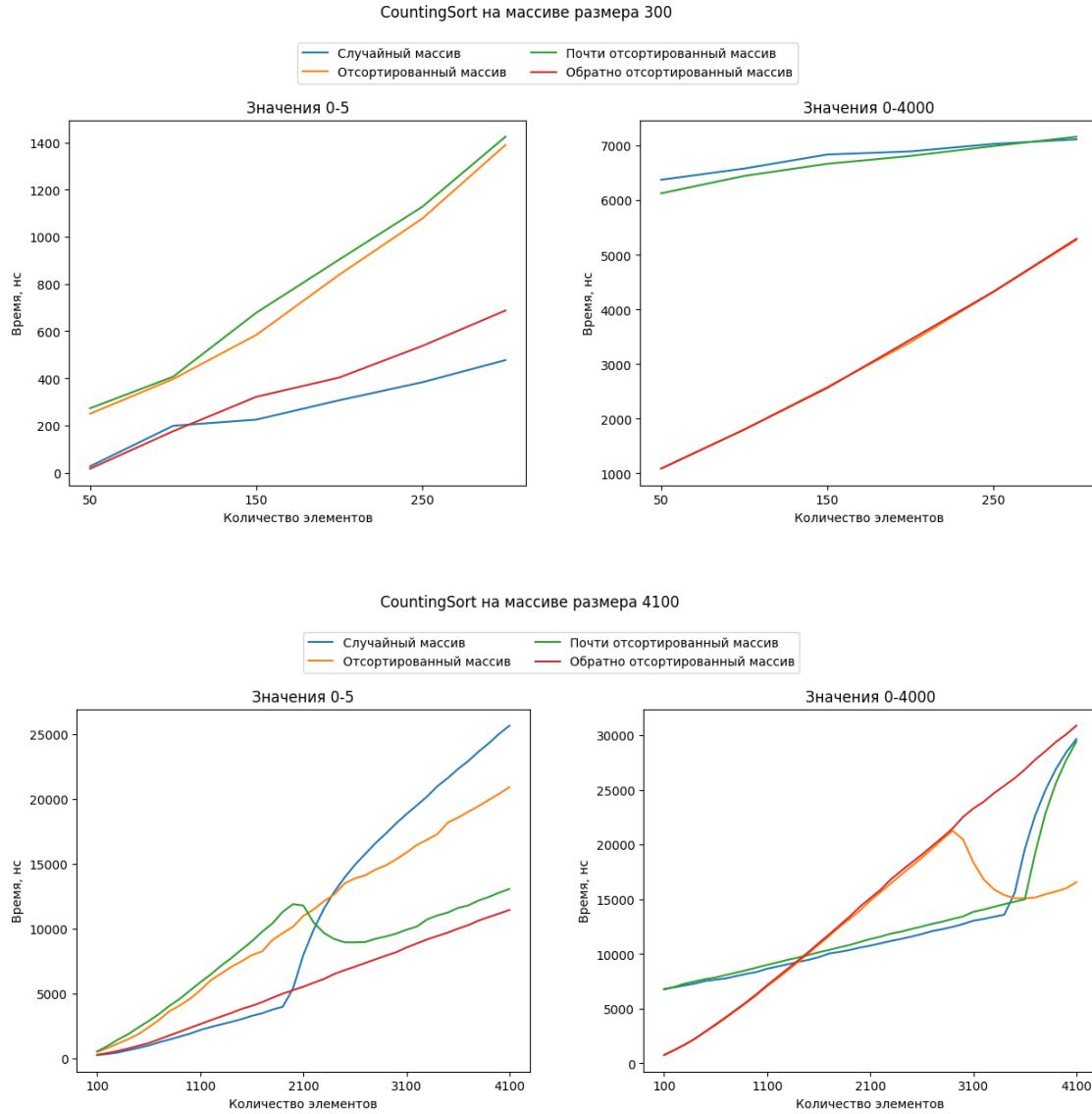
6.3 Вывод по наблюдениям

Теоретические данные о лучшем случае для отсортированного массива подтвердились на всех графиках, а худшем случае для обратно отсортированного только для массивов размера 4100. Возможно размера массива в 300 элементов недостаточно, из-за большой константы. Графики элементарных операций подтверждают худший случай для обратно отсортированного массива.

Сортировка подсчетом

Сортировка подсчетом использует массив для хранения количества вхождений каждого элемента во входном массиве. По количеству вхождений можно построить отсортированный массив.

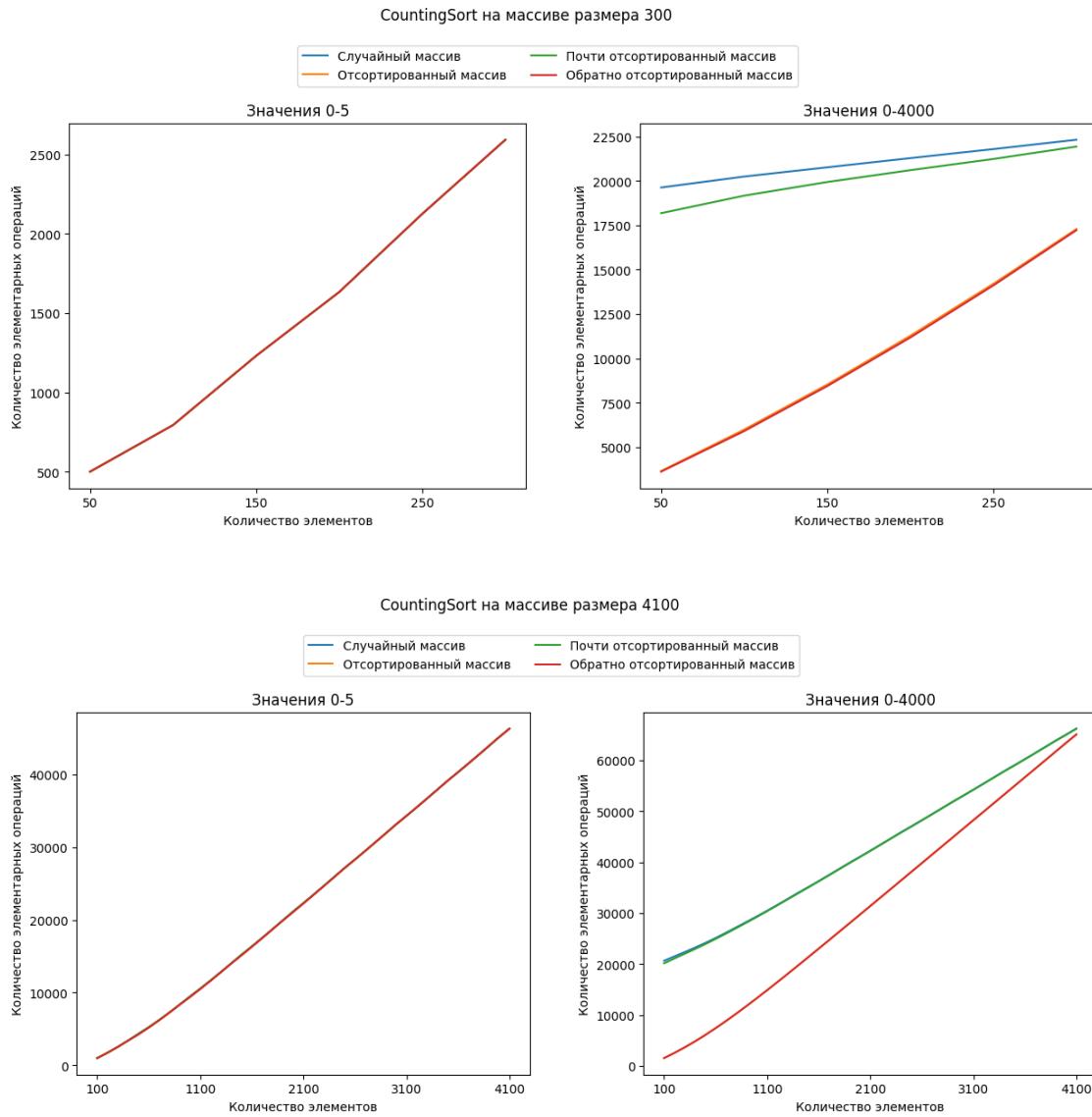
7.1 Измерение времени



Асимптотическая сложность сортировки подсчетом составляет $O(n + k)$, однаков возможны различные варианты:

- В худшем случае при большом разбросе значений k также будет большим. итерации для всех элементов отсортированной части массива.
- В лучшем случае при небольшом диапазоне значений $O(n)$

7.2 Измерение элементарных операций



Количество элементарных операций пропорционально времени выполнения.

7.3 Вывод по наблюдениям

Асимптотика $O(n + k)$ подтвердилась экспериментально - угол наклона у графиков со значениями $[0; 5]$ и $[0; 4000]$ отличается. На правом графике времени для массива размера 300 можно заметить большую константу у случайного и почти отсортированном массивах. То же самое можно заметить в начале графика массива на 4100 элементов. Так как по мере увеличения числа элементов разница нивелируется, можно предположить, что это происходит из-за оптимизаций компилятора, связанных с отсортированными массивами (поиск минимума и максимума например).

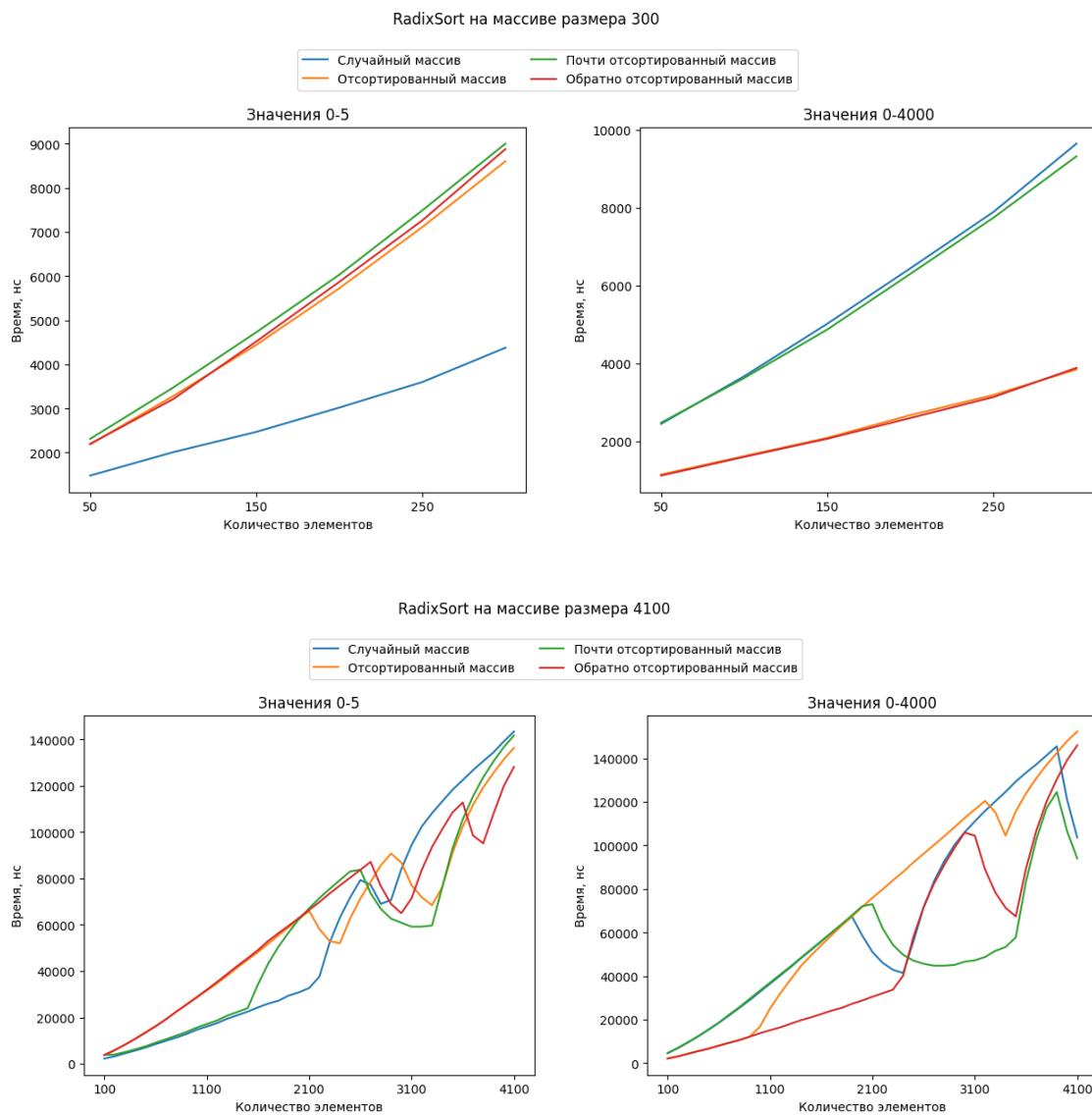
На графиках массивов размера 4100 видны скачки. Я считаю, что из-за маленького масштаба оси у так на графике отражаются фоновые процессы компьютера.

