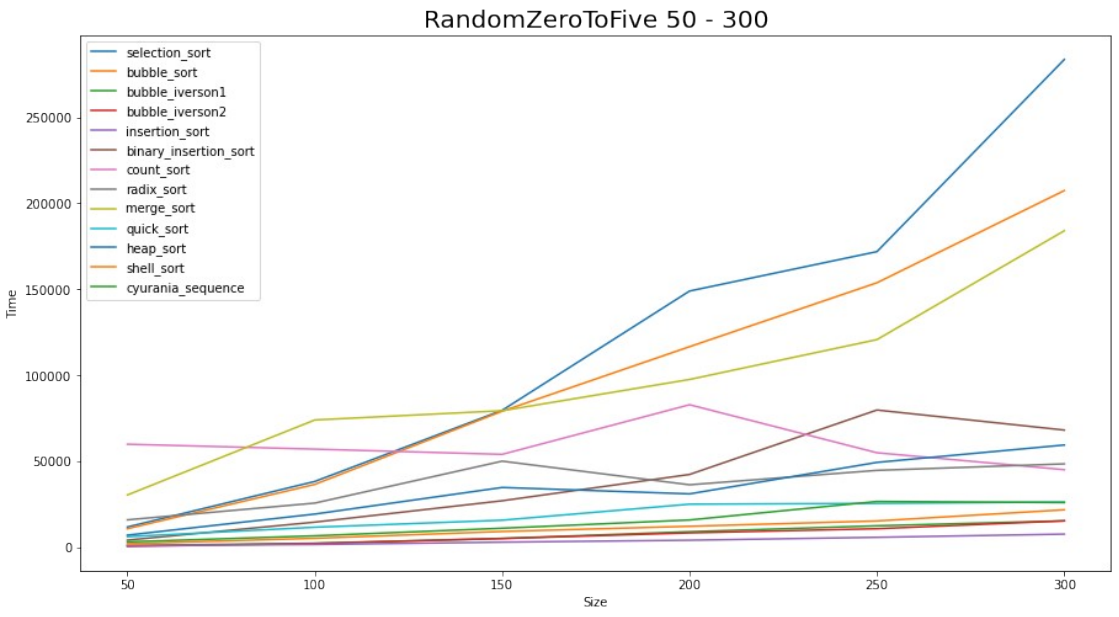
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

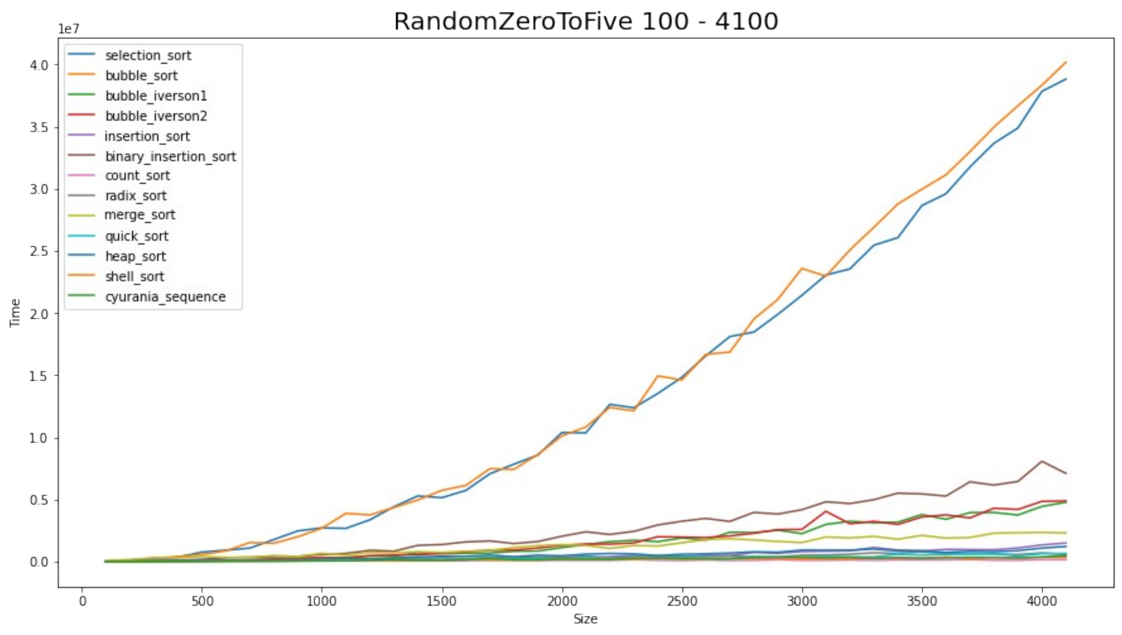
RandomZeroToFive

```
# Размерность 50 - 300
sort_type = ""
size = []
for i in range(50, 301, 50):
    size.append(i)
selection_sort = []
bubble_sort = []
bubble_iverson1 = []
bubble_iverson2 = []
insertion_sort = []
binary_insertion_sort = []
count_sort = []
radix_sort = []
merge_sort = []
quick_sort = []
heap_sort = []
shell_sort = []
cyurania_sequence = []
with open("random_zero_to_five 50 - 300.txt") as f:
    for nums in f:
        if nums [0] == "N":
            sort_type = nums[1:len(nums) - 1]
            continue
        elif sort_type == "selection_sort":
            k = nums.split(" ")
            selection_sort.append(int(k[1]))
        elif sort_type == "bubble_sort":
            k = nums.split(" ")
            bubble_sort.append(int(k[1]))
        elif sort_type == "bubble_iverson1":
            k = nums.split(" ")
            bubble_iverson1.append(int(k[1]))
        elif sort_type == "bubble_iverson2":
            k = nums.split(" ")
            bubble_iverson2.append(int(k[1]))
        elif sort_type == "insertion_sort":
            k = nums.split(" ")
            insertion_sort.append(int(k[1]))
        elif sort_type == "binary_insertion_sort":
            k = nums.split(" ")
            binary_insertion_sort.append(int(k[1]))
        elif sort_type == "count_sort":
            k = nums.split(" ")
            count_sort.append(int(k[1]))
        elif sort_type == "radix_sort":
            k = nums.split(" ")
            radix_sort.append(int(k[1]))
        elif sort_type == "merge_sort":
            k = nums.split(" ")
            merge_sort.append(int(k[1]))
        elif sort_type == "quick_sort":
            k = nums.split(" ")
            quick_sort.append(int(k[1]))
        elif sort_type == "heap_sort":
            k = nums.split(" ")
            heap_sort.append(int(k[1]))
        elif sort_type == "shell_sort":
            k = nums.split(" ")
            shell_sort.append(int(k[1]))
        elif sort_type == "cyurania_sequence":
            k = nums.split(" ")
            cyurania_sequence.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, selection_sort, label = 'selection_sort')
ax.plot(size, bubble_sort, label = 'bubble_sort')
ax.plot(size, bubble_iverson1, label = 'bubble_iverson1')
ax.plot(size, bubble_iverson2, label = 'bubble_iverson2')
ax.plot(size, insertion_sort, label = 'insertion_sort')
ax.plot(size, binary_insertion_sort, label = 'binary_insertion_sort')
ax.plot(size, count_sort, label = 'count_sort')
ax.plot(size, radix_sort, label = 'radix_sort')
ax.plot(size, merge_sort , label = 'merge_sort ')
ax.plot(size, quick_sort, label = 'quick_sort')
ax.plot(size, heap_sort, label = 'heap_sort')
ax.plot(size, shell_sort, label = 'shell_sort')
ax.plot(size, cyurania_sequence, label = 'cyurania_sequence')
ax.set_title("RandomZeroToFive 50 - 300", fontsize= 20)
plt.legend(loc='best')
plt.show()

# Размерность 100 - 4100
size = []
for i in range(100, 4101, 100):
    size.append(i)
selection_sort = []
