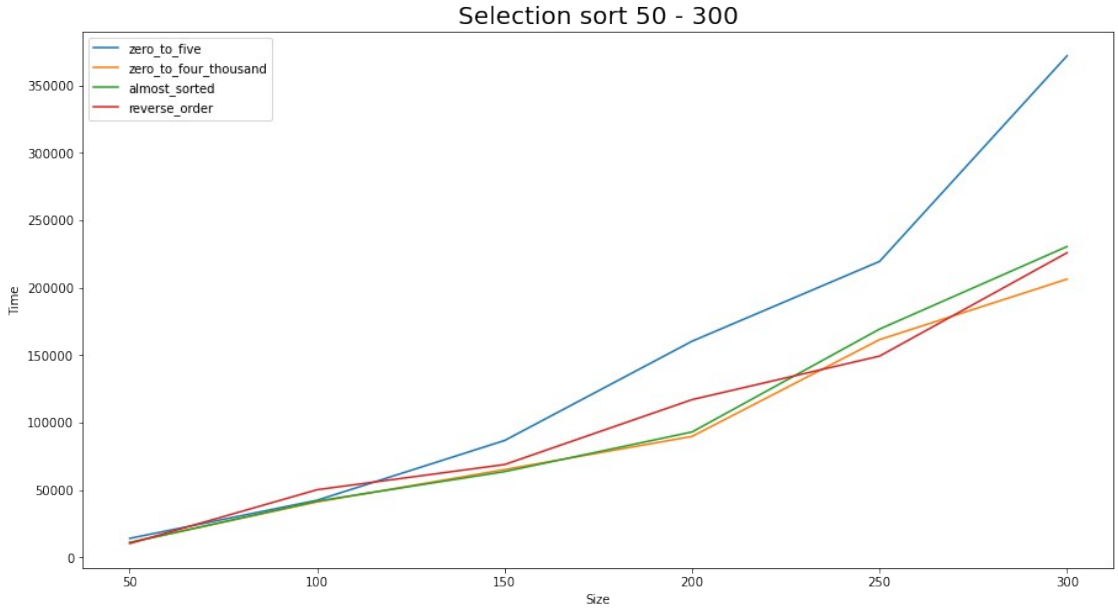
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

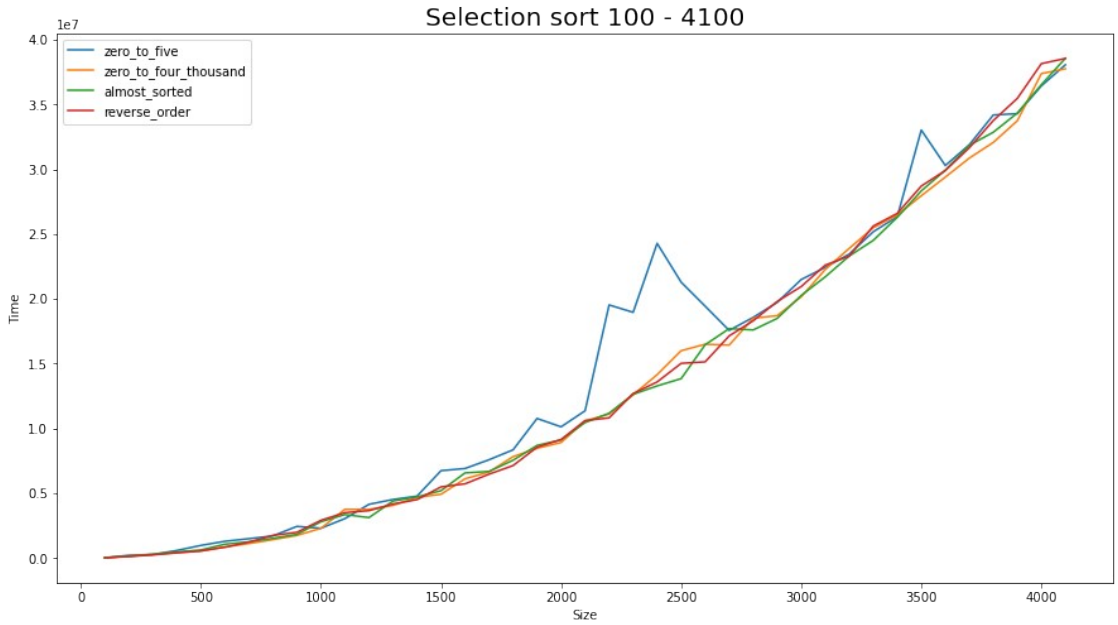
Selection sort

```
# Размерность 50 - 300
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(50, 301, 50):
    size.append(i)
array_type = ""
with open ("selection_sort 50 - 300.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Selection sort 50 - 300", fontsize= 20)
plt.legend(loc='best')
plt.show()
```

```
# Размерность 100 - 4100
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
```

```
reverse_order = []
for i in range(100, 4101, 100):
    size.append(i)
array_type = ""
with open ("selection_sort 100 - 4100.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Selection sort 100 - 4100", fontsize= 20)
plt.legend(loc='best')
plt.show()
```





Вывод к Selection sort:

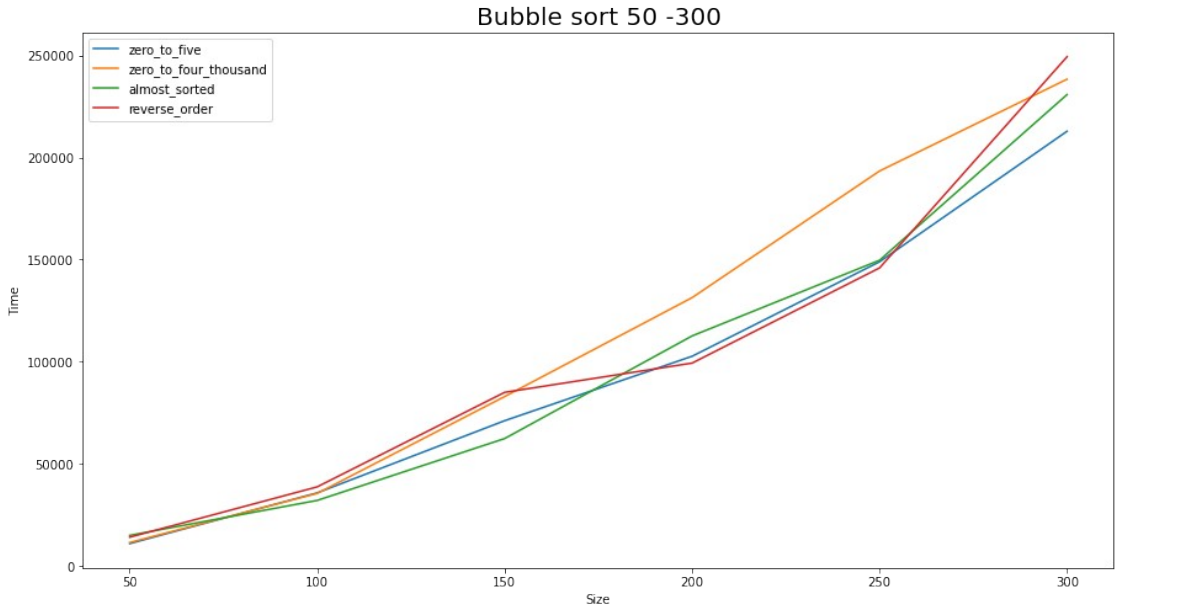
Знаем, что она работает за O(n^2), это хорошо видно на более объёмных данных, получаем что-то похожее на параболу. По графикам также видно, что на больших данных особо ничего не выделается, но на 50 - 300 на массив 0 - 5 тратится больше всего времени для сортировки. На 100 - 4100 лишь иногда наблюдаются скачки для данного массива.

