Были получены результаты по работе программы, созданы 4 таблицы: по две на каждый разброс размеров массива (50-300 и 100-4100) - измерение времени и измерение количество элементарных операций

При измерении времени работы сортировки были возможно выбросы значений, так как время работы зависит от работы самого процессора, который занимается множеством заданий на компьютере

Поэтому на графиках измерения времени заметно, что время выполнения нелинейно, и имеет такие выбросы - пики на графике. Особенно они заметны на больших объемах данных, где время выполнения программы достаточно велико

На больших объемах данных получается наиболее объективно оценивать работу сортировок. Стоит отметить работу некоторых сортировок на почти отсортированных массивах: сортировки пузырьком с условиями Айверсона, сортировки вставками. Малое время работы данных сортировок на таких массивах объясняется их алгоритмом работы. Такая же ситуация наблюдается и при измерениях количества элементарных операций данных сортировок

Графики измерения количества элементарных операций намного более плавные, по сравнению с графиками времени, так как количество не зависит от работы процесса, часто сильно зависят от размера массива, и на графиках прослеживается данная зависимость

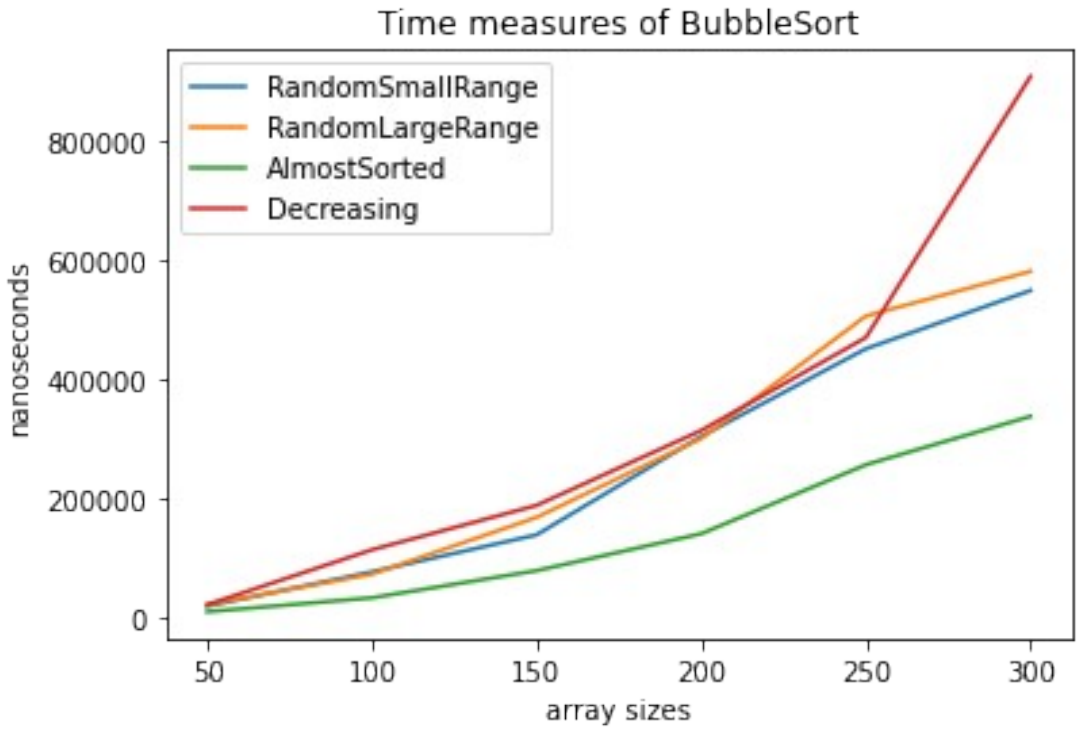
Стоит отметить поведение количества операций сортировок: RadixSort и CountingSort. Количество операций не зависит от самих значений элементов массива, а только от его размера, поэтому графики количества операций для всех генераций массива совпали в одну кривую