Цифровая сортировка

Цифровая сортировка — это алгоритм сортировки, который использует поразрядную сортировку на каждом разряде чисел. Алгоритм работает с целыми числами и сортирует их путем разбиения чисел на цифры и последовательной сортировки по каждой цифре.

В данном случае используется LSD по основанию 16.

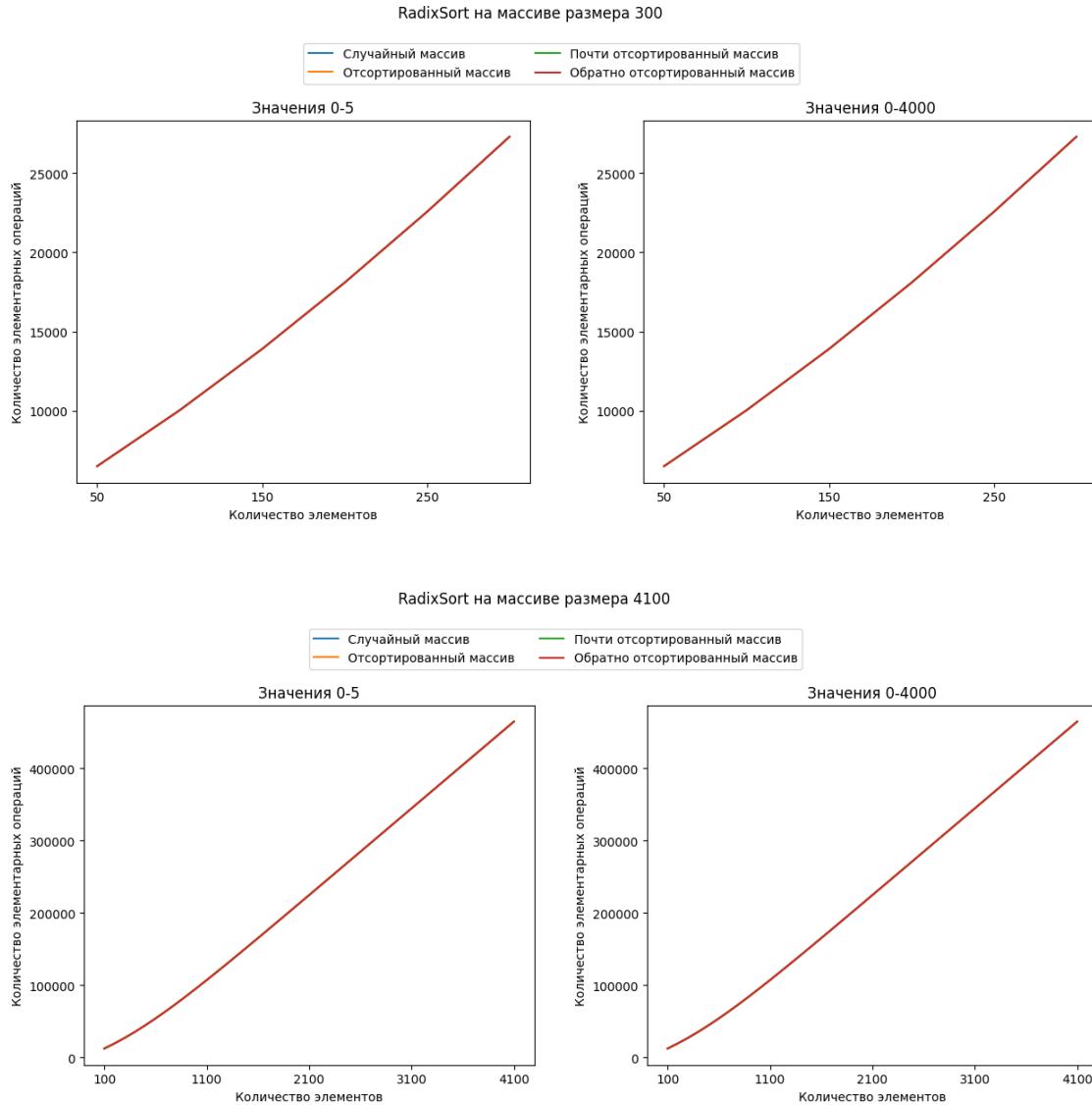
8.1 Измерение времени



Алгоритм имеет асимптотическую сложность $O(d \times (n + k))$, где d - максимальное количество цифр в числе, n - количество элементов в массиве, k - основание системы счисления.

В данном случае это $O(1 \times (n + 16))$ и $O(4 \times (n + 16))$.

8.2 Измерение элементарных операций



Количество элементарных операций стабильно пропорционально $O(d \times (n + k))$, поскольку сортируется каждая цифра независимо от типа массива.

8.3 Вывод по наблюдениям

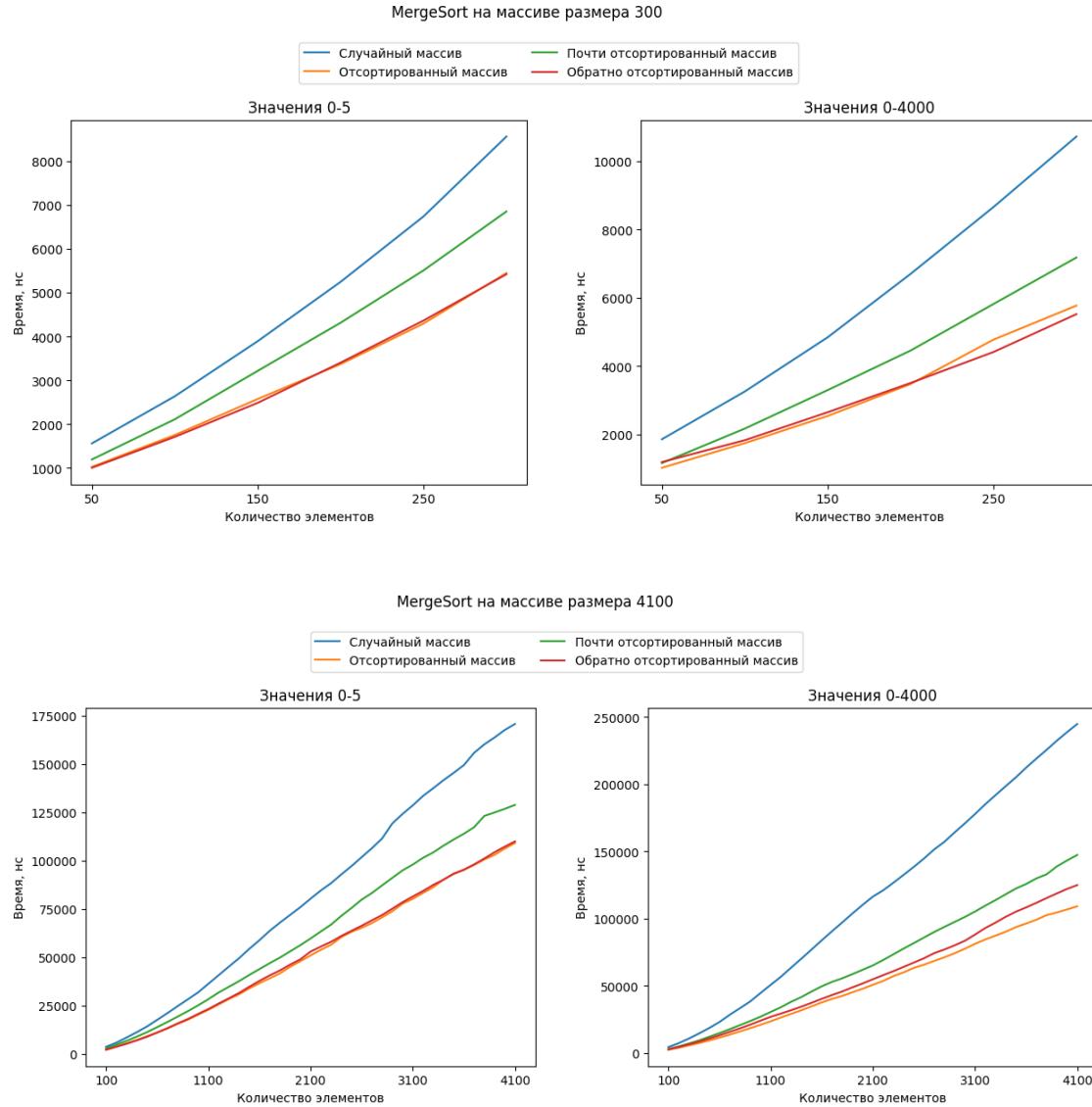
На графиках с большим диапазоном значений можно заметить больший наклон линии, что подтверждает зависимость времени от количества разрядов числа.

Количество элементарных операций не меняется, так как сравнивается каждая для всех чисел. Скачки на графиках имеют ту же природу, что и скачки в сортировке подсчетом, так как она используется для сортировки цифр в разрядах.