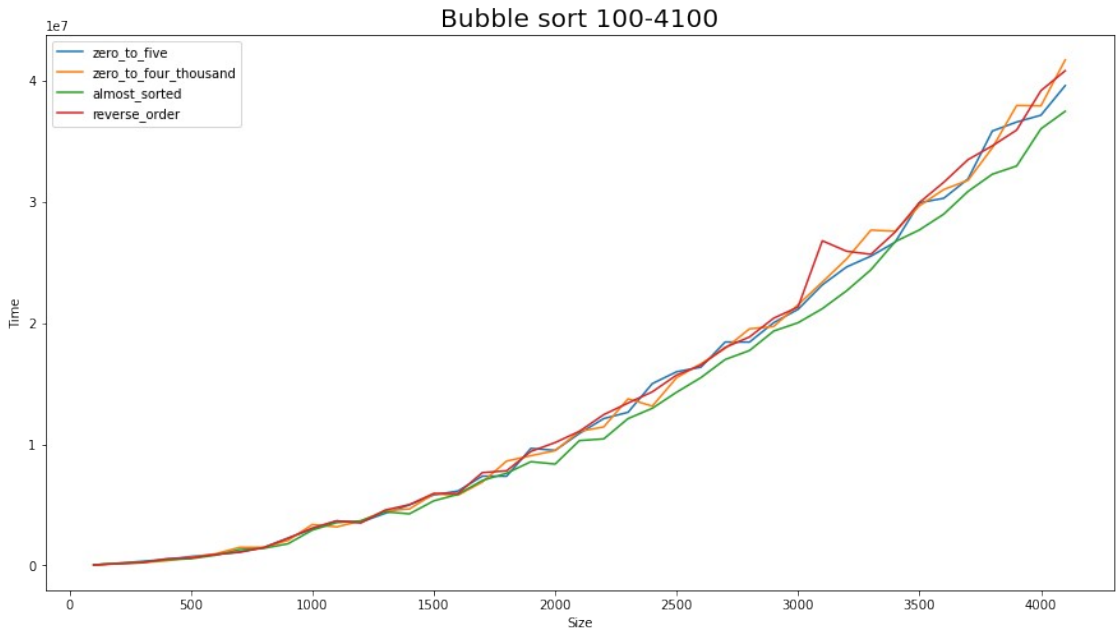
Bubble sort

```
# Размерность 50 - 300
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(50, 301, 50):
    size.append(i)
array_type = ""
with open ("bubble_sort 50 - 300.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Bubble sort 50 -300", fontsize= 20)
plt.legend(loc='best')
plt.show()


# Размерность 100 - 4100
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(100, 4101, 100):
    size.append(i)
array_type = ""
```

```
with open ("bubble_sort 100 - 4100.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Bubble sort 100-4100", fontsize= 20)
plt.legend(loc='best')
plt.show()
```





Вывод к Bubble sort:

На 100 - 4100 очень хорошо видна парабола, что доказывает асимптотику O(n^2). Видно что, что на 50 - 300 с увеличением размера массив на числах от 0 до 4000 сортируется дольше остальных. На обоих графиках сильных скачков не наблюдается.

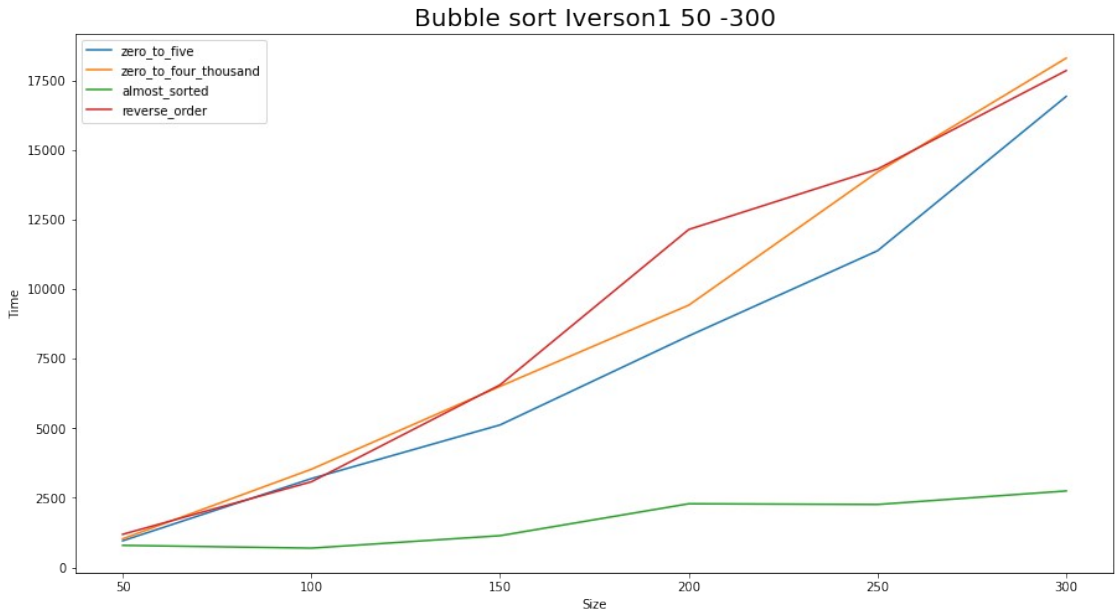
Bubble sort Iverson1

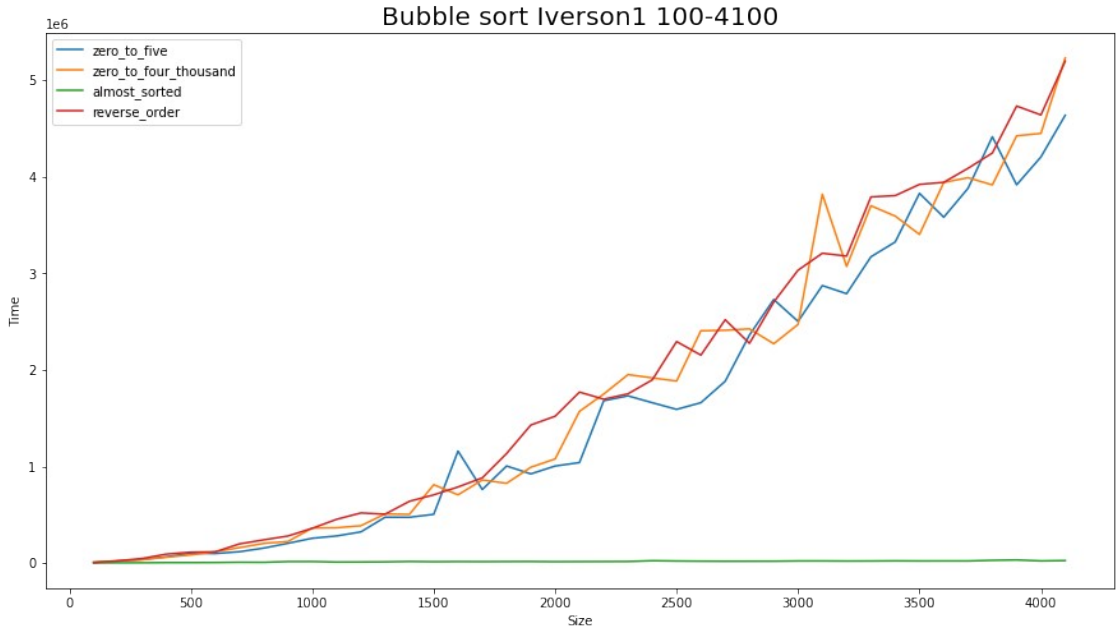
```
# Размерность 50 - 300
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(50, 301, 50):
    size.append(i)
array_type = ""
with open ("bubble_iverson1 50 - 300.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
```

```
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Bubble sort Iverson1 50 -300", fontsize= 20)
plt.legend(loc='best')
plt.show()


# Размерность 100 - 4100
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(100, 4101, 100):
    size.append(i)
array_type = ""
with open ("bubble_iverson1 100 - 4100.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
```

```
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Bubble sort Iverson1 100-4100", fontsize= 20)
plt.legend(loc='best')
plt.show()
```





Вывод к Bubble sort Iverson1:

На обоих графиках сразу бросается в глаза, что почти отсорт. массив тратит очень мало времени на сортирвку, на обычном пузрыке такого не было, как раз помогает тот факт, что если мы ни разу не меняли элементы

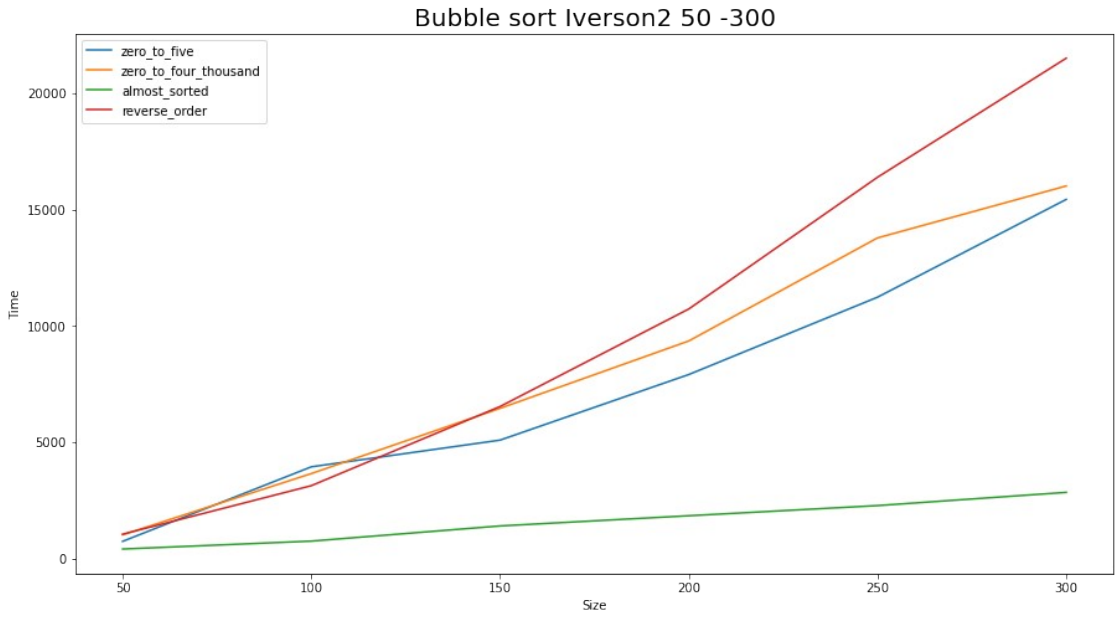
по проходу, то просто выходим из цикла. На графике 100 - 4100 наблюдаются скачки В остальном ничего сильно в глаза не бросается.