bubble_sort = []
bubble_iverson1 = []
bubble_iverson2 = []
insertion_sort = []
```

```python
binary_insertion_sort = []
count_sort = []
radix_sort = []
merge_sort = []
quick_sort = []
heap_sort = []
shell_sort = []
cyurania_sequence = []
with open("random_zero_to_five 100 - 4100.txt") as f:
    for nums in f:
        if nums [0] == "N":
            sort_type = nums[1:len(nums) - 1]
            continue
        elif sort_type == "selection_sort":
            k = nums.split(" ")
            selection_sort.append(int(k[1]))
        elif sort_type == "bubble_sort":
            k = nums.split(" ")
            bubble_sort.append(int(k[1]))
        elif sort_type == "bubble_iverson1":
            k = nums.split(" ")
            bubble_iverson1.append(int(k[1]))
        elif sort_type == "bubble_iverson2":
            k = nums.split(" ")
            bubble_iverson2.append(int(k[1]))
        elif sort_type == "insertion_sort":
            k = nums.split(" ")
            insertion_sort.append(int(k[1]))
        elif sort_type == "binary_insertion_sort":
            k = nums.split(" ")
            binary_insertion_sort.append(int(k[1]))
        elif sort_type == "count_sort":
            k = nums.split(" ")
            count_sort.append(int(k[1]))
        elif sort_type == "radix_sort":
            k = nums.split(" ")
            radix_sort.append(int(k[1]))
        elif sort_type == "merge_sort":
            k = nums.split(" ")
            merge_sort.append(int(k[1]))
        elif sort_type == "quick_sort":
            k = nums.split(" ")
            quick_sort.append(int(k[1]))
        elif sort_type == "heap_sort":
            k = nums.split(" ")
            heap_sort.append(int(k[1]))
        elif sort_type == "shell_sort":
            k = nums.split(" ")
            shell_sort.append(int(k[1]))
        elif sort_type == "cyurania_sequence":
            k = nums.split(" ")
            cyurania_sequence.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, selection_sort, label = 'selection_sort')
ax.plot(size, bubble_sort, label = 'bubble_sort')
ax.plot(size, bubble_iverson1, label = 'bubble_iverson1')
ax.plot(size, bubble_iverson2, label = 'bubble_iverson2')
ax.plot(size, insertion_sort, label = 'insertion_sort')
ax.plot(size, binary_insertion_sort, label = 'binary_insertion_sort')
ax.plot(size, count_sort, label = 'count_sort')
ax.plot(size, radix_sort, label = 'radix_sort')
ax.plot(size, merge_sort , label = 'merge_sort ')
ax.plot(size, quick_sort, label = 'quick_sort')
ax.plot(size, heap_sort, label = 'heap_sort')
ax.plot(size, shell_sort, label = 'shell_sort')
ax.plot(size, cyurania_sequence, label = 'cyurania_sequence')
ax.set_title("RandomZeroToFive 100 - 4100", fontsize= 20)
plt.legend(loc='best')
plt.show()
```





Вывод к массиву чисел от 0 до 5:

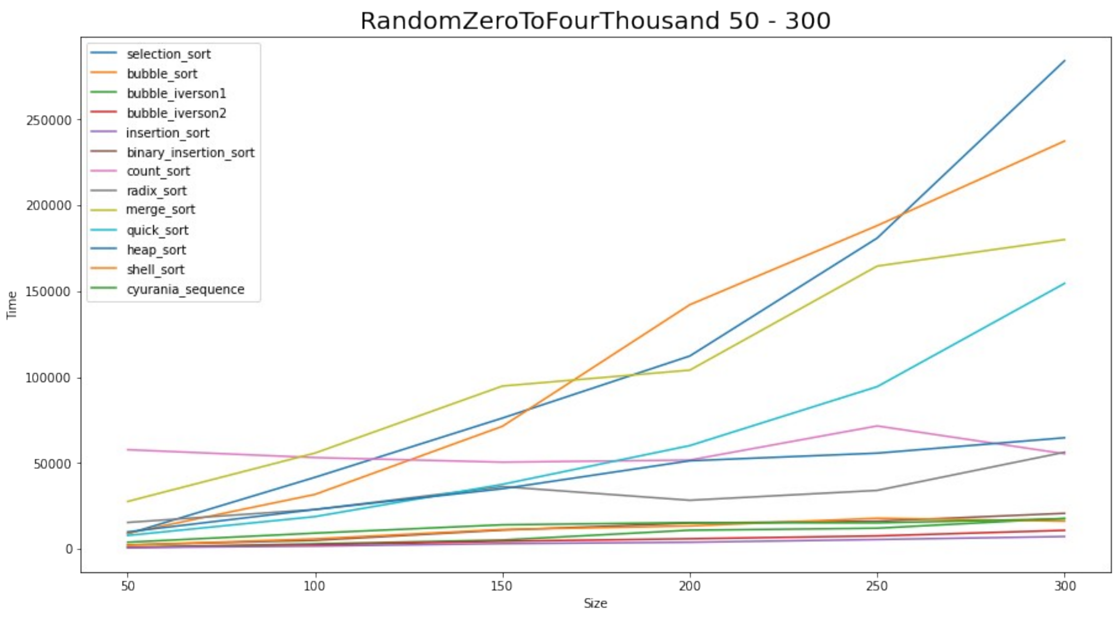
В целом на обоих графиках похожая тенденция, сортировка выбором вставками и пузырьком крайне медленные, ну как минимум видно, что асимптотика квадрата верна. Все остальные как-то внизу спрятались)

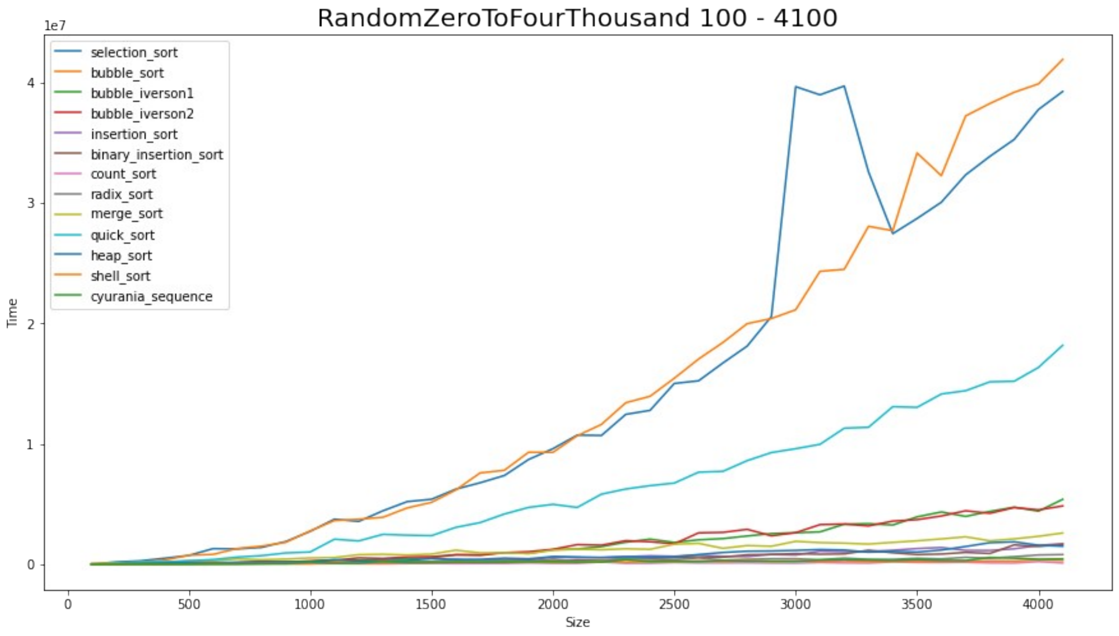
RandomZeroToFourThousand

```