Сортировка слиянием

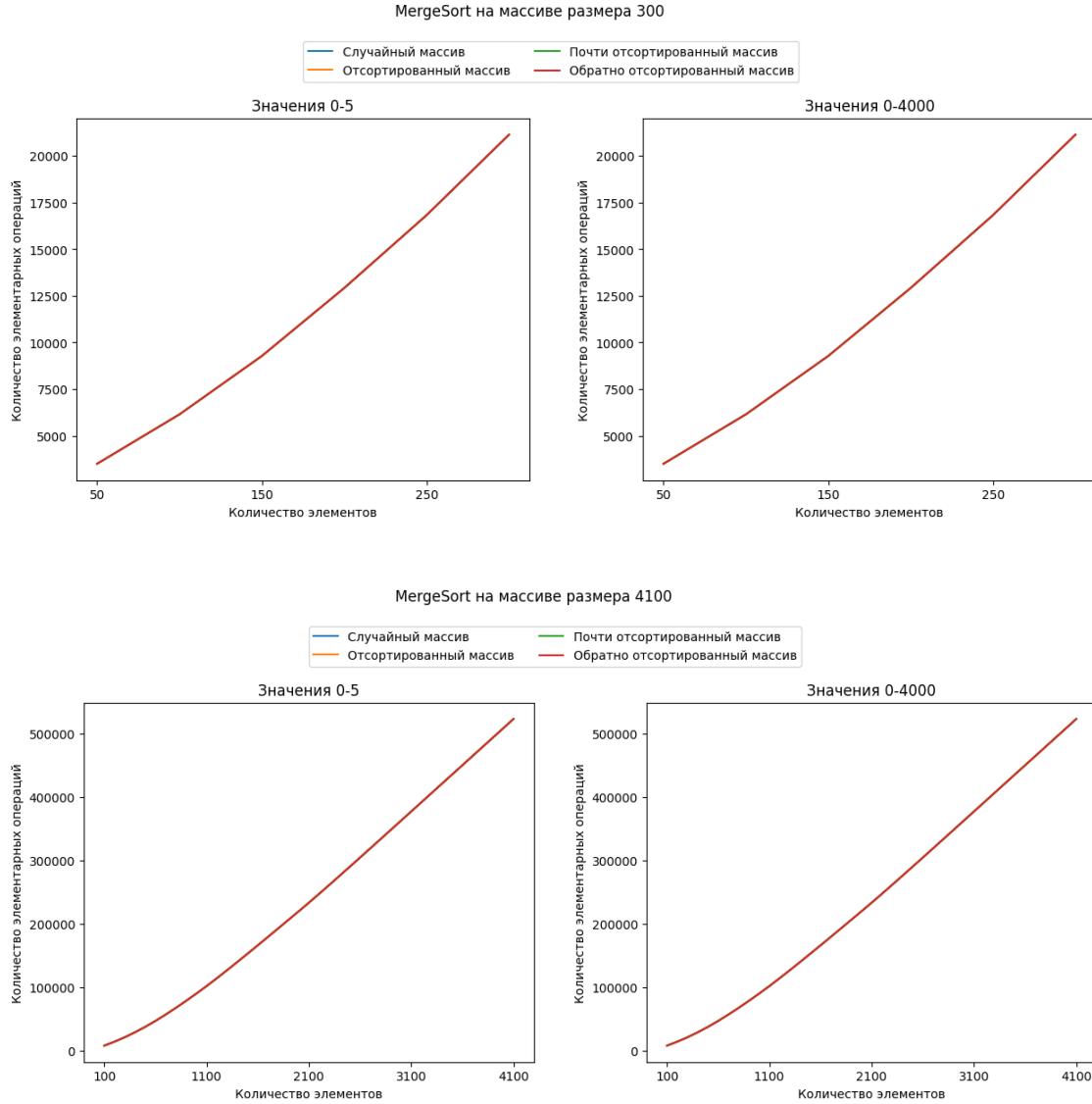
Сортировка слиянием использует подход «разделяй и властвуй». Он разделяет массив на меньшие подмассивы, рекурсивно сортирует их, а затем объединяет их в отсортированный массив.

9.1 Измерение времени



Асимптотическая сложность для худшего, лучшего и среднего случаев одинакова и составляет $O(n \log n)$, где n — количество элементов в массиве. На каждом уровне рекурсии массив разбивается на две половины, что занимает $O(\log n)$ операций, а затем происходит слияние двух отсортированных подмассивов, что занимает $O(n)$ операций.

9.2 Измерение элементарных операций



Элементарные операции, как и время, стабильно пропорциональны $O(n \log n)$.

9.3 Вывод по наблюдениям

Несмотря на то, что теоретически сортировка слиянием не должна зависеть от типа массива, случайный массив выделяется как худший случай, а почти отсортированный имеет большее время, чем отсортированный и обратно отсортированный.

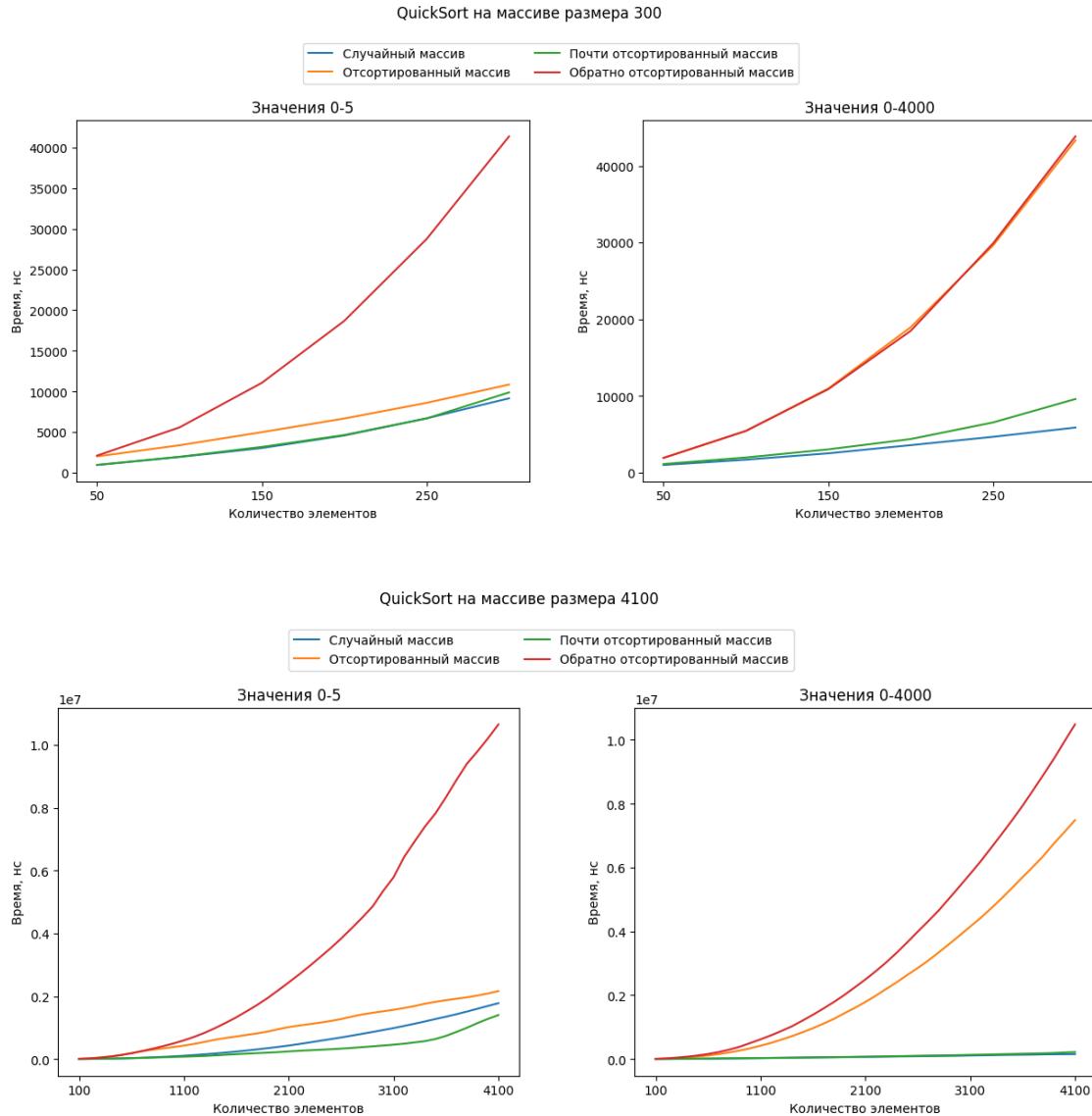
Это может быть связано с тем, что в отсортированном массиве элементы уже упорядочены, и на каждом шаге слияния нужно сравнить только первые элементы каждого подмассива, что занимает меньше времени, чем при слиянии двух случайных подмассивов. При обратно отсортированном массиве, на каждом шаге слияния элементы с конца массива, которые уже отсортированы, сливаются с элементами, которые ближе к началу массива и которые нужно переставлять. Это также может ускорить работу алгоритма, поскольку уже отсортированные элементы могут быть проигнорированы в процессе слияния.

Быстрая сортировка

Быстрая сортировка разбивает массив на две подгруппы, меньшую и большую, относительно выбранного элемента (опорного). Затем каждая из подгрупп сортируется рекурсивно, и объединяется в один отсортированный массив.

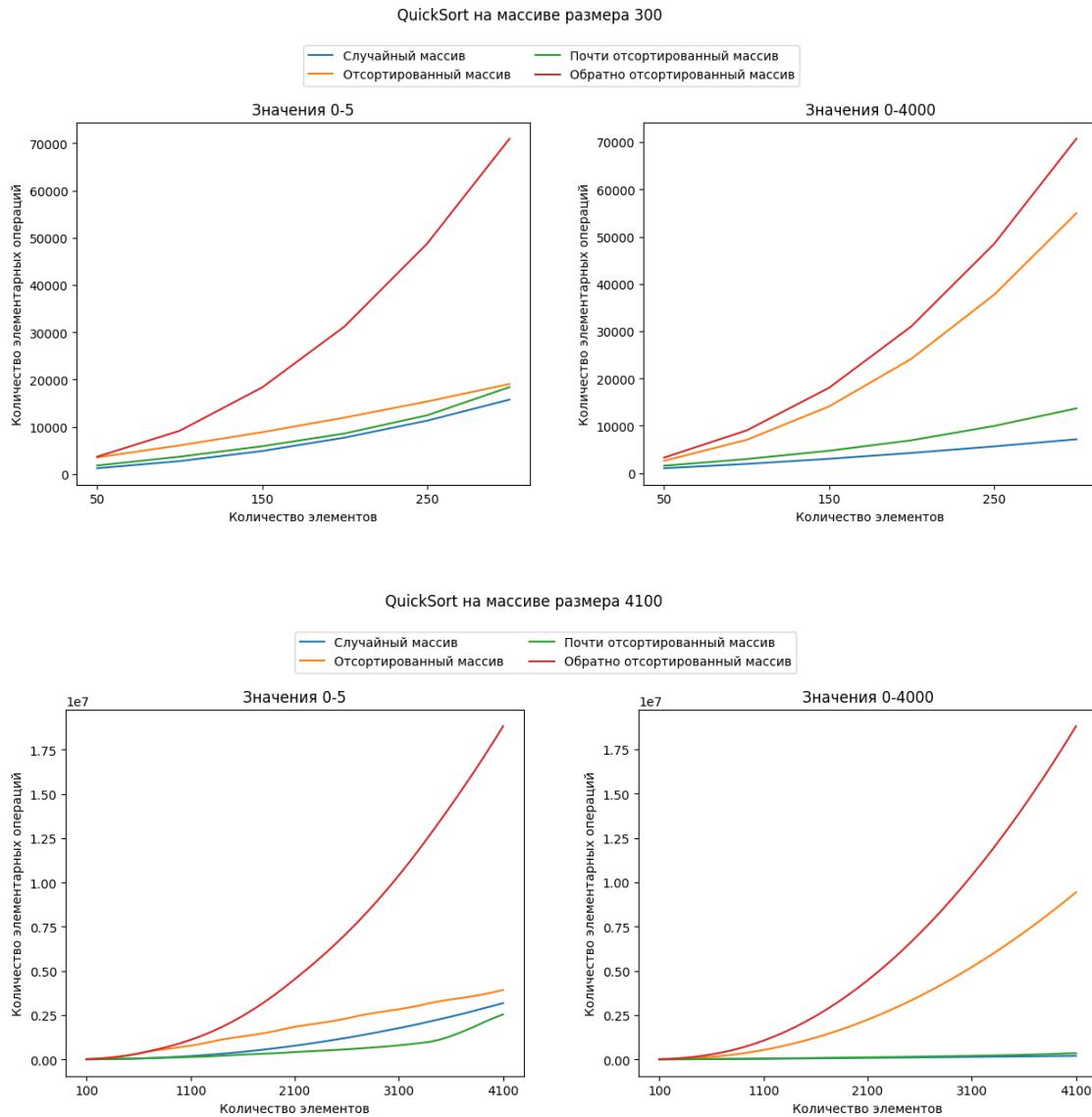
В данном случае в качестве опорного элемента выбирается первый элемент.

10.1 Измерение времени



Асимптотическая сложность алгоритма быстрой сортировки в лучшем и среднем случае составляет $O(n \log n)$, где n — размер сортируемого массива. В наихудшем случае, когда опорный элемент выбирается неудачно и несколько раз попадает на наибольший или наименьший элемент, сложность алгоритма составляет $O(n^2)$.

10.2 Измерение элементарных операций



Количество элементарных операций пропорционально времени выполнения.