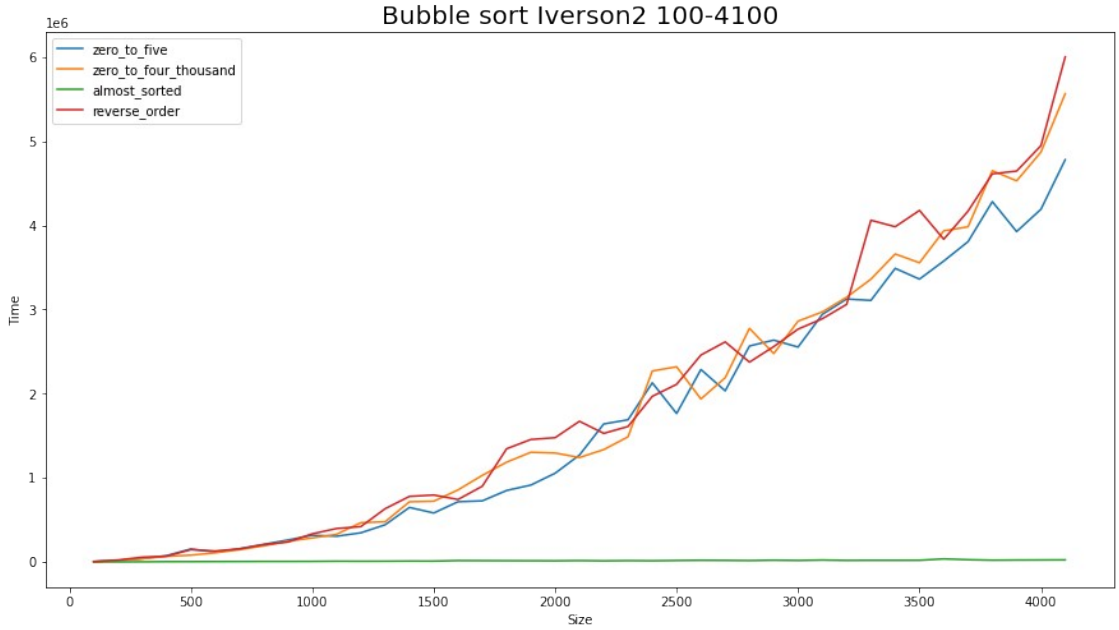
Bubble sort Iverson2

```
# Размерность 50 - 300
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(50, 301, 50):
    size.append(i)
array_type = ""
with open ("bubble_iverson2 50 - 300.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Bubble sort Iverson2 50 -300", fontsize= 20)
plt.legend(loc='best')
plt.show()


# Размерность 100 - 4100
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
```

```
reverse_order = []
for i in range(100, 4101, 100):
    size.append(i)
array_type = ""
with open ("bubble_iverson2 100 - 4100.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Bubble sort Iverson2 100-4100", fontsize= 20)
plt.legend(loc='best')
plt.show()
```





Вывод к Bubble sort Iverson2:

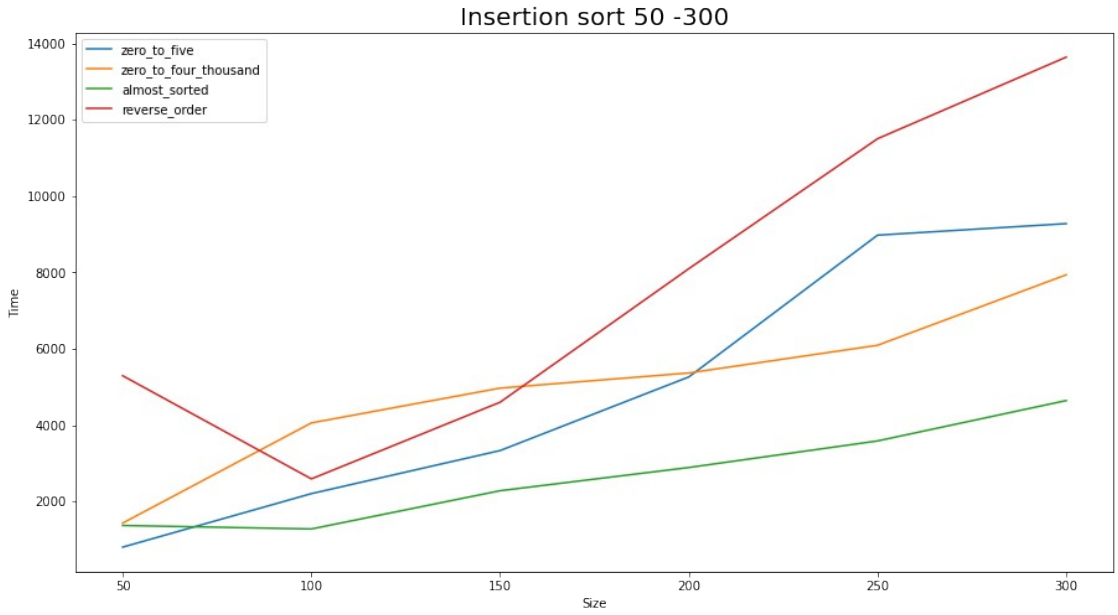
На самом деле тут по отношению к Iverson1 сильных отличий я не вижу, хотя явно стоит обратить внимание на то, что общее время сортирвки массивов по отношению к Iverson1 уменьшилось, это хорошо видно на графике 50 - 300. Также на обоих графиках почти отсортрованный массив очень долго работает. Обращу внимание на то, что массив, который осторт. в обратном порядке очень долго сортируется в обоих случаях. Оно и логичнго, как наши улучшения мало поомогают, самый большой элемент всегда приходится тащить от начала в конец.

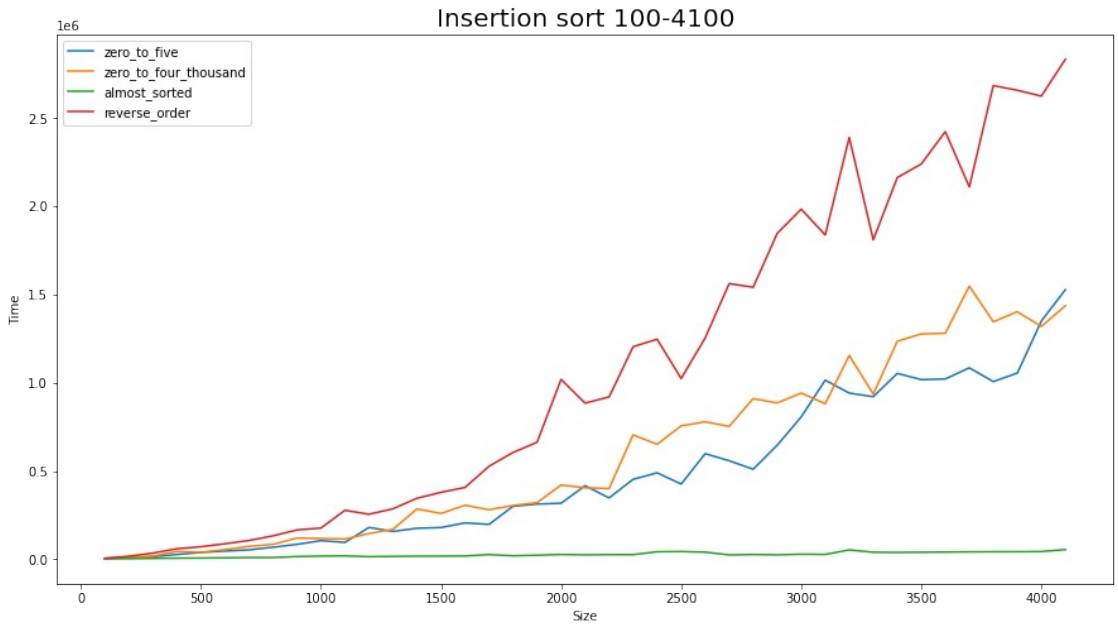
Insertion sort

```
# Размерность 50 - 300
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(50, 301, 50):
    size.append(i)
array_type = ""
with open ("insertion_sort 50 - 300.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Insertion sort 50 -300", fontsize= 20)
plt.legend(loc='best')
plt.show()


# Размерность 100 - 4100
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(100, 4101, 100):
    size.append(i)
array_type = ""
```

```
with open ("insertion_sort 100 - 4100.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Insertion sort 100-4100", fontsize= 20)
plt.legend(loc='best')
plt.show()
```





Вывод к Insertion sort:

Давайте сразу обратим внимание на то, что массив отсорт. в обратном порядке ну уж очень долго работает. Интересно, почему?) Нам приходится вечно самый проходить элементы от текущего и вниз, чтобы найти нужное место, а это долго. Остальое как по мне сильно не выделяется. Ну не считая почти отсорт., тут понятно, нам не так часто приходится двигать элементы, ведь многие уже на своих местах.

