Отчет

«Алгоритмы и анализ сложности»

Студент:

Гусев Виталий Евгеньевич

09-832

Оглавление

[Введение. 2](#_Toc37755651)

[Сортировка случайно генерированных массивов. 6](#_Toc37755652)

[Случайные даты. 6](#_Toc37755653)

[Случайные цифры (от 0 до 9) 8](#_Toc37755654)

[Случайные целые числа 11](#_Toc37755655)

[Случайные строки 14](#_Toc37755656)

[Сортировка массивов из одного элемента. 17](#_Toc37755657)

[Один элемент даты. 17](#_Toc37755658)

[Один элемент цифры. 19](#_Toc37755659)

[Один элемент целых чисел. 21](#_Toc37755660)

[Один элемент строк. 23](#_Toc37755661)

[Сортировка частично отсортированных массивов. 25](#_Toc37755662)

[Отсортированные даты. 25](#_Toc37755663)

[Отсортированные цифры. 27](#_Toc37755664)

[Отсортированные целые числа. 29](#_Toc37755665)

[Отсортированные строки. 32](#_Toc37755666)

[Вывод. 34](#_Toc37755667)

# Введение.

Все сортировки реализовывались на языке Python в среде разработки PyCharm. Все тесы проводились на ноутбуке HP Probook 650 (G1). Процессор Intel® Core™ i5-4300M CPU @ 2.60GHz 2.60GHz. Оперативная память 4,00 ГБ. 64-разрядная операционная система.

Проект расположен в папке Algorithms. Функция каждой сортировки реализовывалась в отдельных файлах с расширением «.py». Были реализованы и изучены следующие сортировки:

-Bubble sort (сортировка пузырьком). Bubble\_Sort.py

-Comb sort (сортировка расческой). Comb\_Sort.py

-Heap sort (сортировка кучей). Heap\_Sort.py

-Insertion sort (сортировка вставками). Insertion\_Sort.py

-Merge sort (сортировка слиянием). Merge\_Sort.py

-Python sort (сортировка, встроенная в язык). Python\_Sort.py

-Quick sort (быстрая сортировка). Quick\_Sort.py

-Radix sort (поразрядная сортировка). Radix\_Sort.py

-Shaker sort (шейкерная сортировка). Shaker\_Sort.py

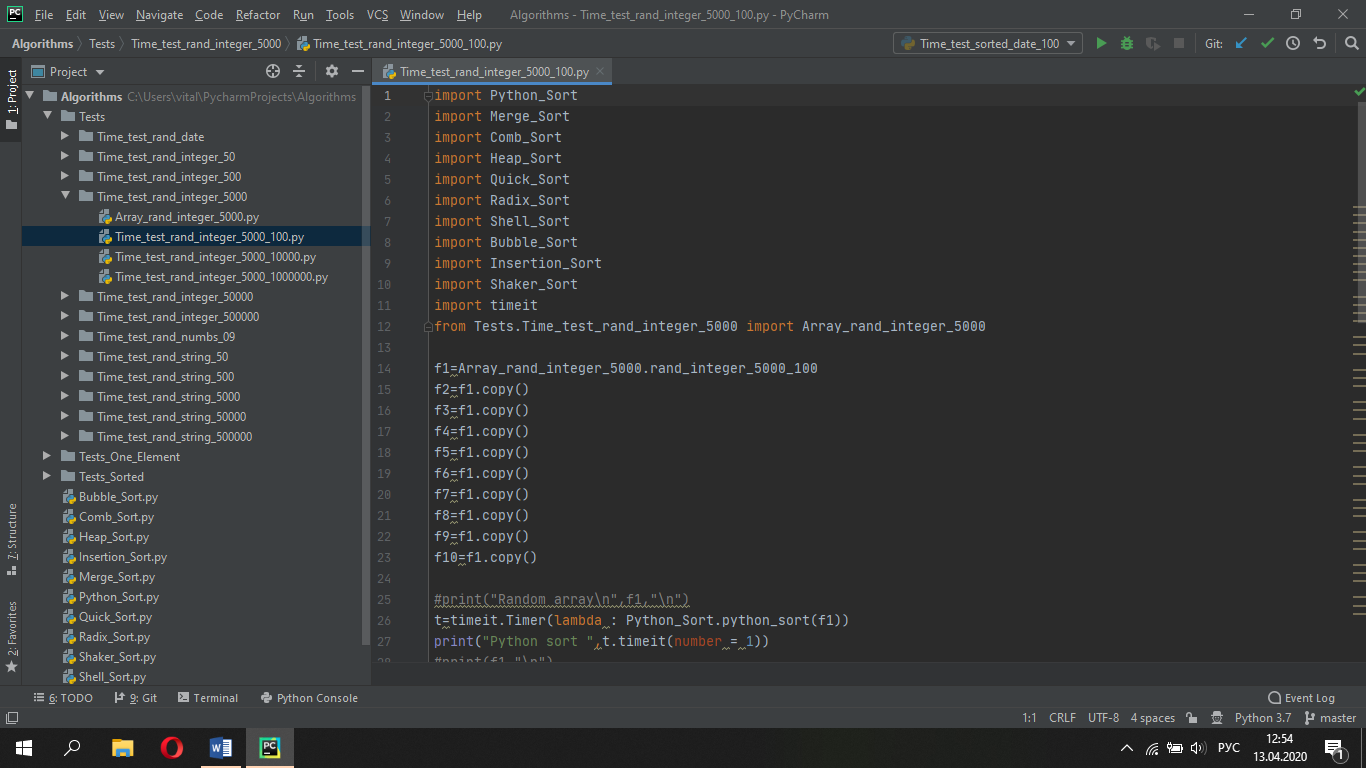
-Shell sort (сортировка Шелла). Shell\_Sort.py