10.3 Вывод по наблюдениям

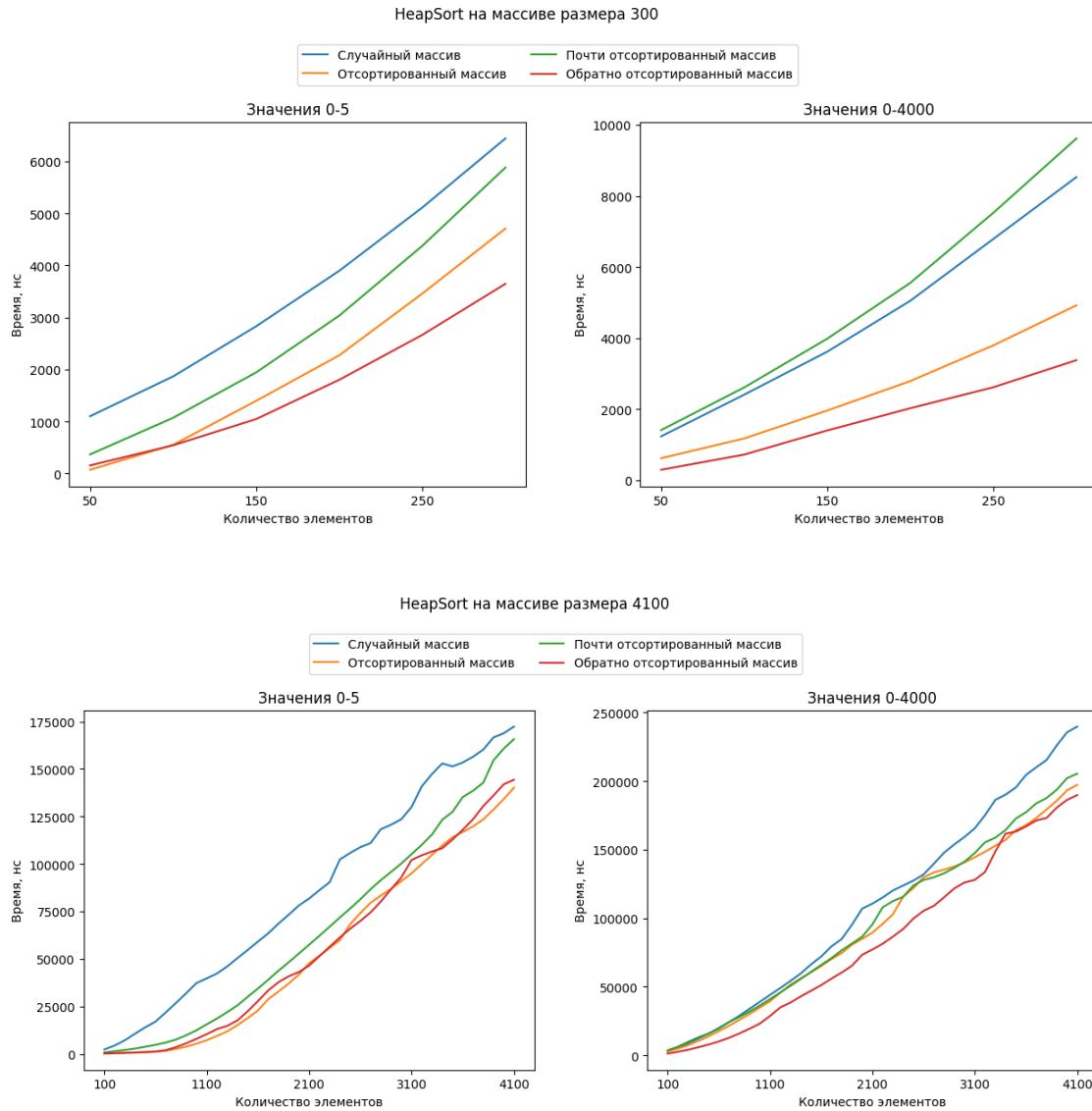
Теоретические данные о худшем случае подтверждаются экспериментально. График для отсортированного и обратно отсортированного массивов на диапазоне значений $[0; 4000]$ представляет собой график параболы.

На диапазоне $[0; 5]$ обратно отсортированный массив не выделяется. Это может быть связано с тем, что из-за небольшого диапазона значений выполняется гораздо меньше перестановок при сравнении с опорным элементом.

Пирамидалная сортировка

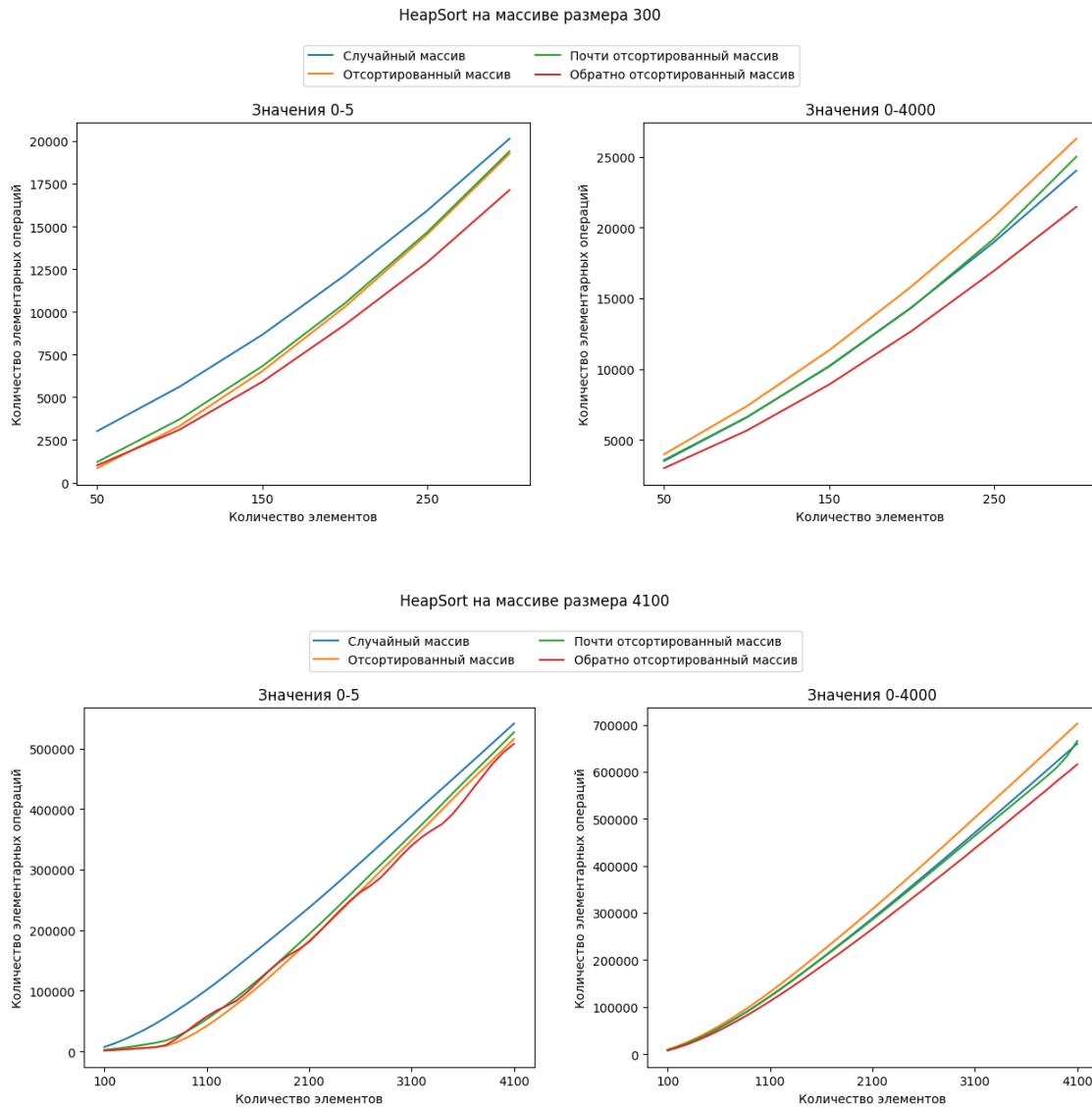
Пирамидалная сортировка использует кучу (пирамиду) для сортировки элементов. Во время сортировки элементы постепенно переносятся в кучу, затем извлекаются из нее в отсортированном порядке.

11.1 Измерение времени



Асимптотика алгорима составляет $O(n \log n)$ для любых входных данных, однако на обратно отсортированном массиве пирамидалная сортировка будет работать быстрее, чем на случайном и почти отсортированном массивах, так как в этом случае потребуется меньше сравнений. По этой же причине на отсортированном массиве пирамидалная будет работать еще быстрее, чем на обратно отсортированном.

11.2 Измерение элементарных операций



Количество элементарных операций для разных типов массивов находится в пределах погрешности и пропорционально $O(n \log n)$.

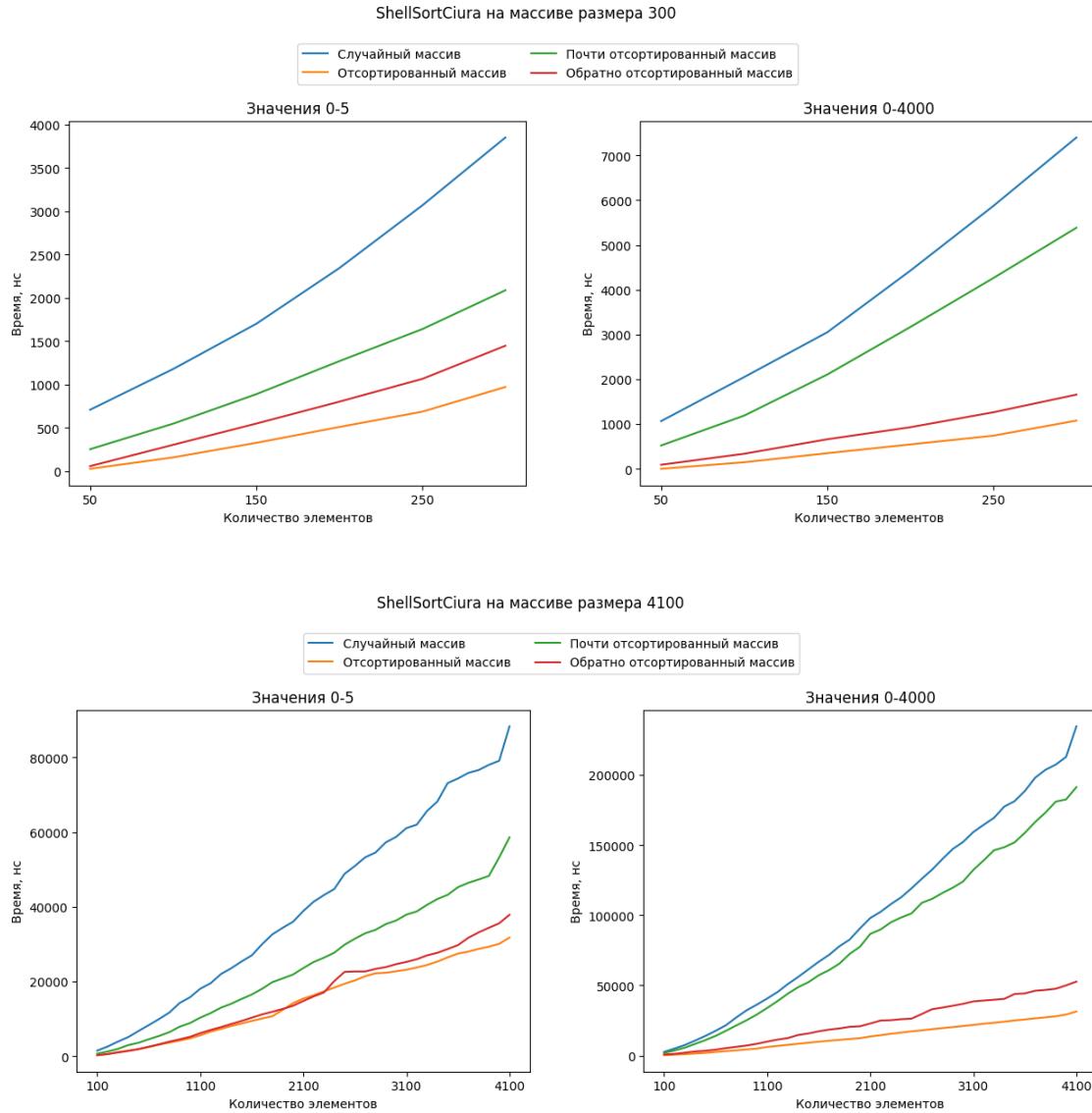
11.3 Вывод по наблюдениям

Теоретические данные полностью подтвердились экспериментально. На массиве размера 4100 можно заметить два скачка, что может быть связано с фоновыми процессами.

