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

RandomZeroToFive

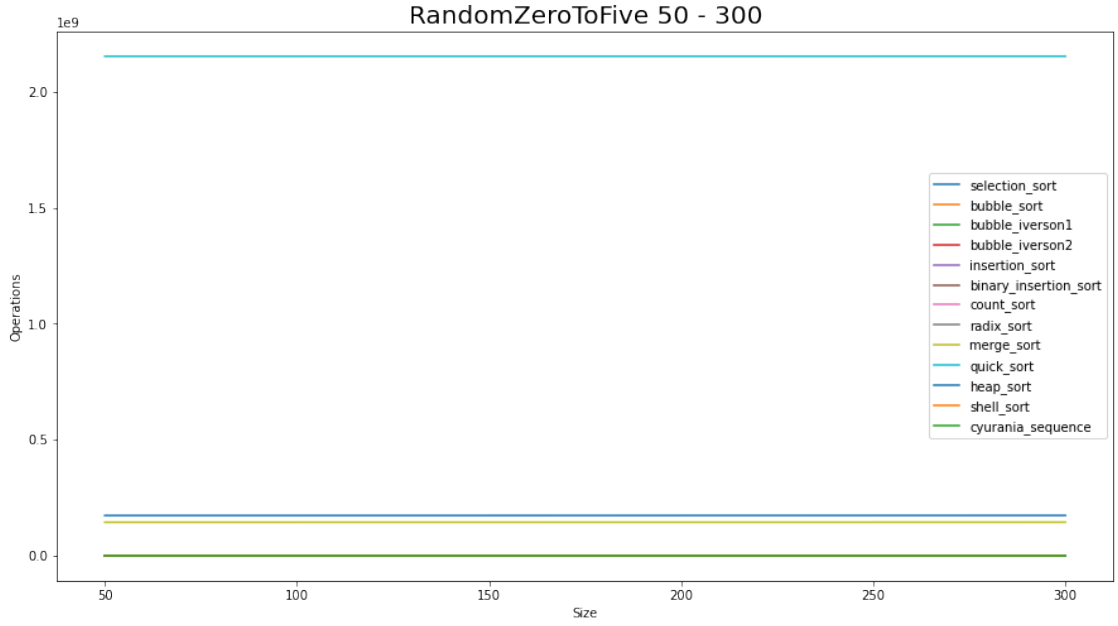
```
# Размерность 50 - 300
sort_type = ""
size = []
for i in range(50, 301, 50):
    size.append(i)
selection_sort = []
bubble_sort = []
bubble_iverson1 = []
bubble_iverson2 = []
insertion_sort = []
binary_insertion_sort = []
count_sort = []
radix_sort = []
merge_sort = []
quick_sort = []
heap_sort =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

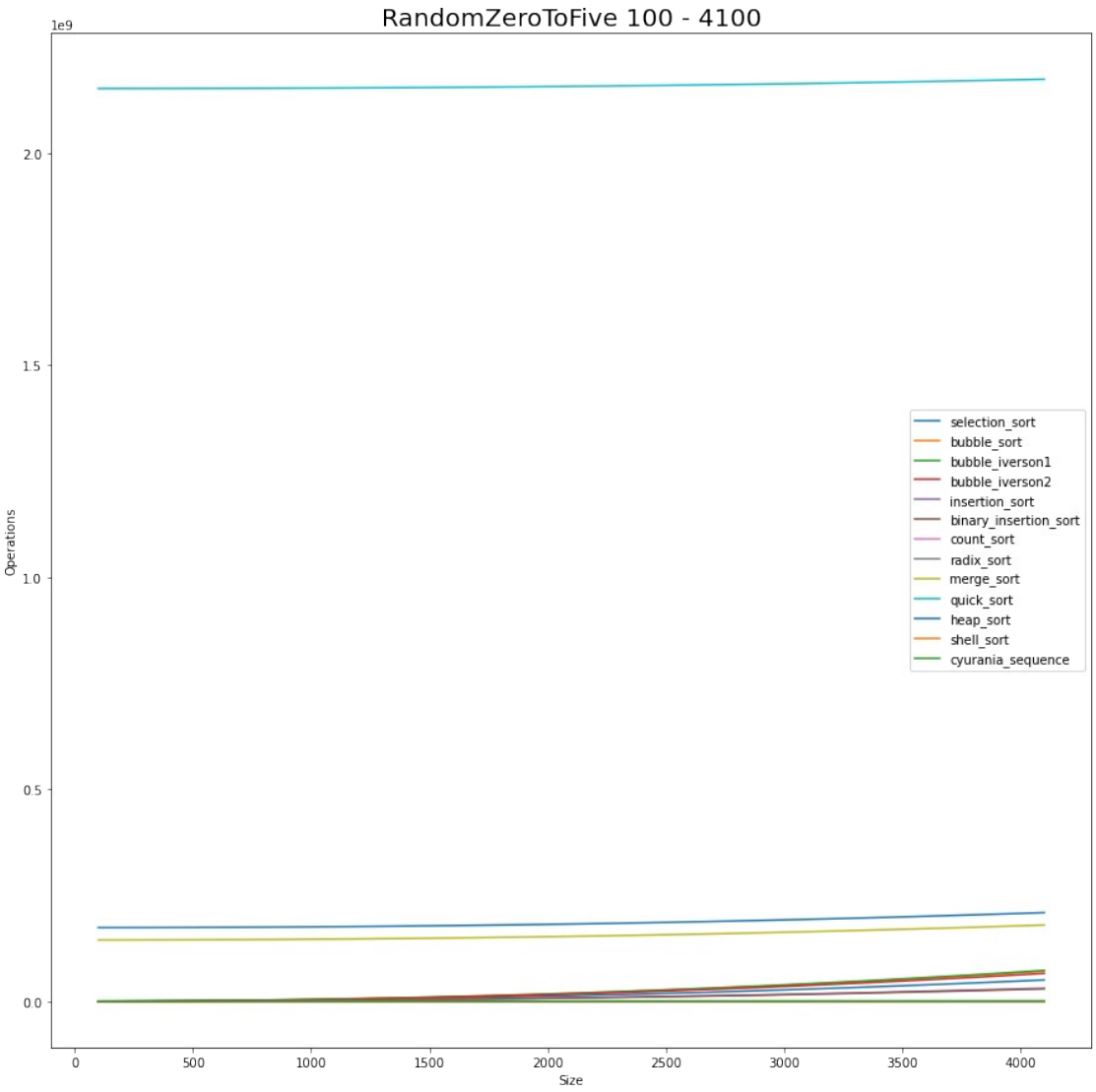
```
            k = nums.split(" ")
            count_sort.append(int(k[1]))
        elif sort_type == "radix_sort":
            k = nums.split(" ")
            radix_sort.append(int(k[1]))
        elif sort_type == "merge_sort":
            k = nums.split(" ")
            merge_sort.append(int(k[1]))
        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("Operations")
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 = []
```