# Размерность 50 - 300
sort_type = ""
size = []
for i in range(50, 301, 50):
    size.append(i)
selection_sort = []
bubble_sort = []
bubble_iverson1 = []
bubble_iverson2 = []
insertion_sort = []
binary_insertion_sort = []
count_sort = []
radix_sort = []
merge_sort = []
quick_sort = []
heap_sort = []
shell_sort = []
cyurania_sequence = []
with open("random_zero_to_four_thousand 50 - 300.txt") as f:
    for nums in f:
        if nums [0] == "N":
            sort_type = nums[1:len(nums) - 1]
```

```
                continue
            elif sort_type == "selection_sort":
                k = nums.split(" ")
                selection_sort.append(int(k[1]))
            elif sort_type == "bubble_sort":
                k = nums.split(" ")
                bubble_sort.append(int(k[1]))
            elif sort_type == "bubble_iverson1":
                k = nums.split(" ")
                bubble_iverson1.append(int(k[1]))
            elif sort_type == "bubble_iverson2":
                k = nums.split(" ")
                bubble_iverson2.append(int(k[1]))
            elif sort_type == "insertion_sort":
                k = nums.split(" ")
                insertion_sort.append(int(k[1]))
            elif sort_type == "binary_insertion_sort":
                k = nums.split(" ")
                binary_insertion_sort.append(int(k[1]))
            elif sort_type == "count_sort":
                k = nums.split(" ")
                count_sort.append(int(k[1]))
            elif sort_type == "radix_sort":
                k = nums.split(" ")
                radix_sort.append(int(k[1]))
            elif sort_type == "merge_sort":
                k = nums.split(" ")
                merge_sort.append(int(k[1]))
            elif sort_type == "quick_sort":
                k = nums.split(" ")
                quick_sort.append(int(k[1]))
            elif sort_type == "heap_sort":
                k = nums.split(" ")
                heap_sort.append(int(k[1]))
            elif sort_type == "shell_sort":
                k = nums.split(" ")
                shell_sort.append(int(k[1]))
            elif sort_type == "cyurania_sequence":
                k = nums.split(" ")
                cyurania_sequence.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, selection_sort, label = 'selection_sort')
ax.plot(size, bubble_sort, label = 'bubble_sort')
ax.plot(size, bubble_iverson1, label = 'bubble_iverson1')
ax.plot(size, bubble_iverson2, label = 'bubble_iverson2')
ax.plot(size, insertion_sort, label = 'insertion_sort')
ax.plot(size, binary_insertion_sort, label = 'binary_insertion_sort')
ax.plot(size, count_sort, label = 'count_sort')
ax.plot(size, radix_sort, label = 'radix_sort')
ax.plot(size, merge_sort , label = 'merge_sort ')
ax.plot(size, quick_sort, label = 'quick_sort')
ax.plot(size, heap_sort, label = 'heap_sort')
ax.plot(size, shell_sort, label = 'shell_sort')
ax.plot(size, cyurania_sequence, label = 'cyurania_sequence')
ax.set_title("RandomZeroToFourThousand 50 - 300", fontsize= 20)
plt.legend(loc='best')
plt.show()

# Размерность 100 - 4100
size = []
for i in range(100, 4101, 100):
    size.append(i)