Все тесты проводились в папках:

-Tests. Случайно сгенерированные массивы.

-Tests\_One\_Element. Массивы из одного элемента.

-Tests\_Sorted. Частично упорядоченные массивы.



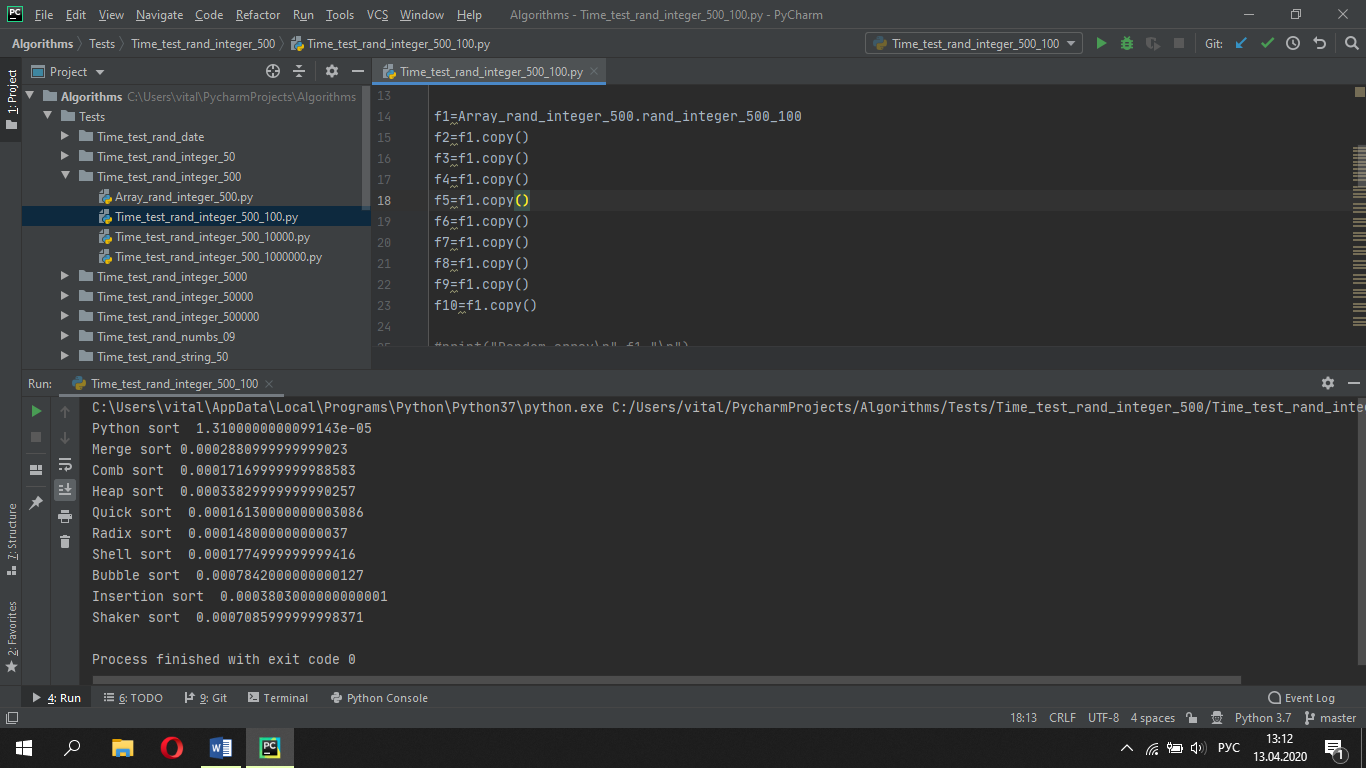
Все тесты в папке «Tests» делились на папки, именованные следующим образом. Например, «Time\_test\_rand\_integer\_5000». «Time\_test\_...», значит тест времени, все папки начинаются с этого имени. «…rand\_...» случайное число, «…integer\_» целые числа, «…5000» порядок 5000 элементов. В этих папках находятся 4 файла. Первый файл «Array\_rand\_integer\_5000», в этом файле генерируются случайные массивы целых чисел длины 100, 10000 и 1000000 порядка 5000 элементов. «Time\_test\_rand\_integer\_5000\_100», в этом файле проводятся тесты всех сортировок, импортированного массива на 100 элементов из «Array\_rand\_integer\_5000». Этот массив копируется для всех сортировок, каждая сортировка сортирует этот массив и выводится в результате время, которое ушло на сортировку этого массива. «Time\_test\_rand\_integer\_5000\_10000» и «Time\_test\_rand\_integer\_5000\_1000000», проводят те же тесты, только с массивами на 10000 и 1000000 элементов соответственно.

Подобным образом реализованы тесты в остальных папках. «Time\_test\_rand\_date» - тест случайных дат. «Time\_test\_rand\_numbs\_09» - тест случайных цифр от 0 до 9. «Time\_test\_rand\_string\_500» - тест случайных строк, длиной 500.