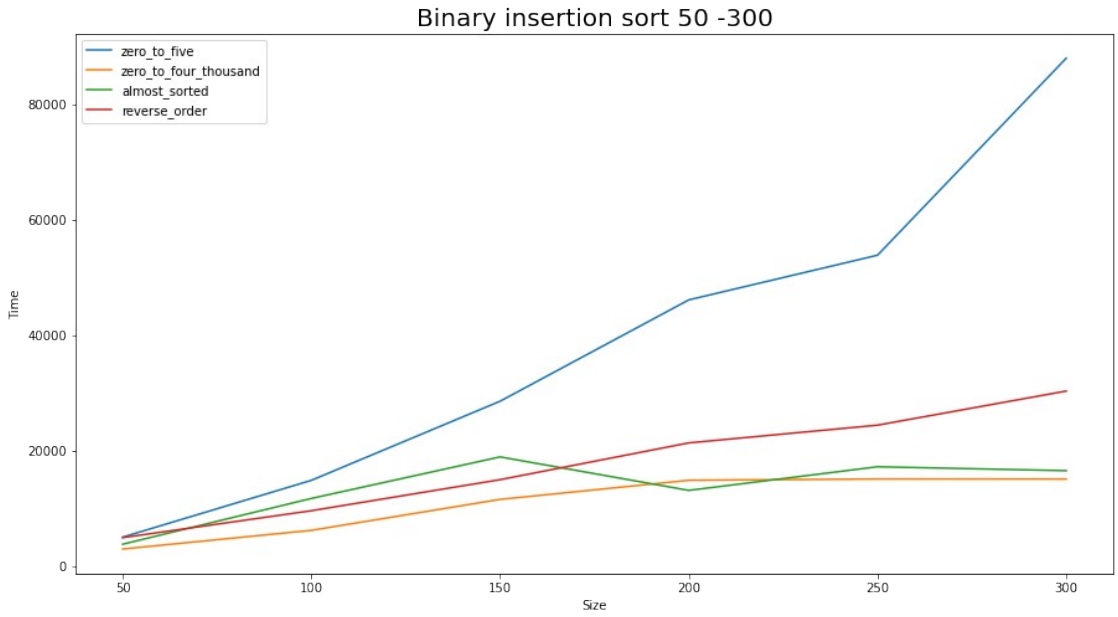
Binary insertion sort

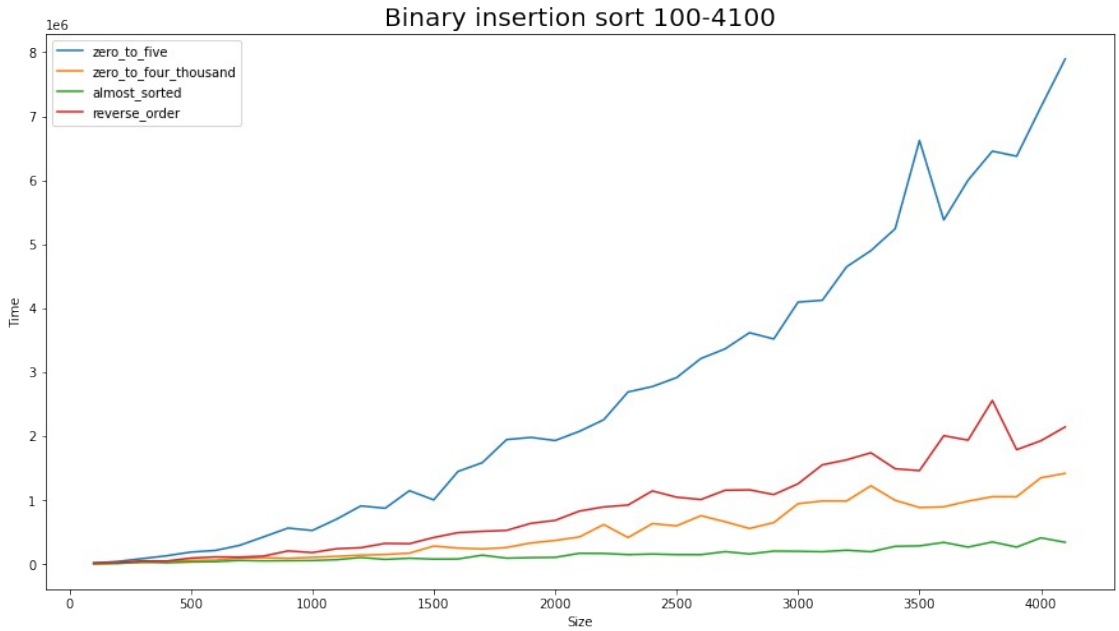
```
# Размерность 50 - 300
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(50, 301, 50):
    size.append(i)
array_type = ""
with open ("binary_insertion_sort 50 - 300.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
```

```
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Binary insertion sort 50 -300", fontsize= 20)
plt.legend(loc='best')
plt.show()


# Размерность 100 - 4100
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(100, 4101, 100):
    size.append(i)
array_type = ""
with open ("binary_insertion_sort 100 - 4100.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
```

```
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Binary insertion sort 100-4100", fontsize= 20)
plt.legend(loc='best')
plt.show()
```





Вывод к Binary insertion sort:

Вообще, забавно выходит, их всех сильно выделяется на обоих графиках zero_to_five (элементы от 0 до 5). Но вот почему так выходит я честно говоря

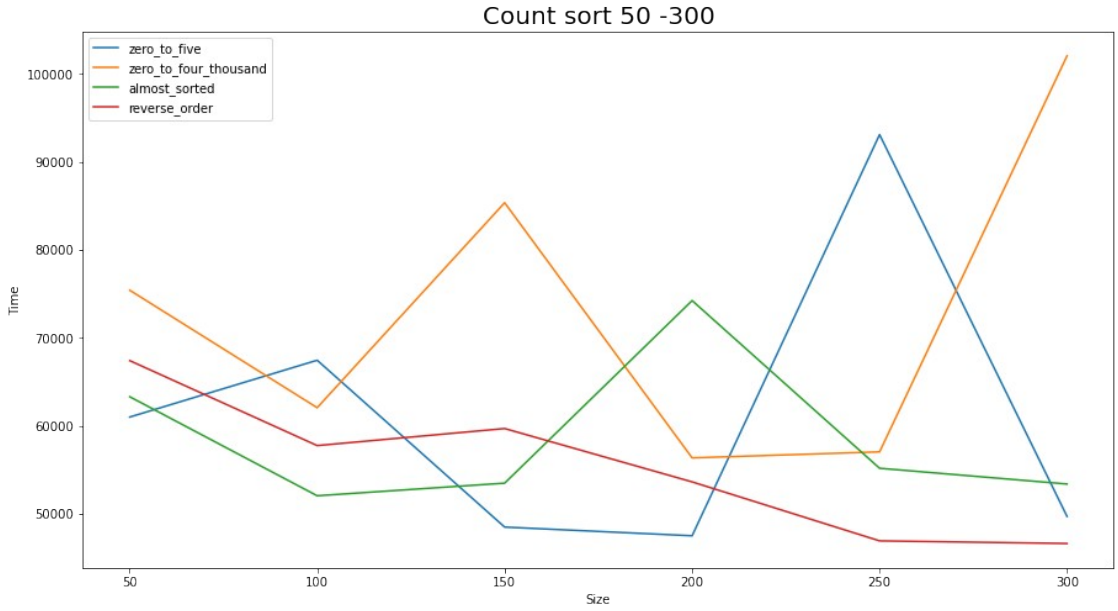
не совсем поняла. Ну и кстати, обратим внимание, что бинарные вставки так сказать в целом уменьшили время сортирвко по отношению к обычным вставкам.