```
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(15,15)
ax.set_xlabel("Size")
ax.set_ylabel("Operations")
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()
```





RandomZeroToFourThousand

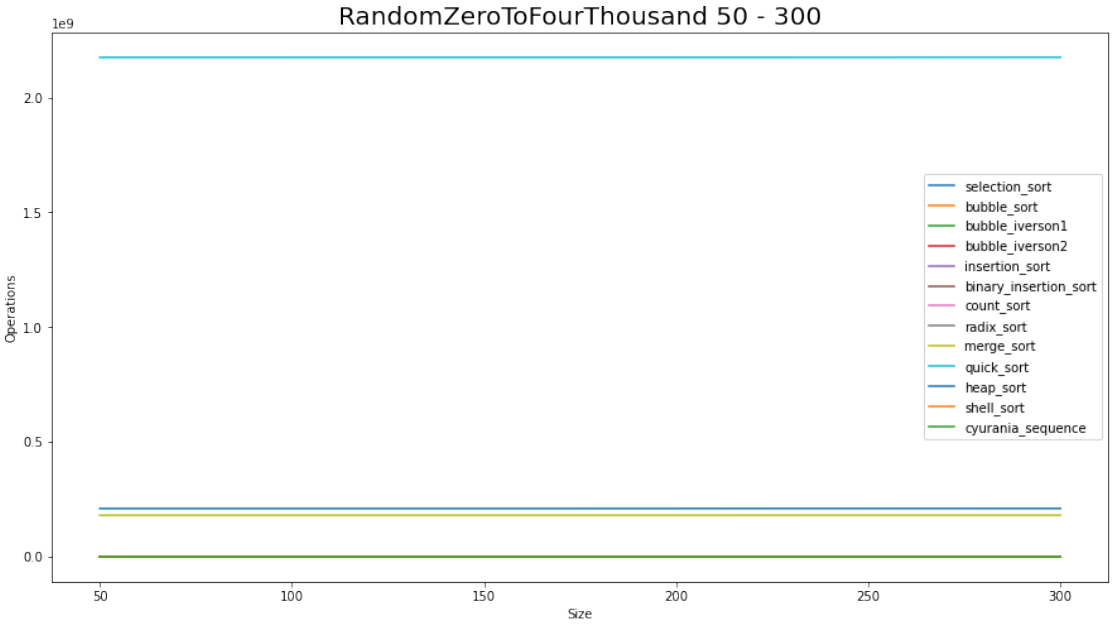
```
# Размерность 50 - 300
sort_type = ""
size = []
for i in range(50, 301, 50):
    size.append(i)
selection_sort = []
bubble_sort = []
bubble_iverson1 = []
bubble_iverson2 = []
insertion_sort = []
binary_insertion_sort = []
count_sort = []
radix_sort = []
merge_sort = []
```

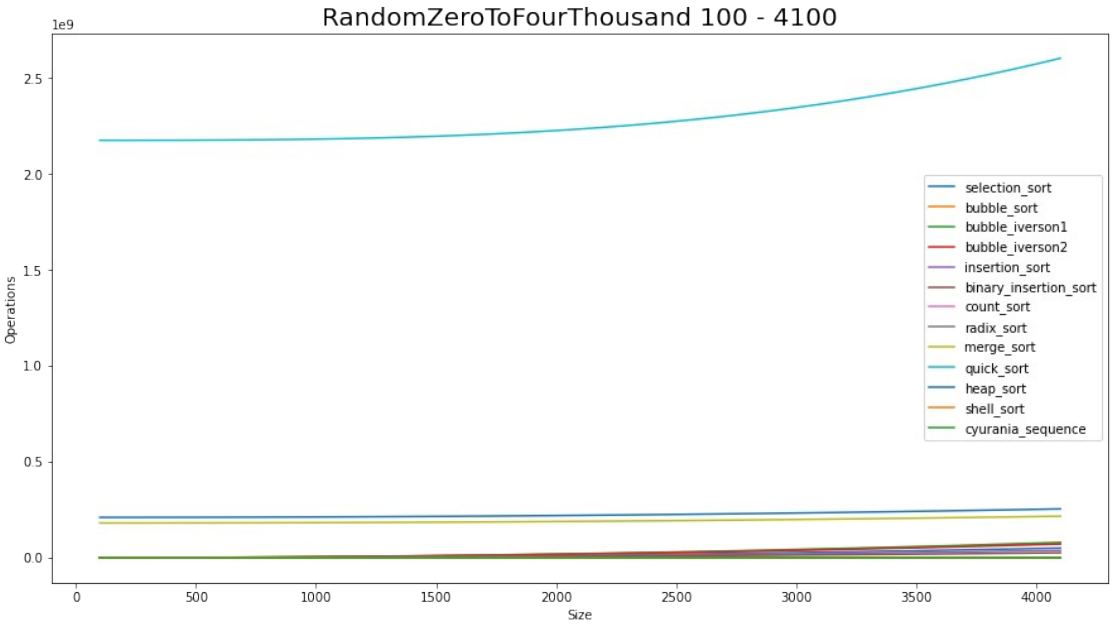
```
quick_sort = []
heap_sort = []
shell_sort = []
cyurania_sequence = []
with open("random_zero_to_four_thousand 50 - 300.txt") as f:
    for nums in f:
        if nums [0] == "N":
            sort_type = nums[1:len(nums) - 1]
            continue
        elif sort_type == "selection_sort":
            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("Operations")
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("Operations")
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()
```





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 = []
```