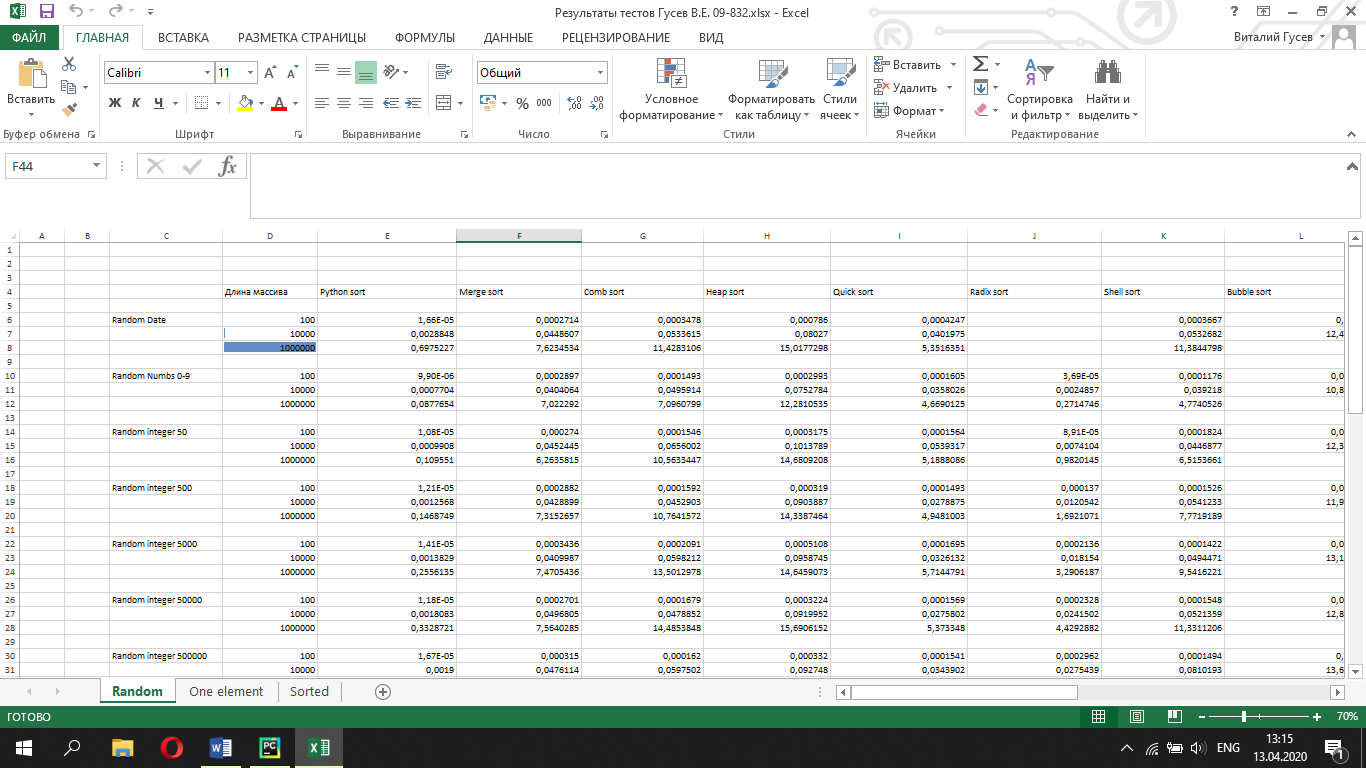
Тесты в папках «Test\_One\_Element» и «Test\_Sorted» именованы подобным образом.

Тесты в «Test\_One\_Element» сортировалась одна дата «date(2018, 8, 13)», одно целое число «1234», одна цифра «7», одна строка «Skibidi».