selection_sort = []
bubble_sort = []
bubble_iverson1 = []
bubble_iverson2 = []
insertion_sort = []
binary_insertion_sort = []
count_sort = []
radix_sort = []
merge_sort = []
quick_sort = []
heap_sort = []
shell_sort = []
cyurania_sequence = []
with open("random_zero_to_four_thousand 100 - 4100.txt") as f:
    for nums in f:
        if nums [0] == "N":
            sort_type = nums[1:len(nums) - 1]
            continue
        elif sort_type == "selection_sort":
            k = nums.split(" ")
            selection_sort.append(int(k[1]))
        elif sort_type == "bubble_sort":
            k = nums.split(" ")
            bubble_sort.append(int(k[1]))
        elif sort_type == "bubble_iverson1":
            k = nums.split(" ")
            bubble_iverson1.append(int(k[1]))
        elif sort_type == "bubble_iverson2":
            k = nums.split(" ")
            bubble_iverson2.append(int(k[1]))
        elif sort_type == "insertion_sort":
            k = nums.split(" ")
            insertion_sort.append(int(k[1]))
        elif sort_type == "binary_insertion_sort":
            k = nums.split(" ")
            binary_insertion_sort.append(int(k[1]))
        elif sort_type == "count_sort":
            k = nums.split(" ")
            count_sort.append(int(k[1]))
        elif sort_type == "radix_sort":
            k = nums.split(" ")
            radix_sort.append(int(k[1]))
        elif sort_type == "merge_sort":
            k = nums.split(" ")
            merge_sort.append(int(k[1]))
        elif sort_type == "quick_sort":
            k = nums.split(" ")
            quick_sort.append(int(k[1]))
        elif sort_type == "heap_sort":
            k = nums.split(" ")
            heap_sort.append(int(k[1]))
        elif sort_type == "shell_sort":
            k = nums.split(" ")
            shell_sort.append(int(k[1]))
        elif sort_type == "cyurania_sequence":
            k = nums.split(" ")
            cyurania_sequence.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, selection_sort, label = 'selection_sort')
ax.plot(size, bubble_sort, label = 'bubble_sort')
ax.plot(size, bubble_iverson1, label = 'bubble_iverson1')
ax.plot(size, bubble_iverson2, label = 'bubble_iverson2')
ax.plot(size, insertion_sort, label = 'insertion_sort')
ax.plot(size, binary_insertion_sort, label = 'binary_insertion_sort')
ax.plot(size, count_sort, label = 'count_sort')
ax.plot(size, radix_sort, label = 'radix_sort')
ax.plot(size, merge_sort , label = 'merge_sort ')
ax.plot(size, quick_sort, label = 'quick_sort')
ax.plot(size, heap_sort, label = 'heap_sort')
ax.plot(size, shell_sort, label = 'shell_sort')
ax.plot(size, cyurania_sequence, label = 'cyurania_sequence')
ax.set_title("RandomZeroToFourThousand 100 - 4100", fontsize= 20)
plt.legend(loc='best')
plt.show()
```





Массив чисел от 0 до 4000

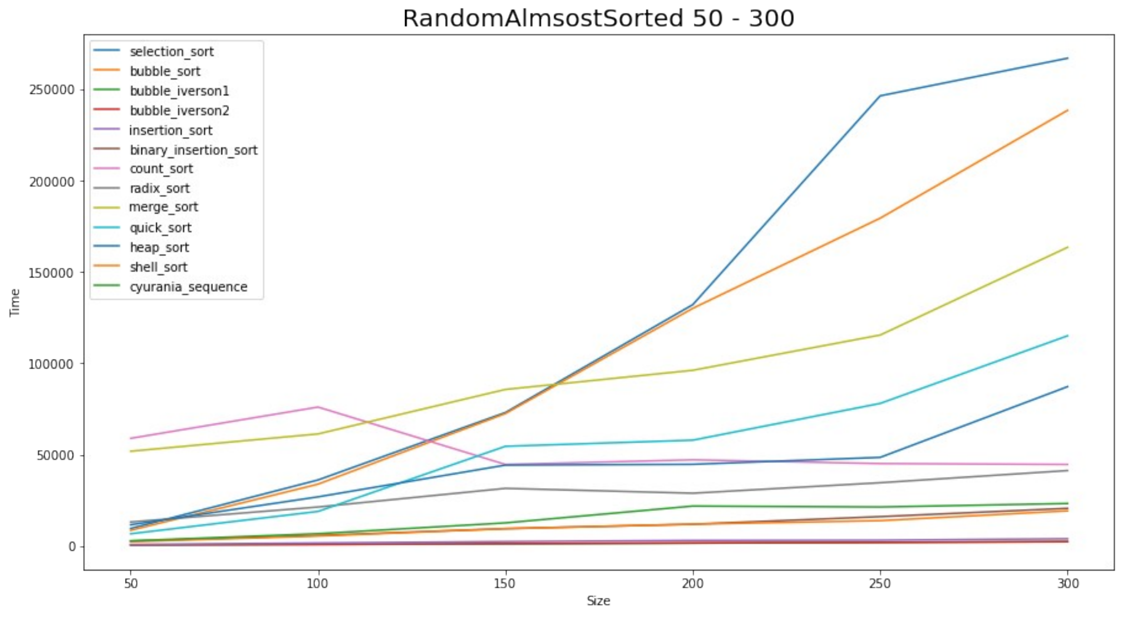
Ну что ж, пузырёк и сортировка выбором опять проигрывают. Теперь к их проигр. ещё присоединяется quick sort, а я думала, что ты быстрая ... Обращу внимание, что merge с nlog(n) далеко не быстрее всех на обоих графиках. Что интересно лично для меня, сортировка подсчётом себя хорошо показала. Ну что, перейдём к последним более интересным случаям.