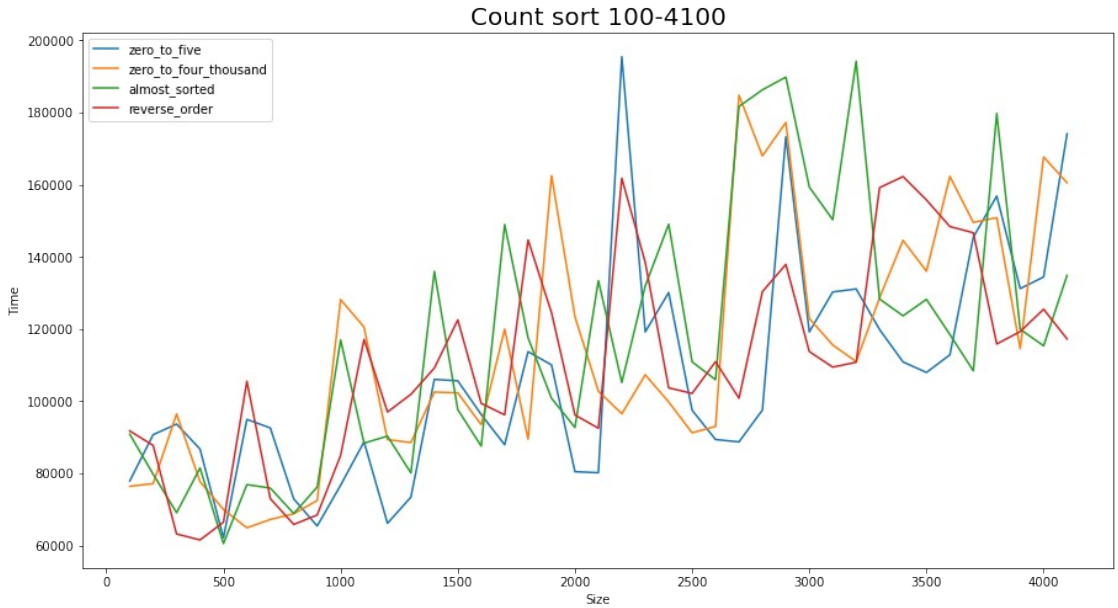
Count sort

```
# Размерность 50 - 300
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(50, 301, 50):
    size.append(i)
array_type = ""
with open ("count_sort 50 - 300.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Count sort 50 -300", fontsize= 20)
plt.legend(loc='best')
plt.show()


# Размерность 100 - 4100
size = []
zero_to_five = []
zero_to_four_thousand = []
```

```
almost_sorted = []
reverse_order = []
for i in range(100, 4101, 100):
    size.append(i)
array_type = ""
with open ("count_sort 100 - 4100.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Count sort 100-4100", fontsize= 20)
plt.legend(loc='best')
plt.show()
```





Вывод к Count sort:

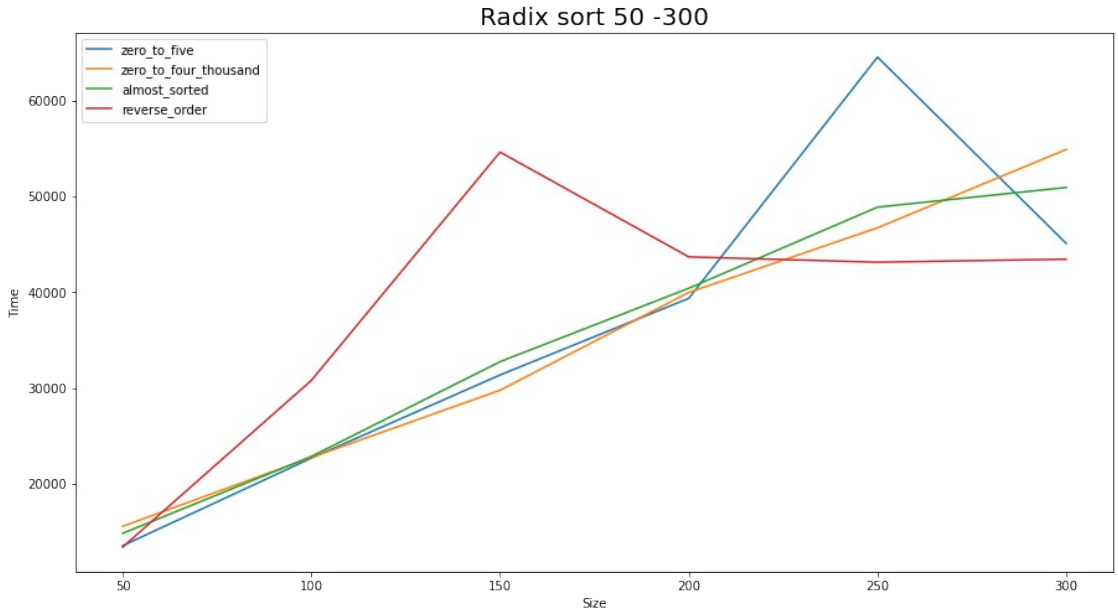
Ох... Ну тут вообще весело на обоих графиках, прям горы какие-то. Эх, покататься бы на сноуборде сейчас, ладно не об этом)) Что могу сказать, время сортировок сильно скачет на всех массивах. Почему конкретно так происходить сказать не могу. Вот для массива от 0 до 5 происходит частое обращение к одним ячейкам памяти, может, это как-то связано, но не думаю ...

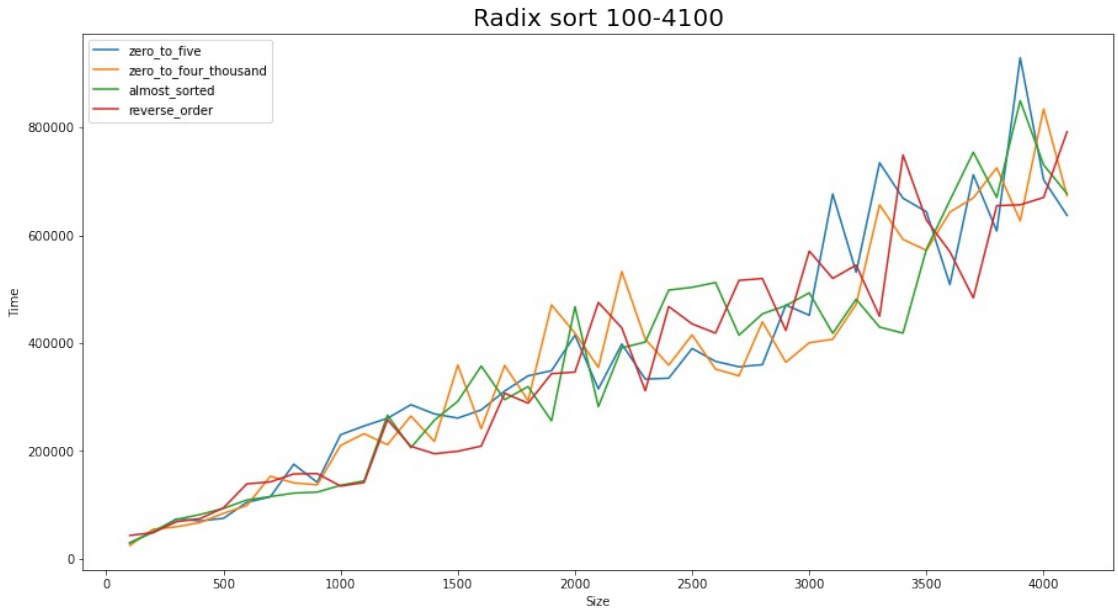
Radix sort

```
# Размерность 50 - 300
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(50, 301, 50):
    size.append(i)
array_type = ""
with open ("radix_sort 50 - 300.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Radix sort 50 -300", fontsize= 20)
plt.legend(loc='best')
plt.show()


# Размерность 100 - 4100
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(100, 4101, 100):
    size.append(i)
array_type = ""
```

```
with open ("radix_sort 100 - 4100.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Radix sort 100-4100", fontsize= 20)
plt.legend(loc='best')
plt.show()
```





Вывод к Radix sort:

Тут тоже не прям всё стабильно, но всё же лучше, чем на прошлых графиках для count. На массивах 50 - 300 особых скачков нет. На 100 - 4100 их много, но они не такие сильные