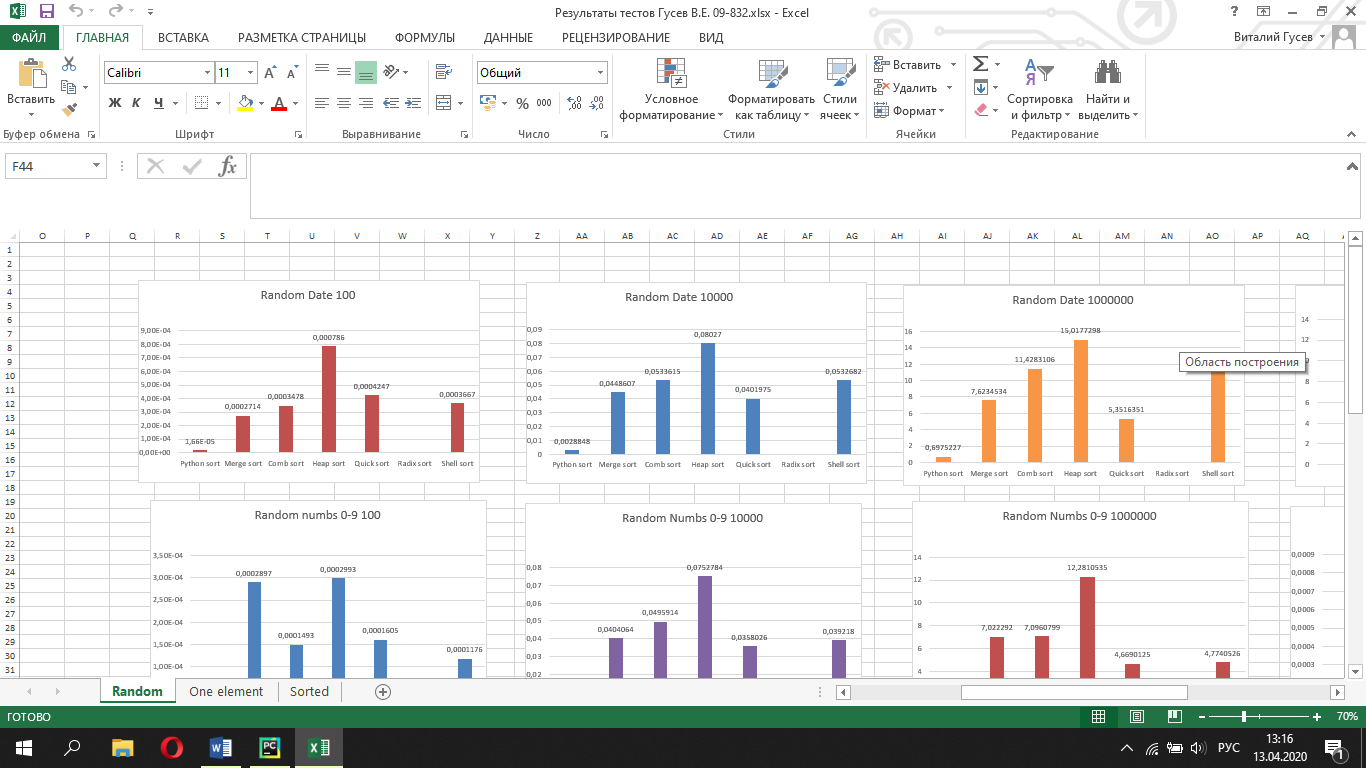
Тесты в «Test\_Sorted» массивы отсортированы на половину в середине. Массивы на 100 элементов отсортированы на интервале [25:75], массивы на 10000 отсортированы на интервале [2500:7500] и массивы на 1000000 на интервале [250000:750000].



Результаты времени заносились в таблицу Excel «Результаты тестов Гусев В.Е. 09-832.xlsx»



Ниже в отдельных листах «Random», «One element» и «Sorted» вводились результаты для случайных данных, одного элемента и отсортированных.



По результатам были созданы наглядные диаграммы. Эти диаграммы вставлены в файл Word «Отчет Гусев В.Е. 09-832.docx» и здесь проанализированы.

# Сортировка случайно генерированных массивов.

## Случайные даты.

Самая эффективная сортировка была встроенная в язык программирования Python. Среди эффективных алгоритмов сортировка массива размерностью 100 самую наибольшую эффективность проявила Сортировка слиянием. На массивах размерностью 10000 и 1000000 наибольшую эффективность начала проявлять Быстрая сортировка. Неэффективность показала на всех массивах Сортировка кучей.

Среди «простых сортировок» самой эффективной проявилась Сортировка вставками. Неэффективной на 100 элементов оказалась Шейкерная сортировка. На 10000 элементов неэффективными оказались Шейкерная и Сортировка пузырьком, время сортировки примерно одинаковы. Сортировка массива на 1000000 «простыми схемами» идет слишком долго.

Поразрядная сортировка не участвовала в тестах.

## Случайные цифры (от 0 до 9)

Наилучшую эффективность сортировки проявил встроенный в язык программирования Python. Среди быстрых сортировок наилучшие результаты с большим отрывом показала Поразрядная сортировка. Наихудший результат продемонстрировала Сортировка кучей.

Среди «простых схем» наилучший результат продемонстрировала Сортировка вставками. Сортировка пузырьком и Шейкерная сортировка показали себя одинаково плохо с небольшой разницей друг от друга.

## Случайные целые числа

Были проведены тесты случайных целых чисел порядка 50, 500, 5000, 50000, 500000 элементов и во всех сортировках размерность не влияло на длительность, кроме Поразрядной сортировки - с увеличением размерности, увеличивалась длительность сортировки. Как обычно, наилучший результат во всех тестах показала сортировка, встроенная в язык программирования Python.