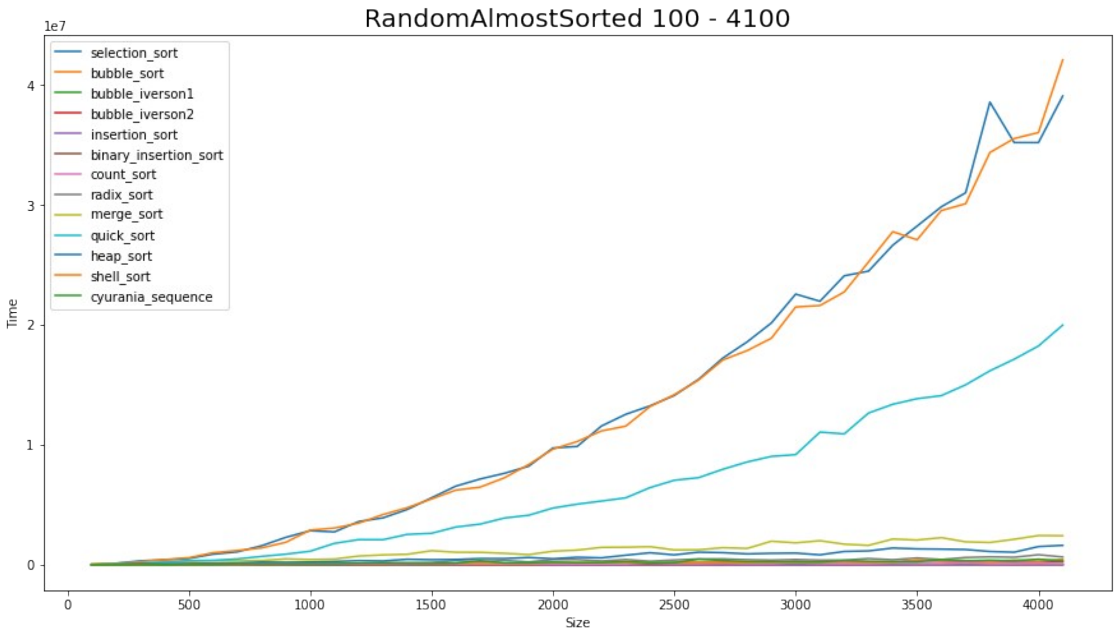
RandomAlmostSorted

```
# Размерность 50 - 300
sort_type = ""
size = []
for i in range(50, 301, 50):
    size.append(i)
selection_sort = []
bubble_sort = []
bubble_iverson1 = []
bubble_iverson2 = []
insertion_sort = []
binary_insertion_sort = []
count_sort = []
radix_sort = []
merge_sort = []
quick_sort = []
heap_sort = []
shell_sort = []
cyurania_sequence = []
with open("random_almost_sorted 50 - 300.txt") as f:
    for nums in f:
        if nums [0] == "N":
            sort_type = nums[1:len(nums) - 1]
            continue
        elif sort_type == "selection_sort":
            k = nums.split(" ")
            selection_sort.append(int(k[1]))
        elif sort_type == "bubble_sort":
            k = nums.split(" ")
            bubble_sort.append(int(k[1]))
        elif sort_type == "bubble_iverson1":
            k = nums.split(" ")
            bubble_iverson1.append(int(k[1]))
        elif sort_type == "bubble_iverson2":
            k = nums.split(" ")
            bubble_iverson2.append(int(k[1]))
        elif sort_type == "insertion_sort":
            k = nums.split(" ")
            insertion_sort.append(int(k[1]))
        elif sort_type == "binary_insertion_sort":
            k = nums.split(" ")
            binary_insertion_sort.append(int(k[1]))
        elif sort_type == "count_sort":
            k = nums.split(" ")
            count_sort.append(int(k[1]))
        elif sort_type == "radix_sort":
            k = nums.split(" ")
            radix_sort.append(int(k[1]))
        elif sort_type == "merge_sort":
            k = nums.split(" ")
            merge_sort.append(int(k[1]))
        elif sort_type == "quick_sort":
            k = nums.split(" ")
            quick_sort.append(int(k[1]))
        elif sort_type == "heap_sort":
            k = nums.split(" ")
            heap_sort.append(int(k[1]))
        elif sort_type == "shell_sort":
            k = nums.split(" ")
            shell_sort.append(int(k[1]))
        elif sort_type == "cyurania_sequence":
            k = nums.split(" ")
            cyurania_sequence.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, selection_sort, label = 'selection_sort')
ax.plot(size, bubble_sort, label = 'bubble_sort')
ax.plot(size, bubble_iverson1, label = 'bubble_iverson1')
ax.plot(size, bubble_iverson2, label = 'bubble_iverson2')
ax.plot(size, insertion_sort, label = 'insertion_sort')
ax.plot(size, binary_insertion_sort, label = 'binary_insertion_sort')
ax.plot(size, count_sort, label = 'count_sort')
ax.plot(size, radix_sort, label = 'radix_sort')
ax.plot(size, merge_sort , label = 'merge_sort ')
ax.plot(size, quick_sort, label = 'quick_sort')
ax.plot(size, heap_sort, label = 'heap_sort')
ax.plot(size, shell_sort, label = 'shell_sort')
ax.plot(size, cyurania_sequence, label = 'cyurania_sequence')
ax.set_title("RandomAlmsostSorted 50 - 300", fontsize= 20)
plt.legend(loc='best')
plt.show()

# Размерность 100 - 4100
size = []
for i in range(100, 4101, 100):
    size.append(i)
selection_sort = []
bubble_sort = []