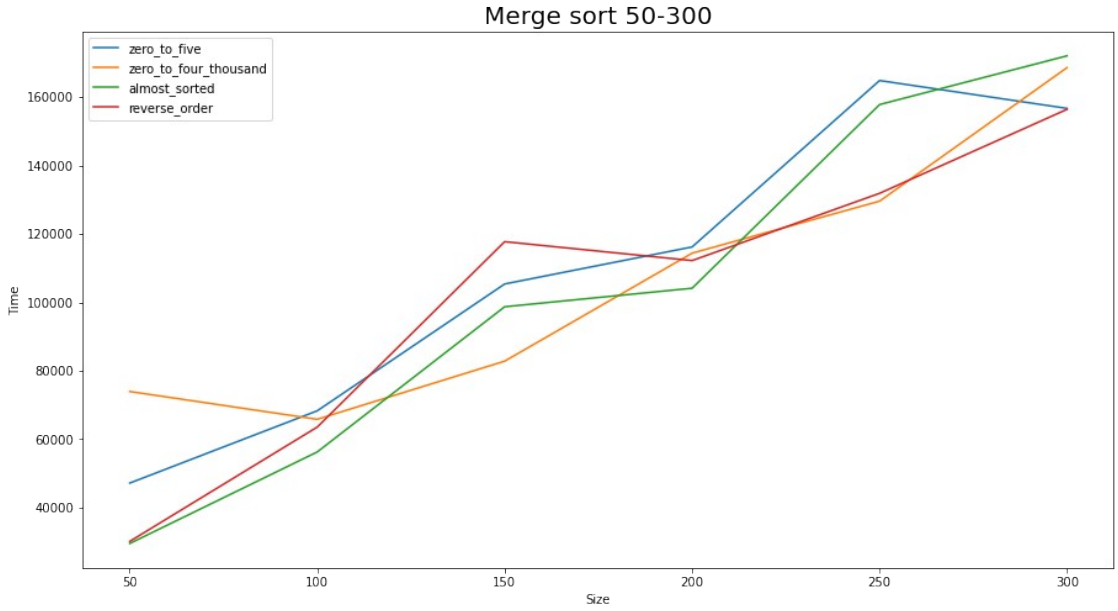
Merge sort

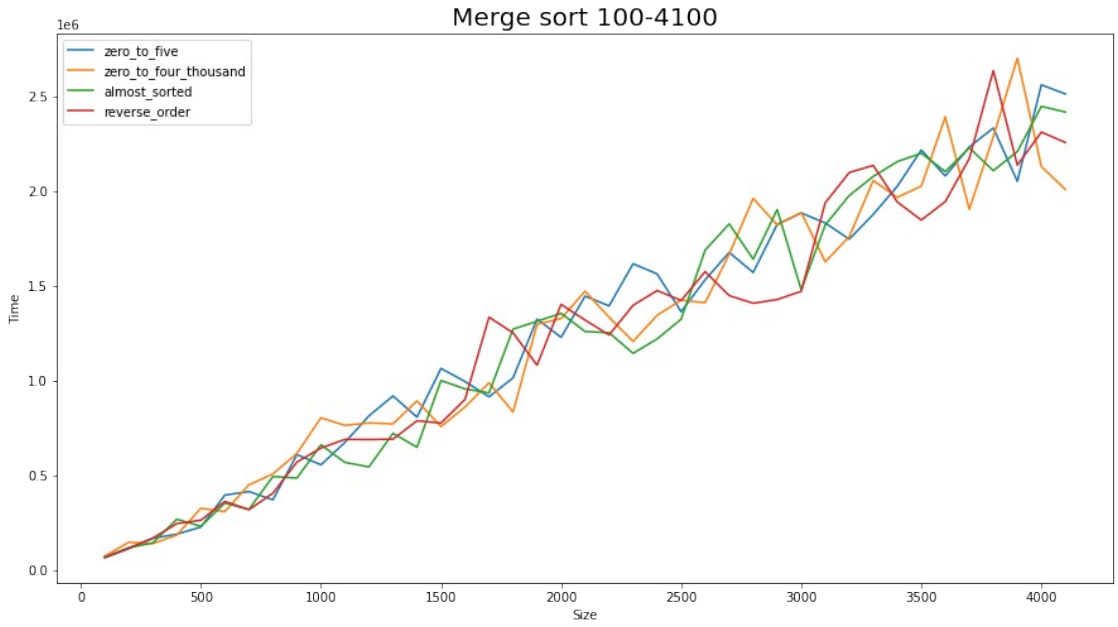
```
# Размерность 50 - 300
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(50, 301, 50):
    size.append(i)
array_type = ""
with open ("merge_sort 50 - 300.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
```

```
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Merge sort 50-300", fontsize= 20)
plt.legend(loc='best')
plt.show()


# Размерность 100 - 4100
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(100, 4101, 100):
    size.append(i)
array_type = ""
with open ("merge_sort 100 - 4100.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
```

```
ax.set_title("Merge sort 100-4100", fontsize= 20)
plt.legend(loc='upper left')
plt.show()
```





Вывод к Merge sort:

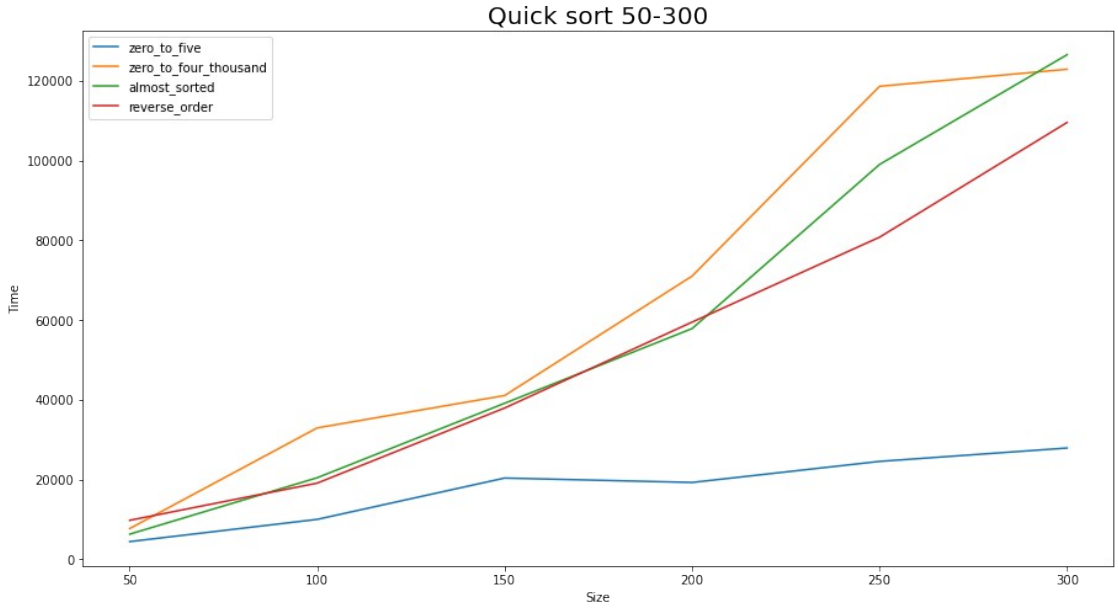
Вот мы и добрались до nlog(n). Ну, явно видно, что это не парабола, но при этом не могу сказать, что супер классное время, сортровка вставками выдавала лучше результат. Может, вопрос в том, что нужно брать больше объём данных, так как все мы знаем. что merge крут для объёмных данных. На этих графиках его крутость как-то не сильно видна (

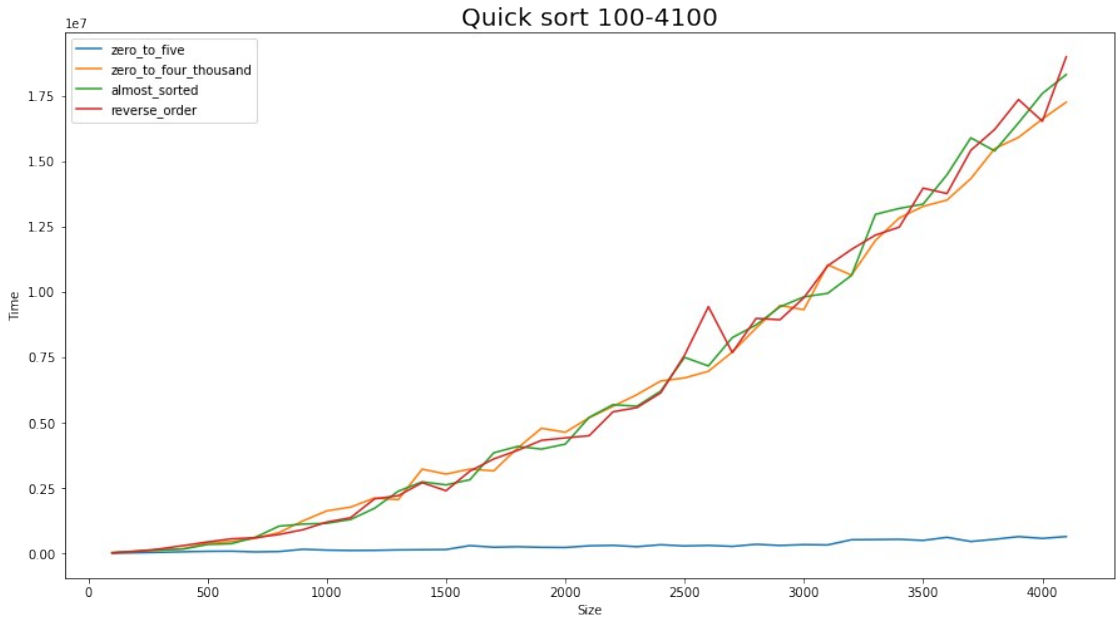
Quick sort

```
# Размерность 50 - 300
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(50, 301, 50):
    size.append(i)
array_type = ""
with open ("quick_sort 50 - 300.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Quick sort 50-300", fontsize= 20)
plt.legend(loc='best')
plt.show()


# Размерность 100 - 4100
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(100, 4101, 100):
    size.append(i)
array_type = ""
```

```
with open ("quick_sort 100 - 4100.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Quick sort 100-4100", fontsize= 20)
plt.legend(loc='upper left')
plt.show()
```





Вывод к Quick sort:

Вот тебе и раз, что-то на квадрат похоже ... Забавно, что на массиве от 0 до 5 время сортировки крайне мало. Полагаю, что здесь свою роль играет фактор маленькой разницы между элементами.