Сортировка Шелла с последовательностью Циура

Сортировка Шелла с последовательностью Циура — это усовершенствованный вариант сортировки вставками. Она использует последовательность шагов, которая позволяет быстрее уменьшать количество инверсий в массиве. Последовательность Циура, которая состоит из шагов вида 1, 4, 10, 23, 57, 132, 301, 701, является одной из самых эффективных последовательностей.

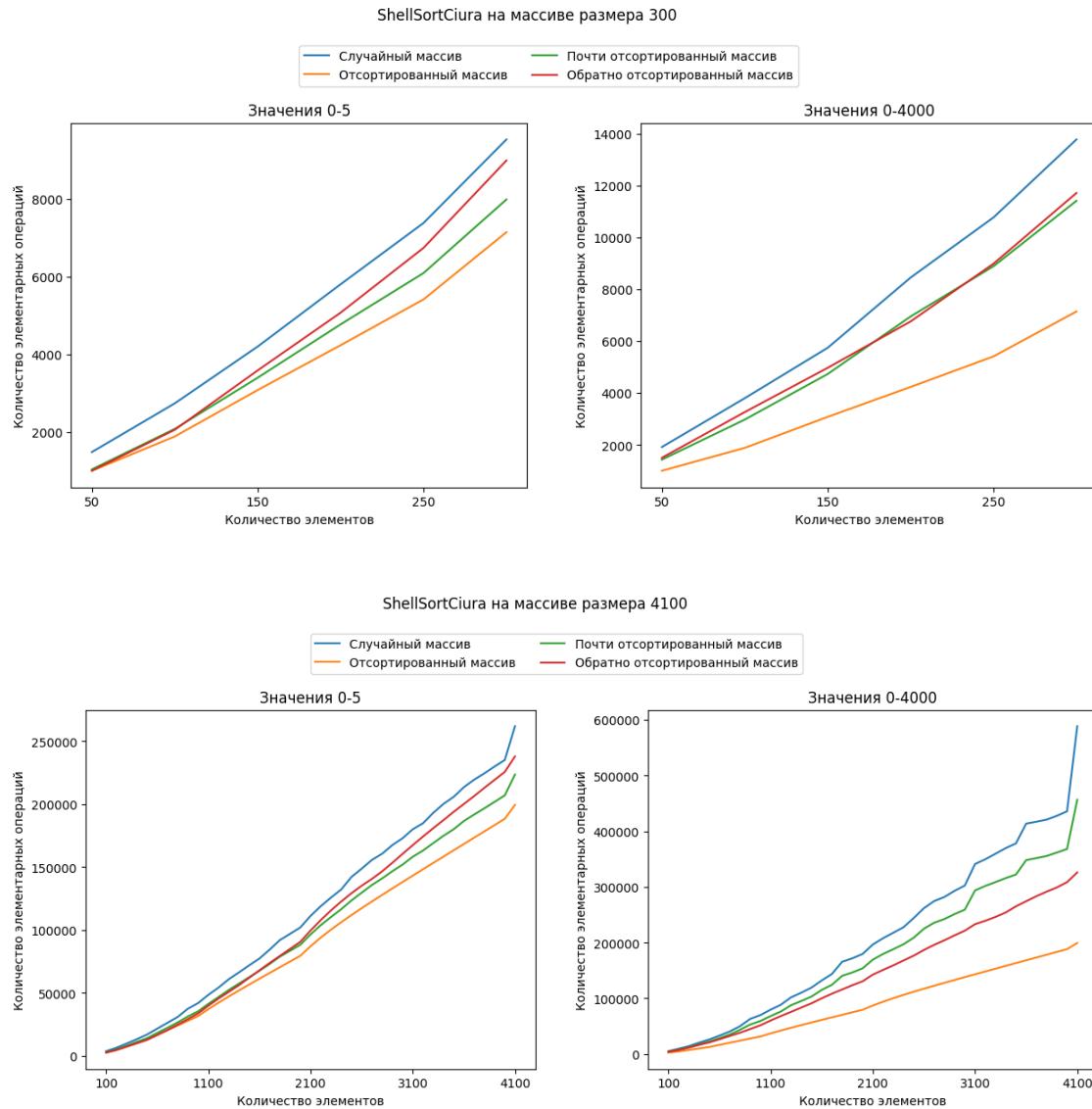
12.1 Измерение времени



Асимптотическая сложность сортировки составляет:

- $O(n^2)$ в худшем случае, при большом количестве инверсий.
- $O(n)$ в лучшем случае на отсортированном или почти отсортированном массиве. Тогда элементы находятся ближе к своим окончательным позициям.
- $O(n \log n)$ в среднем случае.

12.2 Измерение времени



Количество элементарных операций пропорционально сложности для времени.

12.3 Вывод по наблюдениям

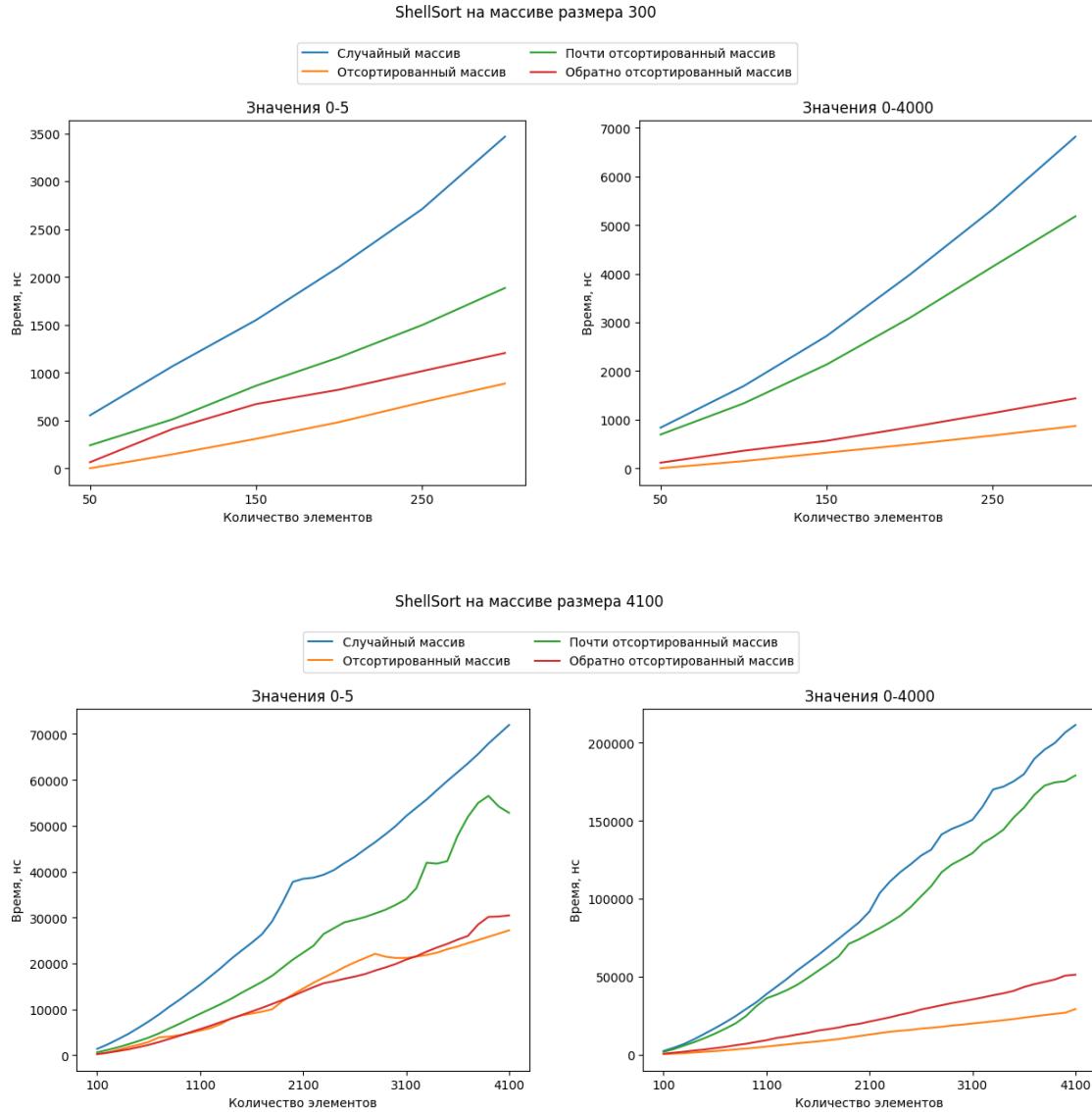
Теоретические данные подтверждаются экспериментально. Все графики похожи на $O(n \log n)$, худший случай не попался. лучший случай замечен на отсортированном массиве.

Сортировка Шелла с последовательностью Шелла

Сортировка Шелла с последовательностью Шелла выбирает шаг d — расстояние между элементами, которые будут сравниваться и меняться местами. Затем выполняется сортировка вставками для элементов, находящихся на расстоянии d . Шаг d уменьшается на каждой итерации до тех пор, пока не достигнет 1.

Первый интервал между элементами равен длине массива. На каждом следующем шаге интервал уменьшается вдвое.