bubble_iverson1 = []
bubble_iverson2 = []
insertion_sort = []
binary_insertion_sort = []
count_sort = []
radix_sort = []
merge_sort = []
quick_sort = []
heap_sort = []
shell_sort = []
```

```python
cyurania_sequence = []
with open("random_almost_sorted 100 - 4100.txt") as f:
    for nums in f:
        if nums [0] == "N":
            sort_type = nums[1:len(nums) - 1]
            continue
        elif sort_type == "selection_sort":
            k = nums.split(" ")
            selection_sort.append(int(k[1]))
        elif sort_type == "bubble_sort":
            k = nums.split(" ")
            bubble_sort.append(int(k[1]))
        elif sort_type == "bubble_iverson1":
            k = nums.split(" ")
            bubble_iverson1.append(int(k[1]))
        elif sort_type == "bubble_iverson2":
            k = nums.split(" ")
            bubble_iverson2.append(int(k[1]))
        elif sort_type == "insertion_sort":
            k = nums.split(" ")
            insertion_sort.append(int(k[1]))
        elif sort_type == "binary_insertion_sort":
            k = nums.split(" ")
            binary_insertion_sort.append(int(k[1]))
        elif sort_type == "count_sort":
            k = nums.split(" ")
            count_sort.append(int(k[1]))
        elif sort_type == "radix_sort":
            k = nums.split(" ")
            radix_sort.append(int(k[1]))
        elif sort_type == "merge_sort":
            k = nums.split(" ")
            merge_sort.append(int(k[1]))
        elif sort_type == "quick_sort":
            k = nums.split(" ")
            quick_sort.append(int(k[1]))
        elif sort_type == "heap_sort":
            k = nums.split(" ")
            heap_sort.append(int(k[1]))
        elif sort_type == "shell_sort":
            k = nums.split(" ")
            shell_sort.append(int(k[1]))
        elif sort_type == "cyurania_sequence":
            k = nums.split(" ")
            cyurania_sequence.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, selection_sort, label = 'selection_sort')
ax.plot(size, bubble_sort, label = 'bubble_sort')
ax.plot(size, bubble_iverson1, label = 'bubble_iverson1')
ax.plot(size, bubble_iverson2, label = 'bubble_iverson2')
ax.plot(size, insertion_sort, label = 'insertion_sort')
ax.plot(size, binary_insertion_sort, label = 'binary_insertion_sort')
ax.plot(size, count_sort, label = 'count_sort')
ax.plot(size, radix_sort, label = 'radix_sort')
ax.plot(size, merge_sort , label = 'merge_sort ')
ax.plot(size, quick_sort, label = 'quick_sort')
ax.plot(size, heap_sort, label = 'heap_sort')
ax.plot(size, shell_sort, label = 'shell_sort')
ax.plot(size, cyurania_sequence, label = 'cyurania_sequence')
ax.set_title("RandomAlmostSorted 100 - 4100", fontsize= 20)
plt.legend(loc='best')
plt.show()
```





Массив почти отсортированных чисел

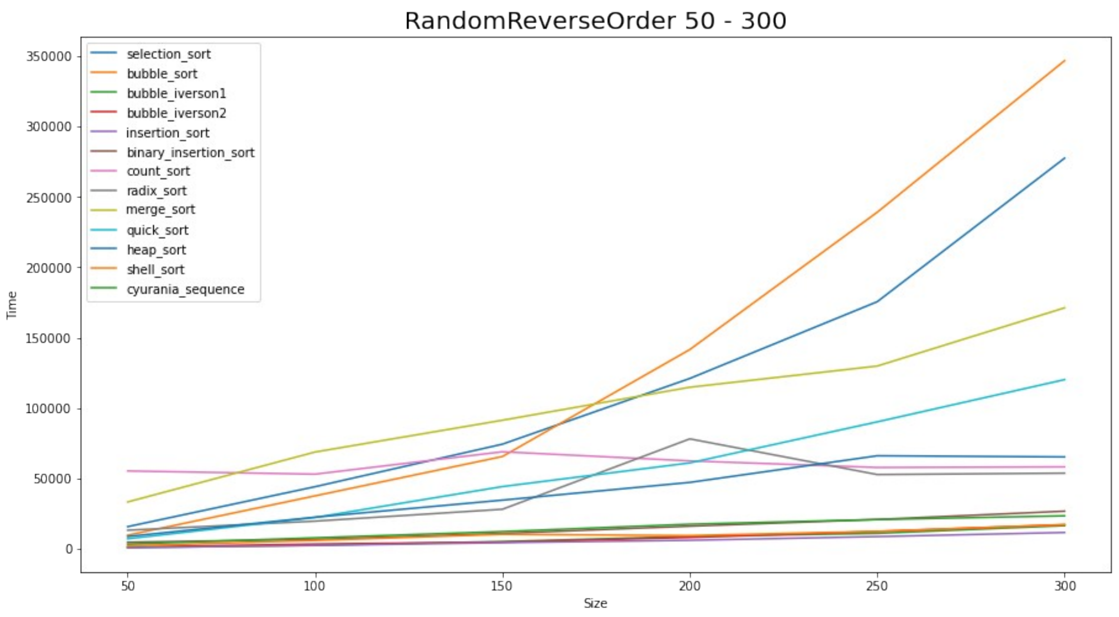
Тааак, ну опять похожая ситуация, что и на прошлых графиках. Хотя погодите, на ранних 2 массивах первые графики выглядели иначе, теперь на 1 все как-то поднялись со дна марианской впадины и стали работать похуже. А вот на втором графике ну как-то сильно ничего не меняется, как жили на дне, так и живут. Только вот merge расстроил, решил прилично подняться. А вот мой любимый heap sort (ну вот да, нравится она мне) не проигрывает merge на данном массиве.