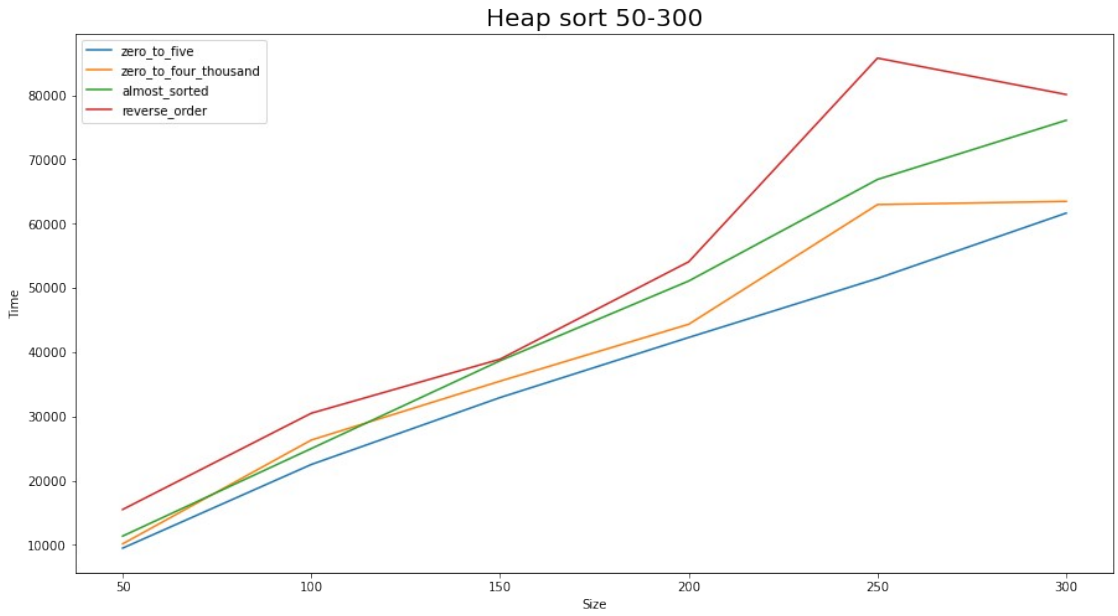
Heap sort

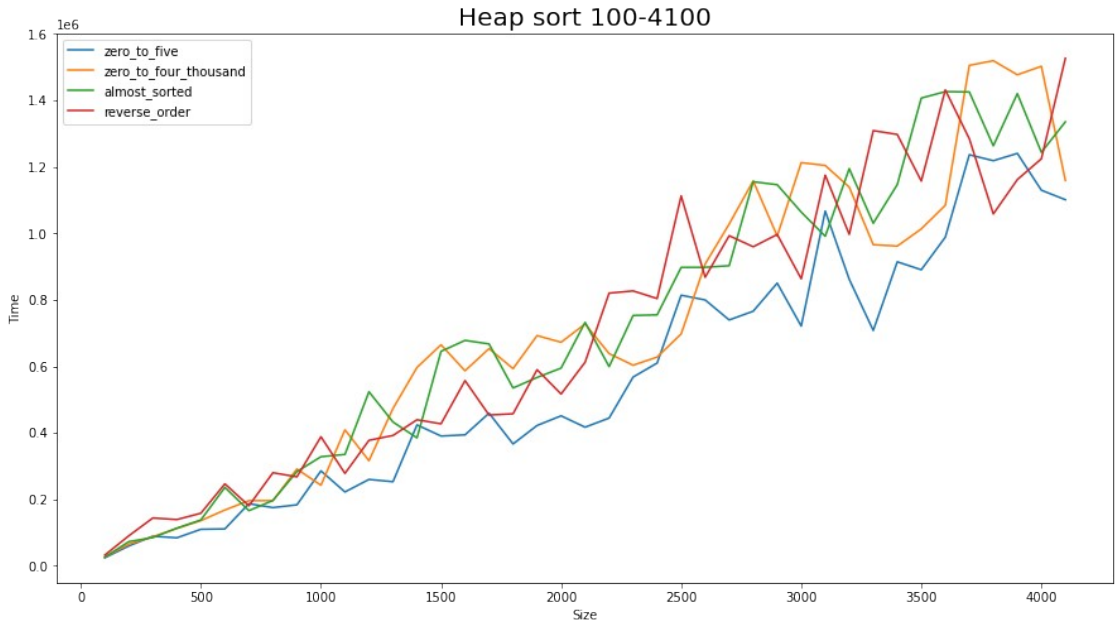
```
# Размерность 50 - 300
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(50, 301, 50):
    size.append(i)
array_type = ""
with open ("heap_sort 50 - 300.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
```

```
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Heap sort 50-300", fontsize= 20)
plt.legend(loc='best')
plt.show()


# Размерность 100 - 4100
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(100, 4101, 100):
    size.append(i)
array_type = ""
with open ("heap_sort 100 - 4100.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
```

```
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Heap sort 100-4100", fontsize= 20)
plt.legend(loc='upper left')
plt.show()
```





Вывод к Heap sort:

По сравнению с quick sort тут хоть видно, что не парабола выходит, это радует. Вот тут я бы обратила внимание на то, что на почти отсорт. массиве время крайне высокое на обоих графиках. Скорее всего связано с тем, что мы берём всегда большие элементы из начала массива и всегда вынуждены проталкивать их наверх, это влияет на время работы сильно.

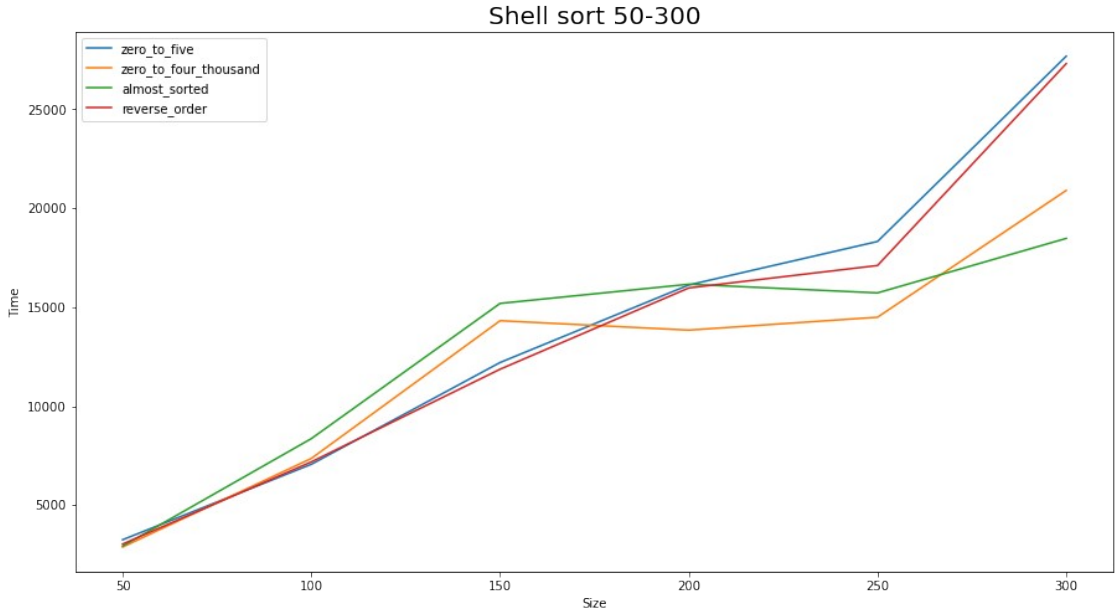
Вот от 0 - 5 массив на обоих мало времени тратит, нам не приходится большие расстояния преодолевать, чтобы элемент поставить где-то. Ну вот добавили 5, толкаем наверх, а там уже много 5, поэтому далеко идти не приходтся. Влияет маленькое расстояние между элементами.