Среди быстрых сортировок в массиве размерностью 100 сортировка целых чисел порядка 50, 500 наилучший результат показала Поразрядная сортировка. С целыми числами порядка 5000, 50000, 500000 она начала проигрывать Сортировке расческой, Быстрой сортировке и Сортировке Шелла. Сортировки кучей и слиянием показали себя одинаково плохо. Поразрядная сортировка порядка 500000 элементов дошла до длительности Сортировки кучей и слиянием.

В массиве размерностью 10000 и 1000000 при любом порядке лучший результат показывала Поразрядная сортировка. Худшей была сортировка кучей.

Среди «простых схем» лучший результат на массивах размерностью 100 и 10000 с элементами любого порядка целых чисел показала Сортировка вставками. Сортировка пузырьком и Шейкерная сортировка показали себя одинаково плохо.

Можно сделать вывод, что с массивами целых чисел лучше использовать поразрядную сортировку, кроме небольших массивов с элементами большего порядка. В этом случае эффективнее использовать Быструю сортировку.

## Случайные строки

Лучшие показатели у сортировки, встроенной в язык программирования Python.

Среди быстрых сортировок с массивом на 100 элементов хорошо справились Быстрая сортировка и Сортировка Шелла. Так же хорошо показала себя Сортировка расческой с элементами размером 50 и 50000. С элементами размером 500 плохо проявила себя Сортировка расческой, с элементами 5000 плохой показатель у Сортировки кучей, с элементами 50000 отстала по времени Сортировка слиянием.

Среди быстрых сортировок с массивом на 10000 элементов хорошо проявила себя Быстрая сортировка, плохо –Сортировка кучей.

Среди «простых схем» на массивах 100 и 10000 элементов хорошо проявила Сортировка вставками. Плохие результаты показали Сортировка пузырьком и Шейкерная сортировка.

Поразрядная сортировка не участвовала в тестах.

# Сортировка массивов из одного элемента.

## Один элемент даты.

Во всех опытах лучший показатель времени сортировки у встроенного в язык программирования Python. Худший показатель во всех тестах был у сортировки Пузырьком. Лучшие показатели у простых сортировок – у Сортировки вставками и у Шейкерной сортировки. Показали лучше результаты, чем эффективные сортировки. Среди эффективных сортировок хорошие показатели у сортировки Кучей. С 100 элементами такой же хороший результат, как у сортировки Кучей у сортировки Шелла. Плохой результат с 100 и 10000 элементами у сортировки Слиянием, а с элементами на 1000000 худший результат у сортировки расческой. Поразрядная сортировка не участвовала в тестах.

## Один элемент цифры.

Лучший результат у встроенной в язык программирования в Python. Лучшие показатели сортировки показали простые схемы, такие как Сортировка вставками и Шейкерная сортировка. Среди эффективных сортировок лучший результат показала поразрядная сортировка, но она уступает простым схемам. Худший результат у сортировки пузырьком. Среди эффективных сортировок худший результат на 100 элементов показала сортировка Слиянием, с 10000 и 1000000 элементами худший результат у сортировки Расческой.

## Один элемент целых чисел.

Лучший результат у встроенной в язык программирования Python. Лучшие результаты показали простые схемы – Сортировка вставками и Шейкерная сортировка. Эффективные алгоритмы показали более плохие результаты. Сортировка пузырьком показала худшие результаты во всех тестах. Среди эффективных алгоритмов лучший результат на 100 элементов у Сортировки Шелла, на 10000 и 1000000 элементов лучший результат показала Сортировка кучей. Худший результат на 100 элементов у сортировки Слиянием, на 10000 и 1000000 у сортировки Расческой.

## Один элемент строк.

Лучший результат у встроенной в язык программирования Python. Лучшие результаты показали простые сортировки – Вставками и Шейкерная. Эффективные сортировки показали хуже результаты. Сортировка пузырьком показала самые плохие результаты. Среди эффективных сортировок на 100 элементов лучший результат у сортировки Шелла, на 10000 и 1000000 лучший результат у Сортировки кучей. Сортировка слиянием, Сортировка расческой и Быстрая сортировка показали худшие результаты. Поразрядная сортировка в тестировании не участвовала.