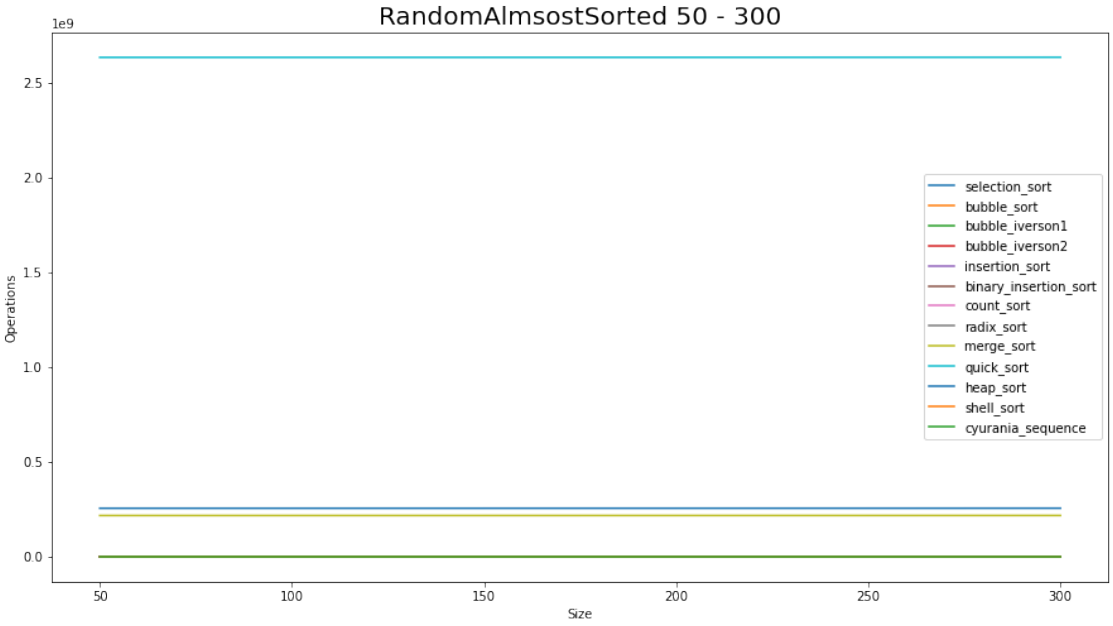
```
insertion_sort = []
binary_insertion_sort = []
count_sort = []
radix_sort = []
merge_sort = []
quick_sort = []
heap_sort = []
shell_sort = []
cyurania_sequence = []
with open("random_almost_sorted 50 - 300.txt") as f:
    for nums in f:
        if nums [0] == "N":
            sort_type = nums[1:len(nums) - 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]))
```