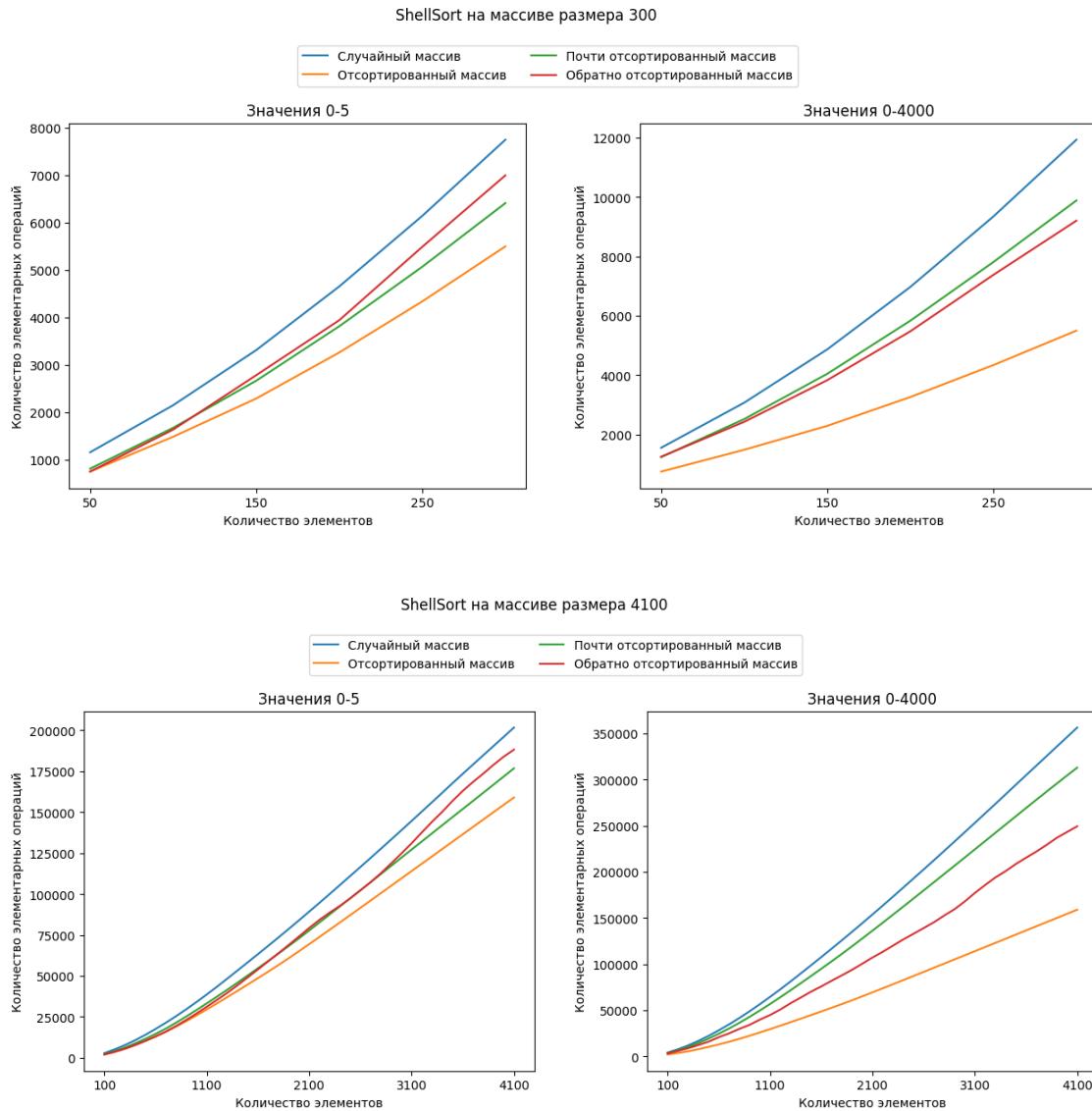
13.1 Измерение времени



Асимптотическая сложность сортировки составляет:

- $O(n^2)$ в худшем случае, при большом количестве инверсий.
- $O(n)$ в лучшем случае на отсортированном или почти отсортированном массиве. Тогда элементы находятся ближе к своим окончательным позициям.
- $O(n \log n)$ в среднем случае.

13.2 Измерение элементарных операций



Количество элементарных операций пропорционально времени сортировки.

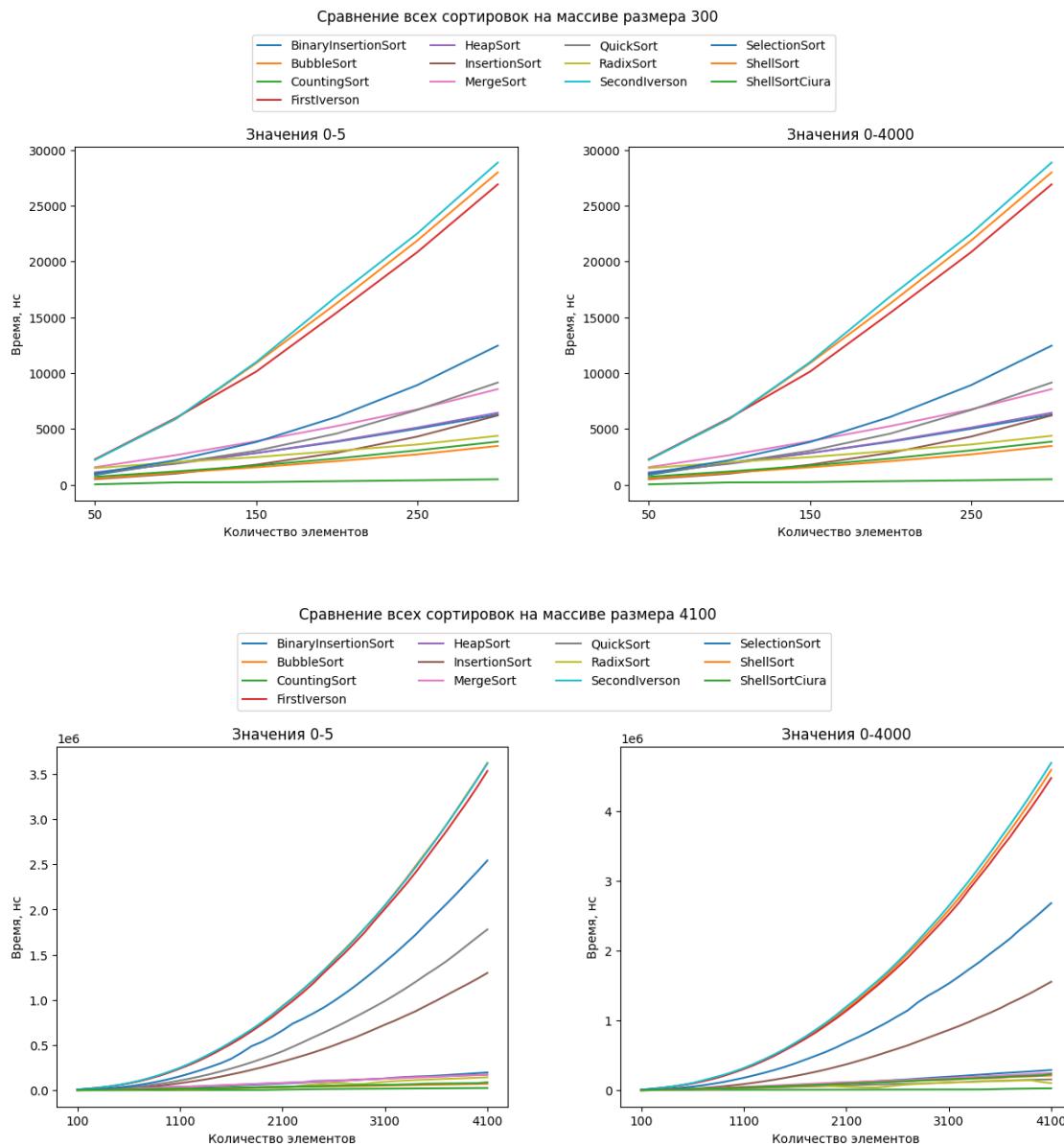
13.3 Вывод по наблюдениям

Теоретические данные подтверждаются экспериментально. Разницу во времени между случайнym и почти отсортированным массивами и отсортированным и отсортированным в обратном порядке можно объяснить тем, что все элементы будут находиться на своих местах сразу после первой сортировки групп элементов с максимальным расстоянием между ними.

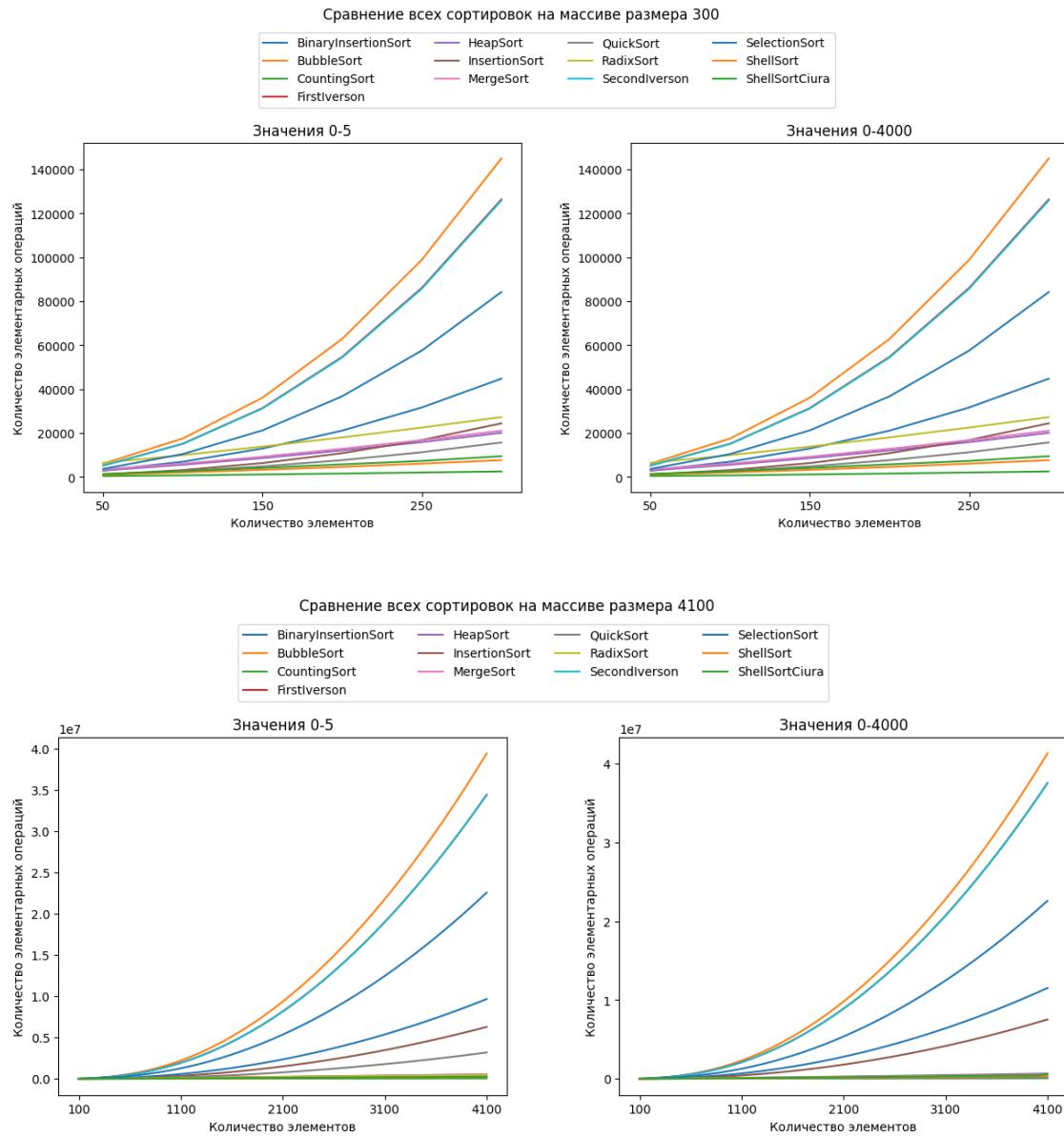
Если массив почти отсортирован, то время работы сортировки Шелла может оказаться даже хуже, чем на случайно расположенных элементах, потому что на начальных этапах сортировки могут происходить большие перестановки элементов, которые уже почти на своих местах.