# Сортировка частично отсортированных массивов.

## Отсортированные даты.

Лучшие результаты у встроенного в язык программирования Python. Среди эффективных сортировок худшие результаты на 100 и 10000 элементов у сортировки кучей, на 1000000 у быстрой сортировки. На 100 элементов хорошие показатели у сортировки расческой, быстрой сортировки и сортировки Шелла. На 10000 и 1000000 элементов быстрой оказалась сортировка слиянием. У простых схем плохие показатели у сортировки пузырьком и шейкерной сортировки. Сортировка вставками показала лучше результат, но уступает эффективным сортировкам. Поразрядная сортировка не участвовала в тестах.

## Отсортированные цифры.

Лучший показатель у встроенного в язык программирования Python. Среди эффективных сортировок худший показатель у сортировки кучей. Лучший показатель у поразрядной сортировки. Среди простых сортировок худшие показатели у сортировки пузырьком и шейкерной сортировки, хороший показатель у сортировки вставками.

## Отсортированные целые числа.

Лучший показатель у встроенной в язык программирования Python. На массивах 100 элементов на числах размерностью 50 и 500 выигрывает поразрядная сортировка, но на 5000, 50000 и 500000 она начинает проигрывать сортировке расческой, быстрой сортировке и сортировке Шелла. На массивах размерностью 10000 и 1000000 везде выигрывает поразрядная сортировка. На массивах 100 и 10000 элементов проигрывает сортировка кучей. На массивах 1000000 элементов на числах порядка 50 и 500 проигрывает сортировка кучей, с порядком 5000, 50000 и 500000 проигрывает быстрая сортировка. Среди простых сортировок проигрывают сортировка пузырьком и шейкерная сортировка, лучше результат у сортировки вставками.

## Отсортированные строки.

Лучшие показатели у встроенной в язык программирования Python. Среди эффективных сортировок хорошие показатели у сортировки слиянием, сортировки расческой, быстрой сортировки и сортировки Шелла. Худший показатель у сортировки кучей. Среди простых сортировок худшие показатели у сортировки пузырьком и шейкерной сортировки, лучше показатели у сортировки вставками. Поразрядная сортировка не участвовала в тестировании.

# Вывод.

По результатам тестов встроенная в язык программирования Python сортировка показывает лучшие результаты во всех тестах.

Сортировка слиянием оказалась очень эффективной, но неэффективной при сортировке малых массивов из одного элемента. O(n log n)

Сортировка расческой оказалась эффективной при сортировке малых массивов, с увеличением размера массива эффективность сортировки падает. Лучшее время O(n logn), худшее время O(n^2)

Сортировка кучей показала самую плохую эффективность при сортировке случайно сгенерированных и частично отсортированных массивов, но очень хорошую эффективность при сортировке массивов из одного элемента. Время O(n log n)

Быстрая сортировка показала хорошую эффективность при сортировке больших случайно сгенерированных массивов. Плохая эффективность при сортировке массивов из одного элемента. Среднее время O(n log n)

Поразрядная сортировка используется только при сортировке чисел. Очень эффективная во всех тестах, но при сортировке маленьких массивов большого порядка эффективность по сравнению с другими сортировками падает. В среднем сложность O(nlogn)

Сортировка Шелла показала среднюю эффективность, по сравнению с сильными сортировками. Хорошие результаты при сортировке малых массивов. Лучшее время O(n log2 n)

Сортировка пузырьком оказалась одной из самых неэффективных сортировок. При работе с малыми массивами не сильно отстает от эффективных алгоритмов, при работе с большими сортирует слишком долго. Время O(n^2)

Сортировка вставками показывает хорошую эффективность на ровне с сильными сортировками при работе с малыми массивами. Оказалась самой эффективной при сортировке при работе с массивами из одного элемента, показатели лучше, чем у эффективных алгоритмов. Время в среднем O(n^2)

Шейкерная сортировка показывает такие же плохие результаты, как и сортировка пузырьком при работе со случайно сгенерированными массивами. Показывает лучшую эффективность, по сравнению с эффективными алгоритмами, при работе с массивами из одного элемента. В среднем O(n^2)