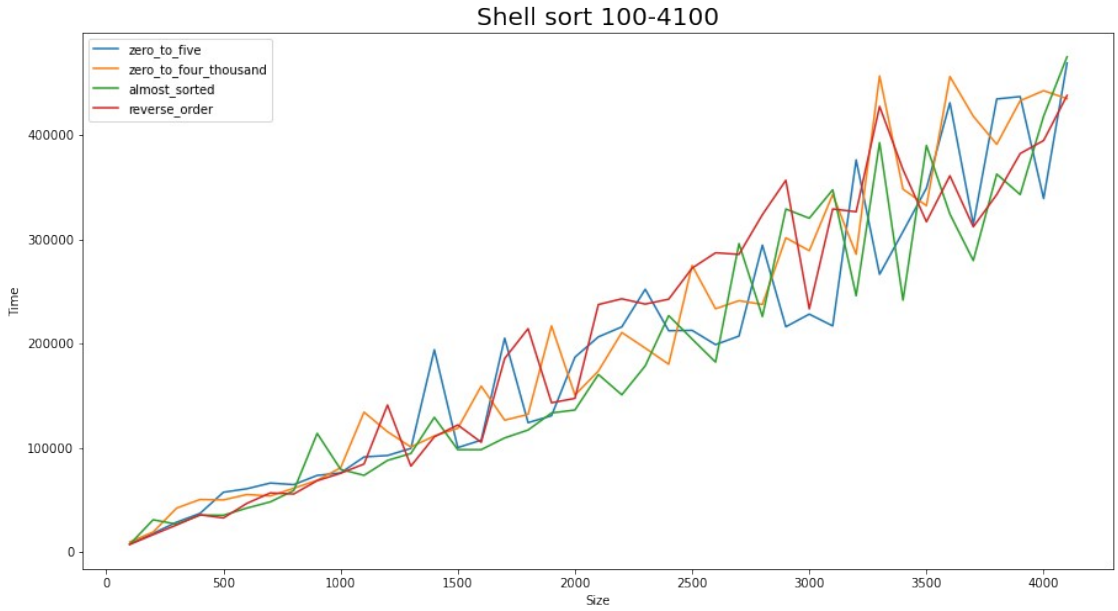
Shell sort

```
# Размерность 50 - 300
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(50, 301, 50):
    size.append(i)
array_type = ""
with open ("shell_sort 50 - 300.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Shell sort 50-300", fontsize= 20)
plt.legend(loc='best')
plt.show()


# Размерность 100 - 4100
size = []
zero_to_five = []
```

```
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(100, 4101, 100):
    size.append(i)
array_type = ""
with open ("shell_sort 100 - 4100.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Shell sort 100-4100", fontsize= 20)
plt.legend(loc='upper left')
plt.show()
```





Вывод к Shell sort:

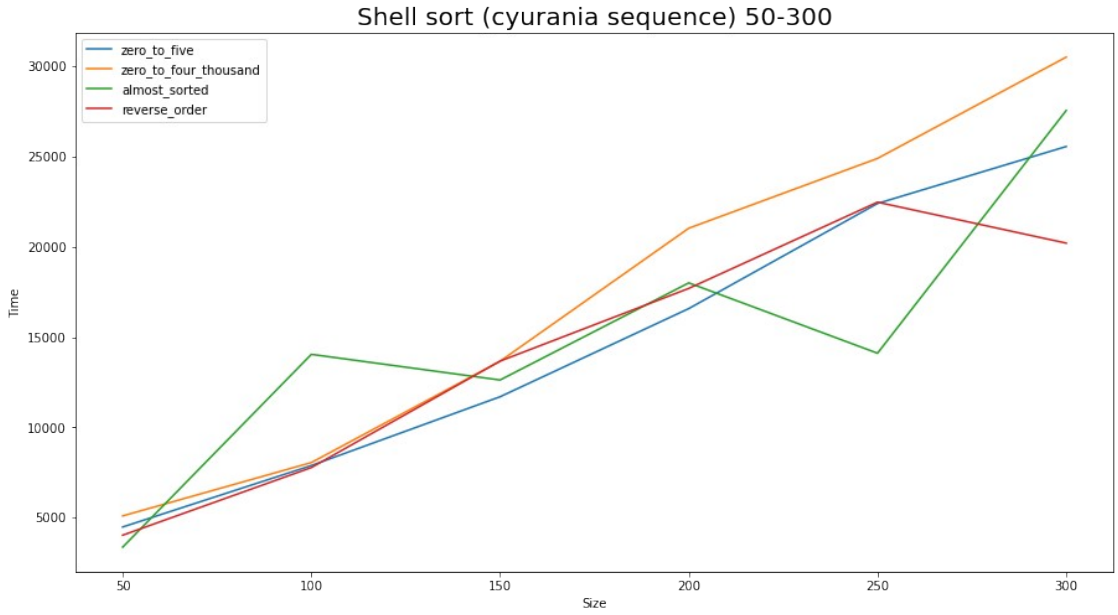
Так, по второму графику особо ничего и не скажешь, скачки крайне большие, но вот видно, что reverse_order долго работает, так как постоянно нужно менять элементы на другой порядок. Так же можно заметить, что почти отсорт. часто оказывается снизу, не всегда что-то меняем. По первому графику можно наблюдать некую стабильность в работе на почти отсорт. В диап. от 150 до 250 роста особого нет, на 0 - 4000, что интересно тоже

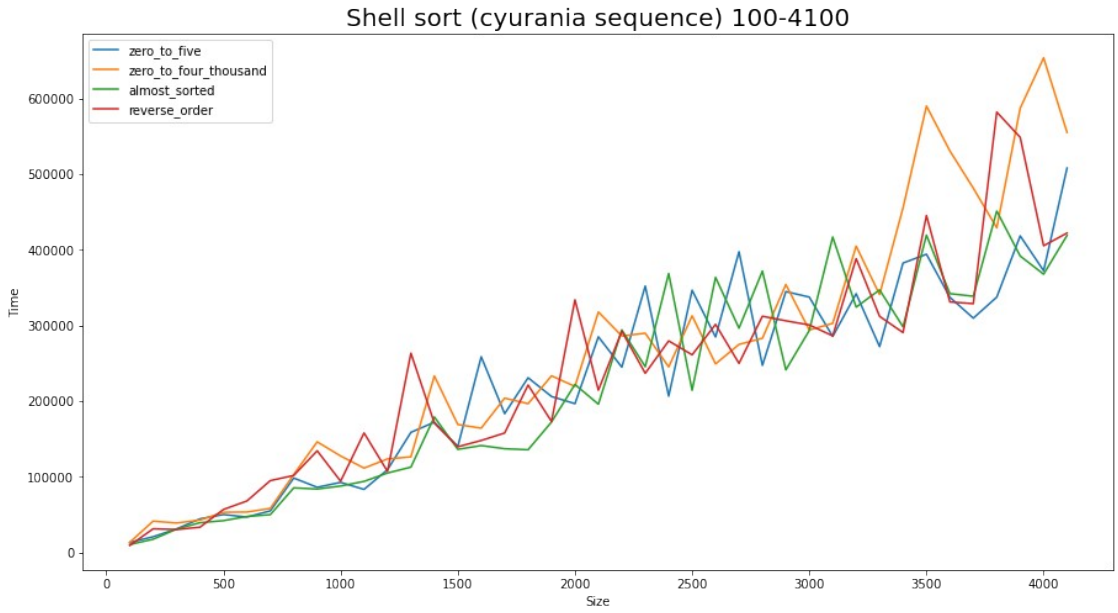
Shell sort (cyurania sequence)

```
# Размерность 50 - 300
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(50, 301, 50):
    size.append(i)
array_type = ""
with open ("cyurania_sequence 50 - 300.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Shell sort (cyurania sequence) 50-300", fontsize= 20)
plt.legend(loc='best')
plt.show()


# Размерность 100 - 4100
size = []
zero_to_five = []
zero_to_four_thousand = []
almost_sorted = []
reverse_order = []
for i in range(100, 4101, 100):
    size.append(i)
array_type = ""
```

```
with open ("cyurania_sequence 100 - 4100.txt") as f:
    for nums in f:
        if nums[0] == "R":
            array_type = nums[0:len(nums)-1]
            continue
        if array_type == "RandomZeroToFive":
            k = nums.split(" ")
            zero_to_five.append(int(k[1]))
        elif array_type == "RandomZeroToFourThousand":
            k = nums.split(" ")
            zero_to_four_thousand.append(int(k[1]))
        elif array_type == "RandomAlmostSorted":
            k = nums.split(" ")
            almost_sorted.append(int(k[1]))
        elif array_type == "RandomReverseOrder":
            k = nums.split(" ")
            reverse_order.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Time")
ax.plot(size, zero_to_five, label = 'zero_to_five')
ax.plot(size, zero_to_four_thousand, label ="zero_to_four_thousand")
ax.plot(size, almost_sorted, label = "almost_sorted")
ax.plot(size, reverse_order, label = "reverse_order")
ax.set_title("Shell sort (cyurania sequence) 100-4100", fontsize= 20)
plt.legend(loc='upper left')
plt.show()
```





Вывод к Shell sort (cyurania sequence):

Тут интересно будет сравнить с прошлой сортировкой. По первому графику хорошо видно, что 0 - 4000 теперь работает медленне всего, хотя на прошлой было иначе. По второму графику тоже можно наблюдать такую зависимость, особенно на больших данных. Вот если в обычной сортировке Shell можно было на 2 графике заметить какие-то стабильности, то тут с этим как-то похуже