Случайный массив

14.1 Время

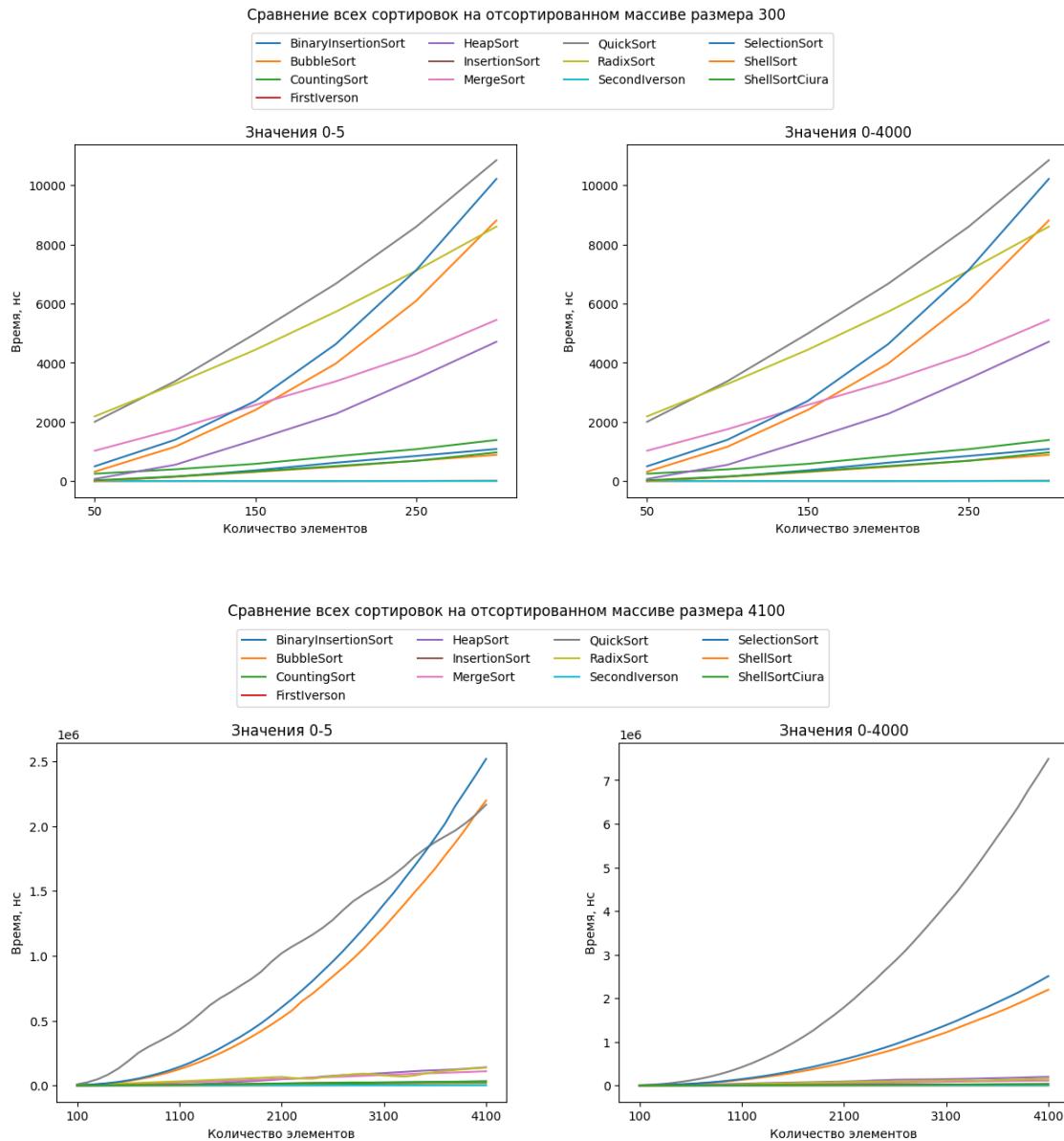


14.2 Элементарные операции

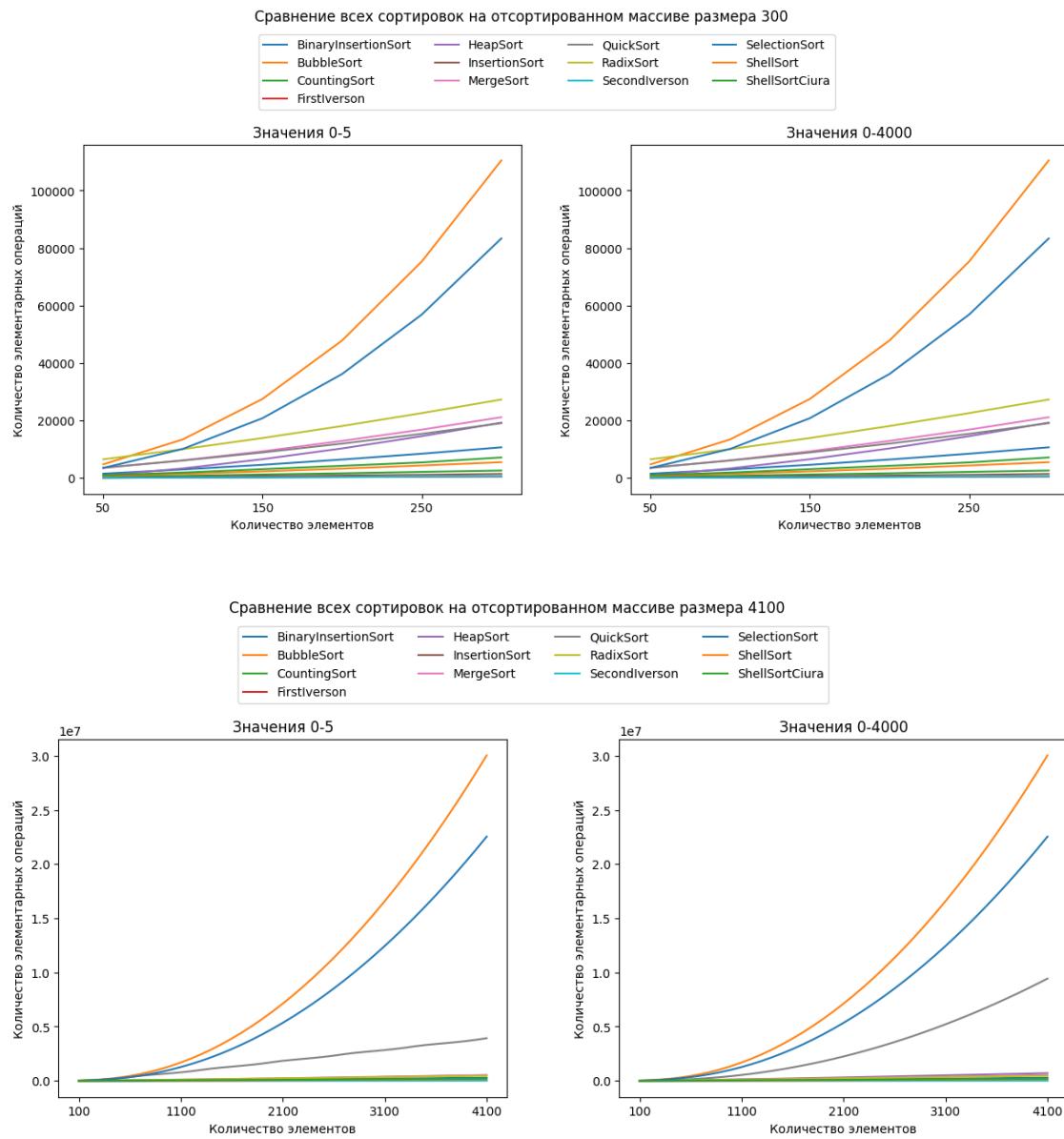


Отсортированный массив

15.1 Время



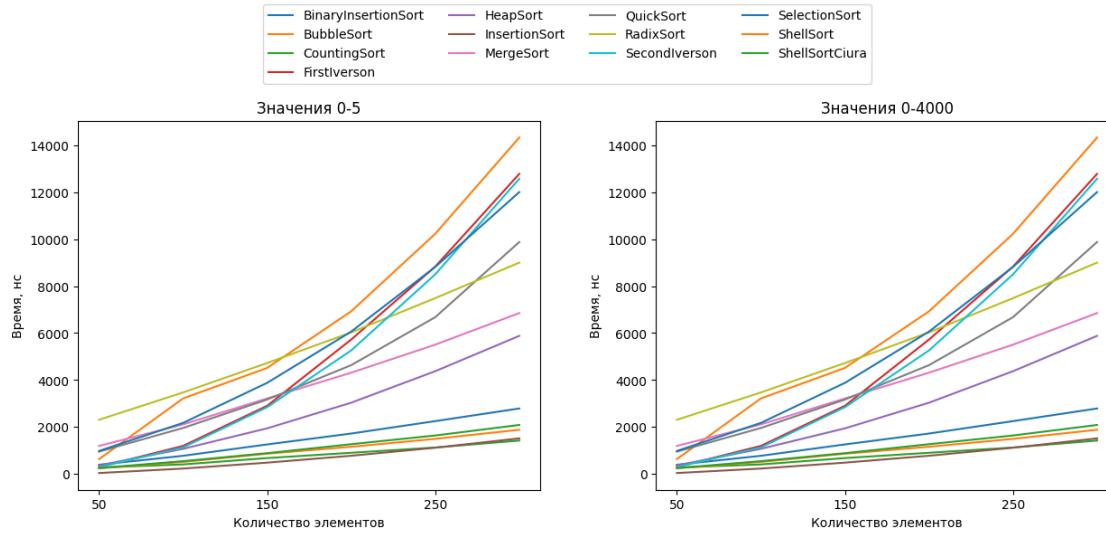
15.2 Элементарные операции



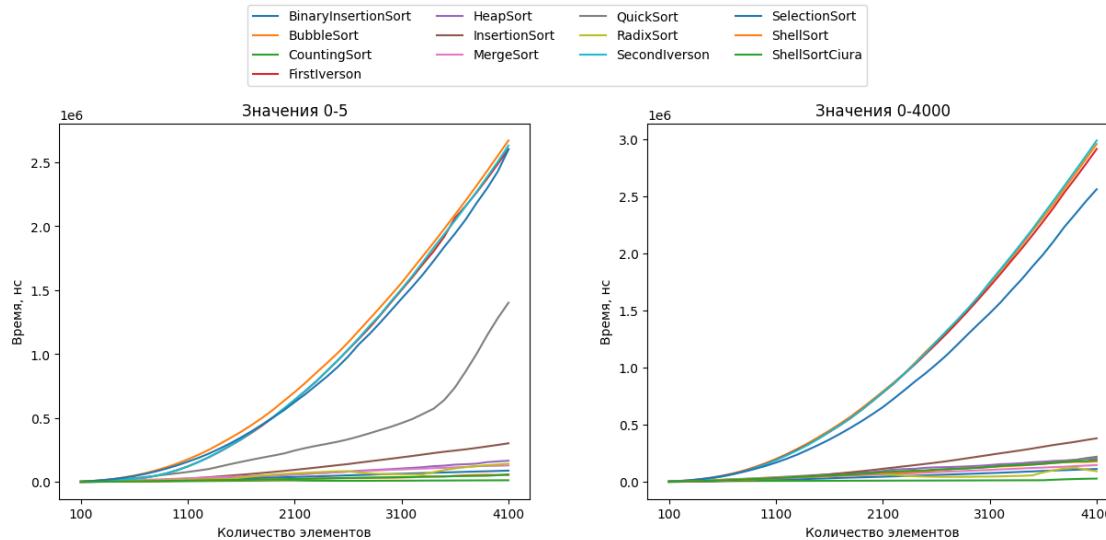
Почти отсортированный массив

16.3 Время

Сравнение всех сортировок на почти отсортированном массиве размера 300

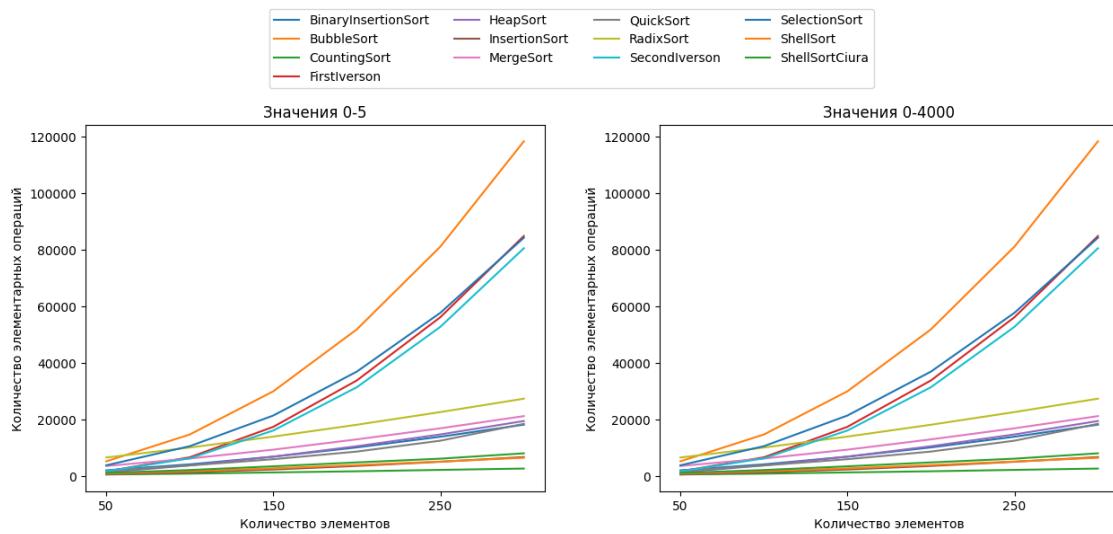


Сравнение всех сортировок на почти отсортированном массиве размера 4100

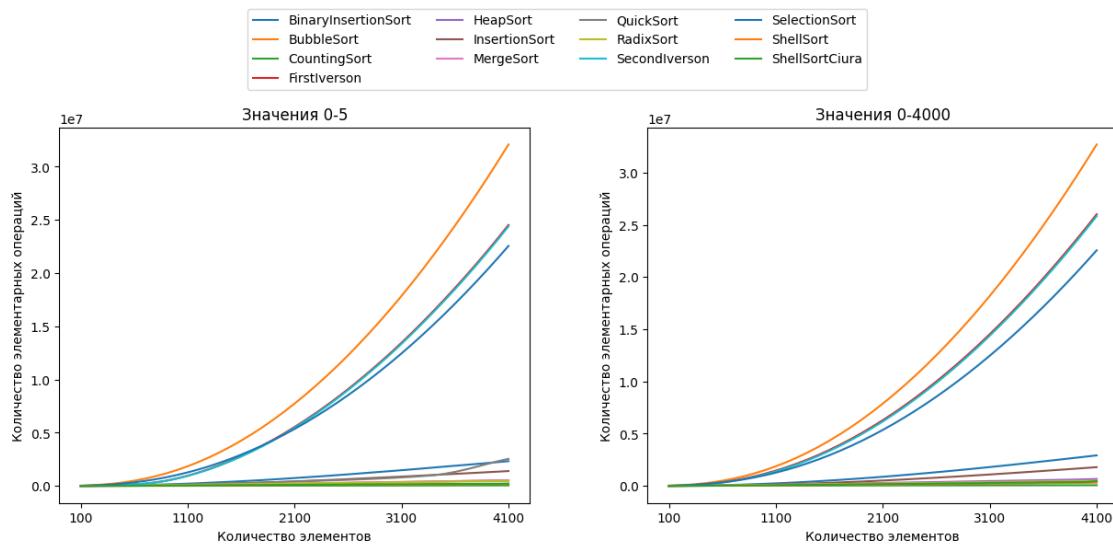


16.4 Элементарные операции

Сравнение всех сортировок на почти отсортированном массиве размера 300



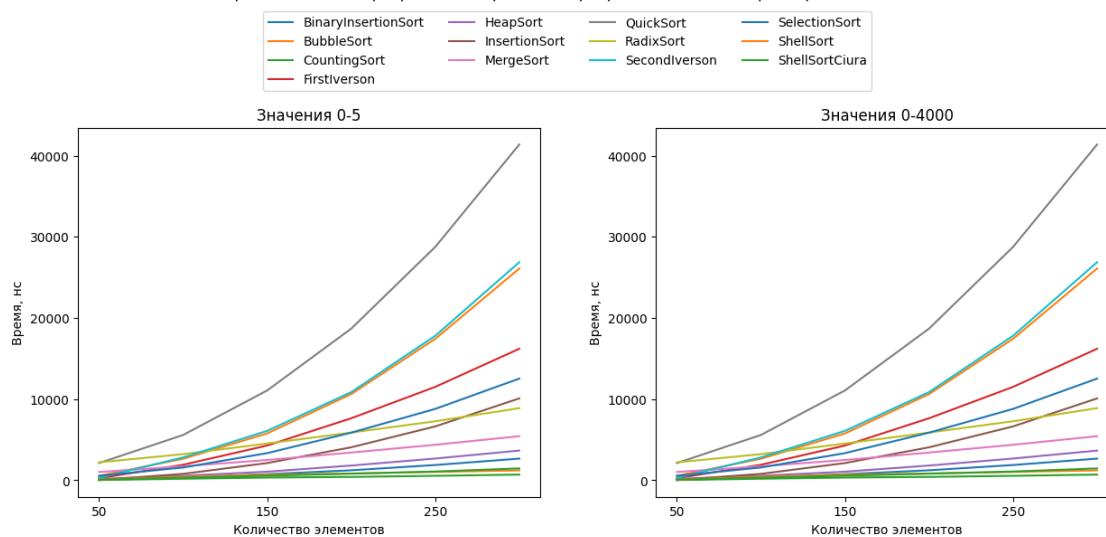
Сравнение всех сортировок на почти отсортированном массиве размера 300



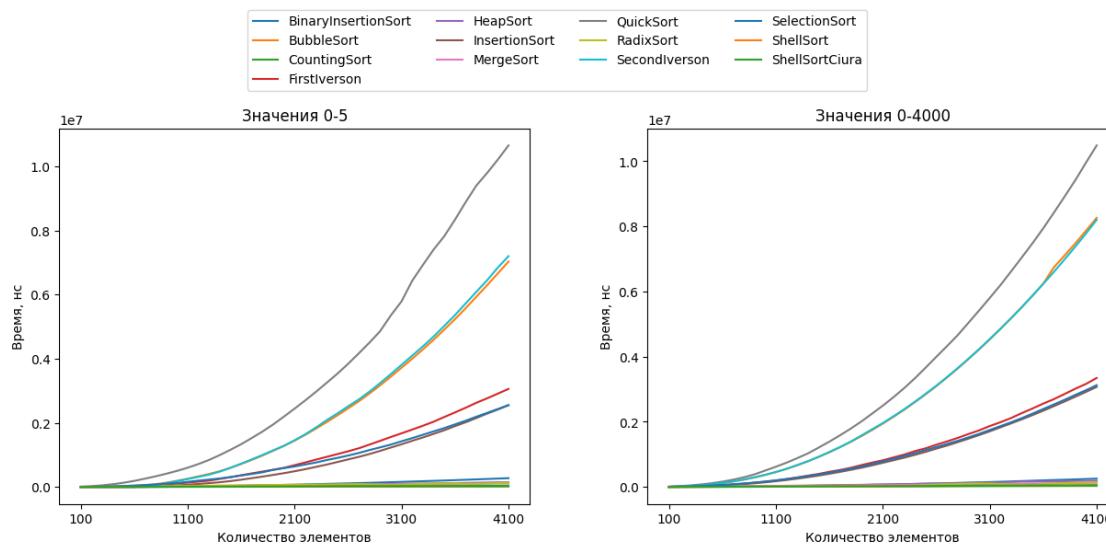
Обратно отсортированный массив

17.5 Время

Сравнение всех сортировок на обратно отсортированном массиве размера 300

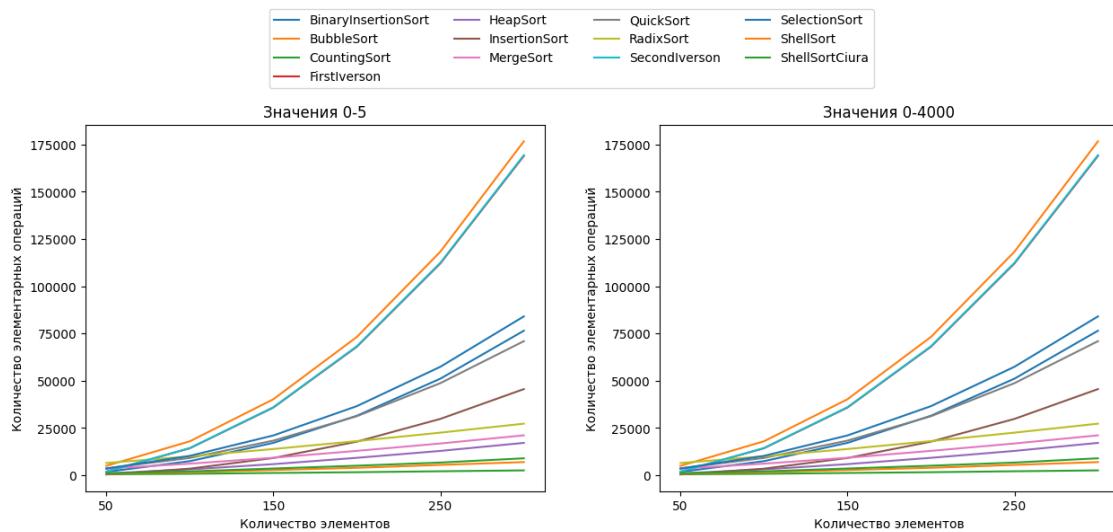