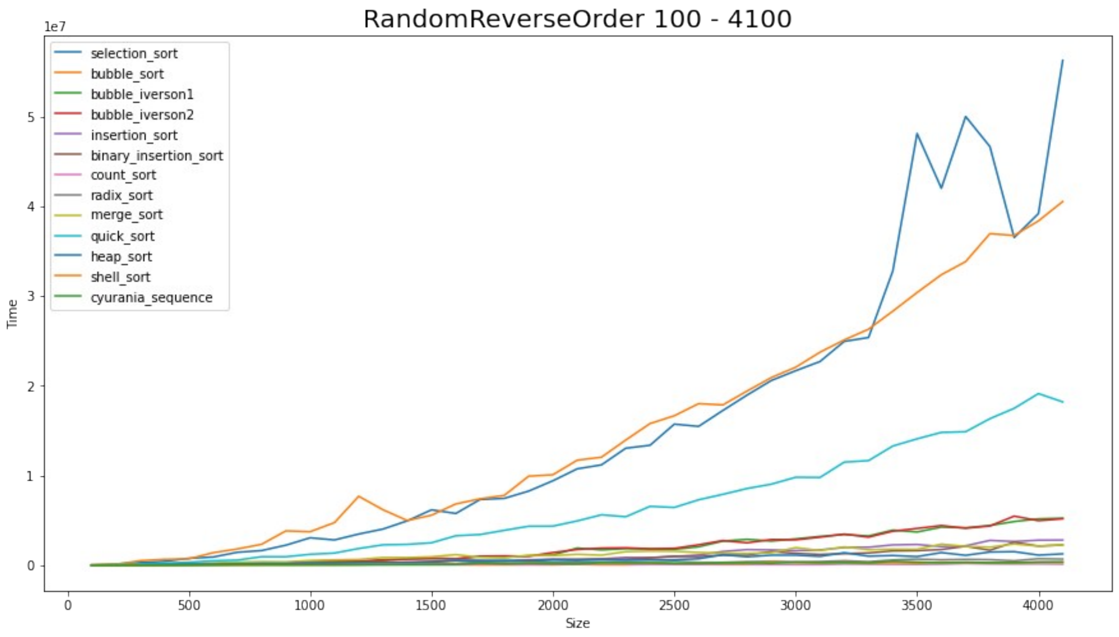
RandomReverseOrder

```
# Размерность 50 - 300
sort_type = ""
size = []
for i in range(50, 301, 50):
    size.append(i)
selection_sort = []
bubble_sort = []
bubble_iverson1 = []
bubble_iverson2 = []
insertion_sort = []
binary_insertion_sort = []
count_sort = []
radix_sort = []
merge_sort = []
quick_sort = []
heap_sort = []
shell_sort = []
cyurania_sequence = []
```

```
with open("random_reverse_order 50 - 300.txt") as f:
    for nums in f:
        if nums [0] == "N":
            sort_type = nums[1:len(nums) - 1]
            continue
        elif sort_type == "selection_sort":
            k = nums.split(" ")
            selection_sort.append(int(k[1]))
        elif sort_type == "bubble_sort":
            k = nums.split(" ")
            bubble_sort.append(int(k[1]))
        elif sort_type == "bubble_iverson1":
            k = nums.split(" ")
            bubble_iverson1.append(int(k[1]))
        elif sort_type == "bubble_iverson2":
            k = nums.split(" ")
            bubble_iverson2.append(int(k[1]))
        elif sort_type == "insertion_sort":
            k = nums.split(" ")
            insertion_sort.append(int(k[1]))
        elif sort_type == "binary_insertion_sort":
            k = nums.split(" ")
            binary_insertion_sort.append(int(k[1]))
        elif sort_type == "count_sort":
            k = nums.split(" ")
            count_sort.append(int(k[1]))
        elif sort_type == "radix_sort":
            k = nums.split(" ")
            radix_sort.append(int(k[1]))
        elif sort_type == "merge_sort":
            k = nums.split(" ")
            merge_sort.append(int(k[1]))
        elif sort_type == "quick_sort":
            k = nums.split(" ")
            quick_sort.append(int(k[1]))
        elif sort_type == "heap_sort":
            k = nums.split(" ")
            heap_sort.append(int(k[1]))
        elif sort_type == "shell_sort":
            k = nums.split(" ")
            shell_sort.append(int(k[1]))
        elif sort_type == "cyurania_sequence":
            k = nums.split(" ")
            cyurania_sequence.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, selection_sort, label = 'selection_sort')
ax.plot(size, bubble_sort, label = 'bubble_sort')
ax.plot(size, bubble_iverson1, label = 'bubble_iverson1')
ax.plot(size, bubble_iverson2, label = 'bubble_iverson2')
ax.plot(size, insertion_sort, label = 'insertion_sort')
ax.plot(size, binary_insertion_sort, label = 'binary_insertion_sort')
ax.plot(size, count_sort, label = 'count_sort')
ax.plot(size, radix_sort, label = 'radix_sort')
ax.plot(size, merge_sort , label = 'merge_sort ')
ax.plot(size, quick_sort, label = 'quick_sort')
ax.plot(size, heap_sort, label = 'heap_sort')
ax.plot(size, shell_sort, label = 'shell_sort')
ax.plot(size, cyurania_sequence, label = 'cyurania_sequence')
ax.set_title("RandomReverseOrder 50 - 300", fontsize= 20)
plt.legend(loc='best')
plt.show()

# Размерность 100 - 4100
size = []