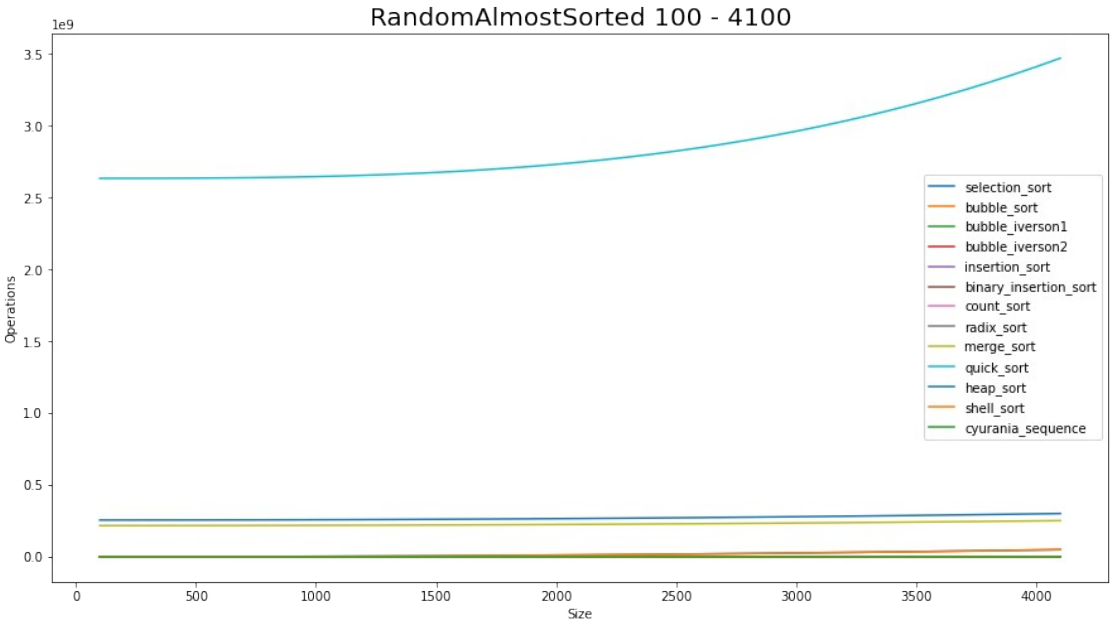
```
        elif sort_type == "cyurania_sequence":
            k = nums.split(" ")
            cyurania_sequence.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Size")
ax.set_ylabel("Operations")
ax.plot(size, selection_sort, label = 'selection_sort')
ax.plot(size, 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 = []
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("Operations")
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()
```