Сравнение всех сортировок на обратно отсортированном массиве размера 4100

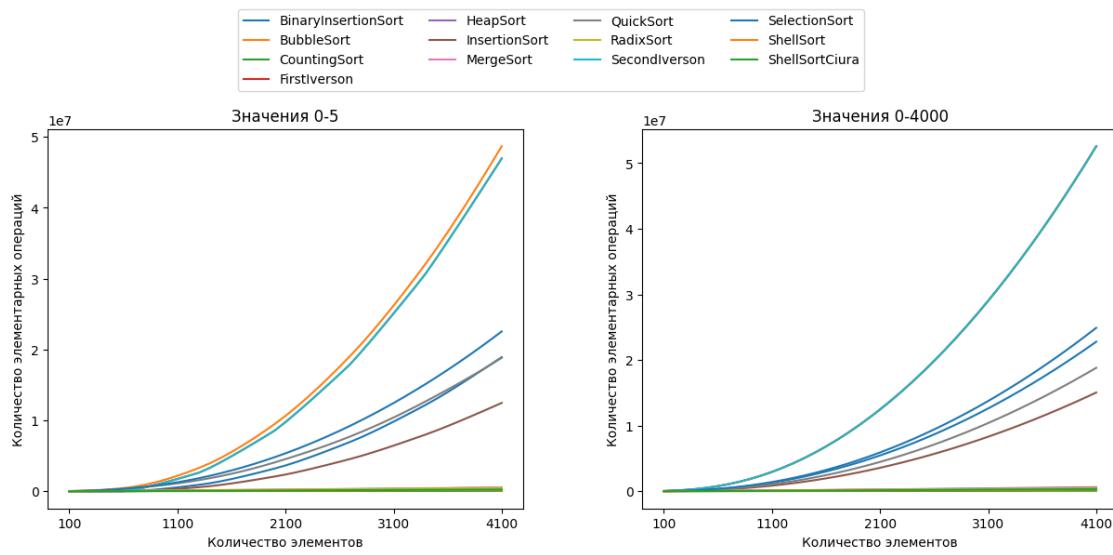


17.6 Элементарные операции

Сравнение всех сортировок на обратно отсортированном массиве размера 300



Сравнение всех сортировок на обратно отсортированном массиве размера 300



Вывод

По результатам измерений удалось практически подтвердить большинство теоретических оценок асимптотики алгоритмов. Для расходящихся с теорией данных удалось найти объяснение, из-за чего это произошло.

На общих графиках можно увидеть, как алгоритмы со сложностью $O(n^2)$ выделяются на фоне сложностей $O(n \log n)$ и $O(n)$.

Из-за большого масштаба оси y время работы части алгоритмов неотличимо от 0, однако это не так, стоит посмотреть на отдельные графики этих сортировок.

Выбрать лучший алгоритм невозможно, так как их эффективность зависит от входных данных. Однако теперь, зная причины по которым тот или иной алгоритм работает дольше, можно выбирать оптимальный под конкретную задачу.