for i in range(100, 4101, 100):
    size.append(i)
selection_sort = []
bubble_sort = []
bubble_iverson1 = []
bubble_iverson2 = []
insertion_sort = []
binary_insertion_sort = []
count_sort = []
radix_sort = []
merge_sort = []
quick_sort = []
heap_sort = []
shell_sort = []
cyurania_sequence = []
with open("random_reverse_order 100 - 4100.txt") as f:
    for nums in f:
        if nums [0] == "N":
            sort_type = nums[1:len(nums) - 1]
            continue
        elif sort_type == "selection_sort":
            k = nums.split(" ")
            selection_sort.append(int(k[1]))
        elif sort_type == "bubble_sort":
            k = nums.split(" ")
            bubble_sort.append(int(k[1]))
        elif sort_type == "bubble_iverson1":
            k = nums.split(" ")
            bubble_iverson1.append(int(k[1]))
        elif sort_type == "bubble_iverson2":
            k = nums.split(" ")
            bubble_iverson2.append(int(k[1]))
        elif sort_type == "insertion_sort":
            k = nums.split(" ")
            insertion_sort.append(int(k[1]))
        elif sort_type == "binary_insertion_sort":
            k = nums.split(" ")
            binary_insertion_sort.append(int(k[1]))
        elif sort_type == "count_sort":
            k = nums.split(" ")
            count_sort.append(int(k[1]))
        elif sort_type == "radix_sort":
            k = nums.split(" ")
            radix_sort.append(int(k[1]))
        elif sort_type == "merge_sort":
            k = nums.split(" ")
            merge_sort.append(int(k[1]))
        elif sort_type == "quick_sort":
            k = nums.split(" ")
            quick_sort.append(int(k[1]))
        elif sort_type == "heap_sort":
            k = nums.split(" ")
            heap_sort.append(int(k[1]))
        elif sort_type == "shell_sort":
            k = nums.split(" ")
            shell_sort.append(int(k[1]))
        elif sort_type == "cyurania_sequence":
            k = nums.split(" ")
            cyurania_sequence.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, selection_sort, label = 'selection_sort')
ax.plot(size, bubble_sort, label = 'bubble_sort')
ax.plot(size, bubble_iverson1, label = 'bubble_iverson1')
ax.plot(size, bubble_iverson2, label = 'bubble_iverson2')
ax.plot(size, insertion_sort, label = 'insertion_sort')
ax.plot(size, binary_insertion_sort, label = 'binary_insertion_sort')
ax.plot(size, count_sort, label = 'count_sort')
ax.plot(size, radix_sort, label = 'radix_sort')
ax.plot(size, merge_sort , label = 'merge_sort ')
ax.plot(size, quick_sort, label = 'quick_sort')
ax.plot(size, heap_sort, label = 'heap_sort')
ax.plot(size, shell_sort, label = 'shell_sort')
ax.plot(size, cyurania_sequence, label = 'cyurania_sequence')
ax.set_title("RandomReverseOrder 100 - 4100", fontsize= 20)
plt.legend(loc='best')
plt.show()
```





Массив чисел, идущих в обратном порядке

Так, ну на пузырёк и сорт. выбором смотреть не буду, квадрат есть квадрат. Тут мы видим, что пузырёк с 2 Айверсонами поднялись наверх, но quick всё равно медленнее… Моя любимая heap sort всё так же показывает хорошую асимптотику, что радует.