RandomReverseOrder

```
# Размерность 50 - 300
sort_type = ""
size = []
```

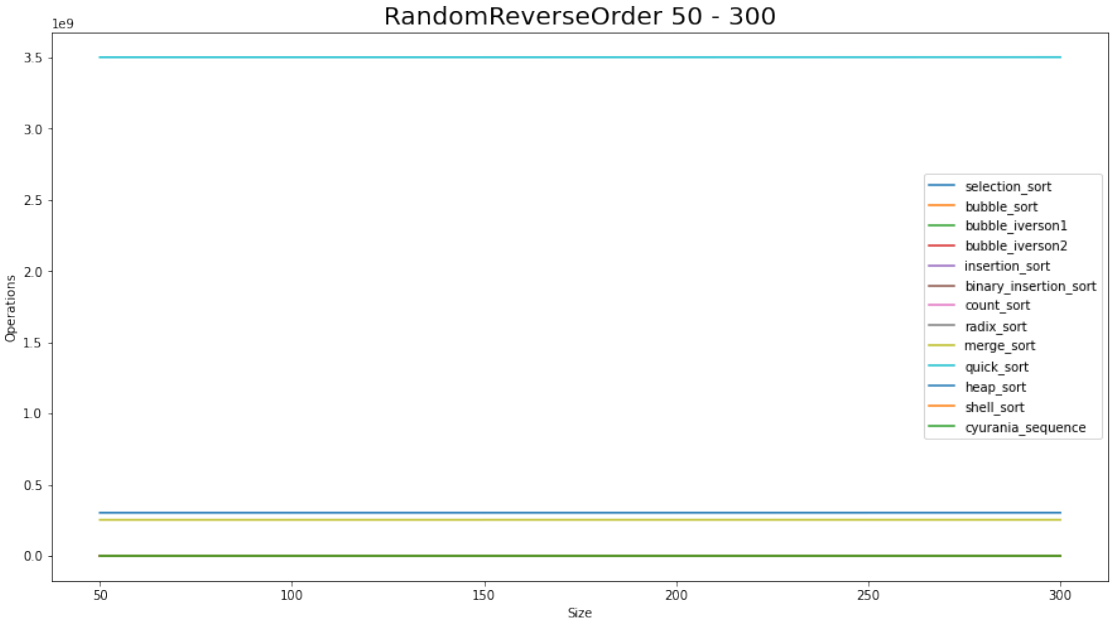
```
for i in range(50, 301, 50):
    size.append(i)
selection_sort = []
bubble_sort = []
bubble_iverson1 = []
bubble_iverson2 = []
insertion_sort = []
binary_insertion_sort = []
count_sort = []
radix_sort = []
merge_sort = []
quick_sort = []
heap_sort = []
shell_sort = []
cyurania_sequence = []
with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




```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
<ipython-input-24-313c7d1d0900> in <module>
    147 ax.set_xlabel("Size")
    148 ax.set_ylabel("Operations")
--> 149 ax.plot(size, selection_sort, label = 'selection_sort')
    150 ax.plot(size, bubble_sort, label = 'bubble_sort')
    151 ax.plot(size, bubble_iverson1, label = 'bubble_iverson1')

~\anaconda3\lib\site-packages\matplotlib\axes\_axes.py in plot(self,
scalex, scaley, data, *args, **kwargs)
   1741         """
   1742         kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D)
-> 1743         lines = [*self._get_lines(*args, data=data, **kwargs)]
   1744         for line in lines:
   1745             self.add_line(line)
```

```
~\anaconda3\lib\site-packages\matplotlib\axes\_base.py in
__call__(self, data, *args, **kwargs)
    271                 this += args[0],
    272                 args = args[1:]
--> 273             yield from self._plot_args(this, kwargs)
    274
    275     def get_next_color(self):

~\anaconda3\lib\site-packages\matplotlib\axes\_base.py in
_plot_args(self, tup, kwargs)
    397
    398         if x.shape[0] != y.shape[0]:
--> 399             raise ValueError(f"x and y must have same first
dimension, but "
    400                              f"have shapes {x.shape} and
{y.shape}")
    401         if x.ndim > 2 or y.ndim > 2:

ValueError: x and y must have same first dimension, but have shapes
(41,) and (81,)
```