Если же рассматривать поведение сортировок для конкретной генерации массива, то можно заметить, что для почти отсортированных массивов хуже всего работает обычная сортировка пузырьком и сортировка выбором, а также быстрая сортировка. В остальных типах массивов можно заметить расслоение сортировок по времени их выполнения, в среднем хуже всего работают сортировки пузырьком, далее идут сортировки выбором, вставками и быстрая сортировка, и наиболее быстроработающими (по времени и по количеству операций) на графиках оказываются сортировки: пирамидная, подсчетом, цифровая, Шелла, слиянием и Шелла с последовательностью Циура

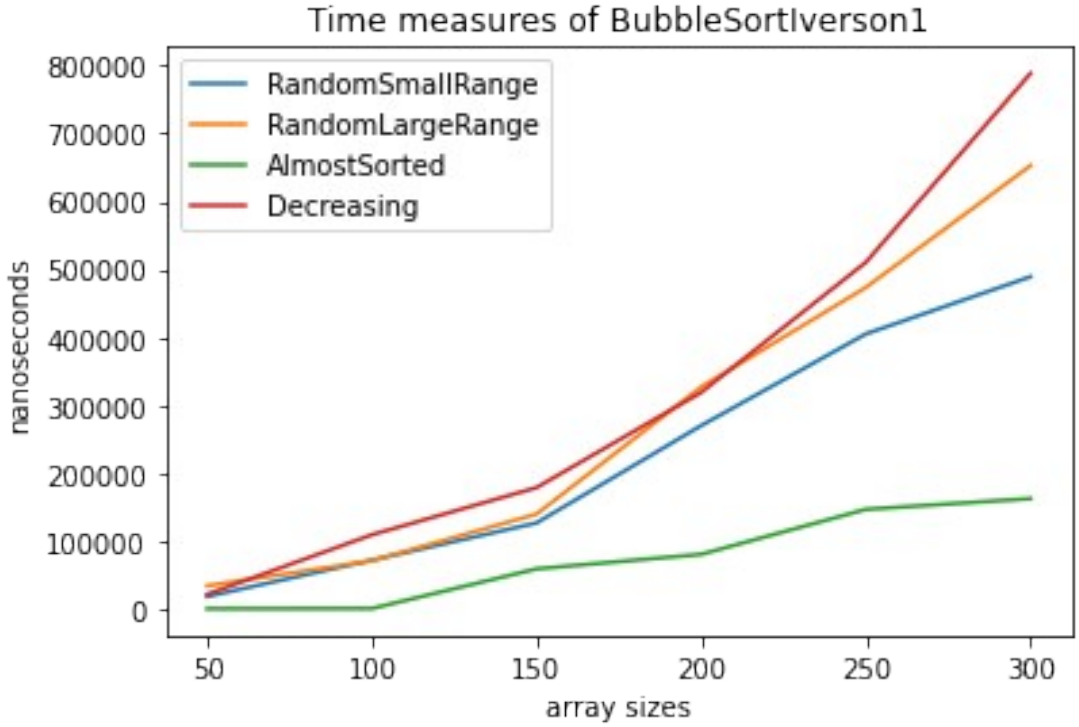
```
import pandas as pd
import matplotlib.pyplot as plt

times_small = pd.read_csv("smallSizesTimes.csv", sep=';')
oper_small = pd.read_csv("smallSizesOper.csv", sep=';')
times_big = pd.read_csv("bigSizesTimes.csv", sep=';')
```

```
oper_big = pd.read_csv("bigSizesOper.csv", sep=';')
gens = ["RandomSmallRange", "RandomLargeRange", "AlmostSorted",
"Decreasing"]
sorts = ["BubbleSort", "BubbleSortIverson1", "BubbleSortIverson12",
         "CountingSort", "HeapSort", "SimpleInsertionSort",
"BinaryInsertionSort",
         "MergeSort", "QuickSort", "RadixSort", "SelectionSort",
         "ShellSort", "CiuraShellSort"]

#@title Графики измеренного времени по всем генерациям для конкретной
сортировки на маленьких массивах
#small size range, for all sorts, time
for curr in sorts:
  plt.figure(figsize=(10, 10),dpi=500)
  fig, ax = plt.subplots()

  for gen in gens:
      plt.plot(times_small["Array sizes"], times_small[gen + " & " +
curr], label=gen)
  ax.set(xlabel='array sizes', ylabel='nanoseconds',
        title='Time measures of ' + curr)

  fig.savefig(curr + "_time.png")
  plt.legend()
  plt.show()
```

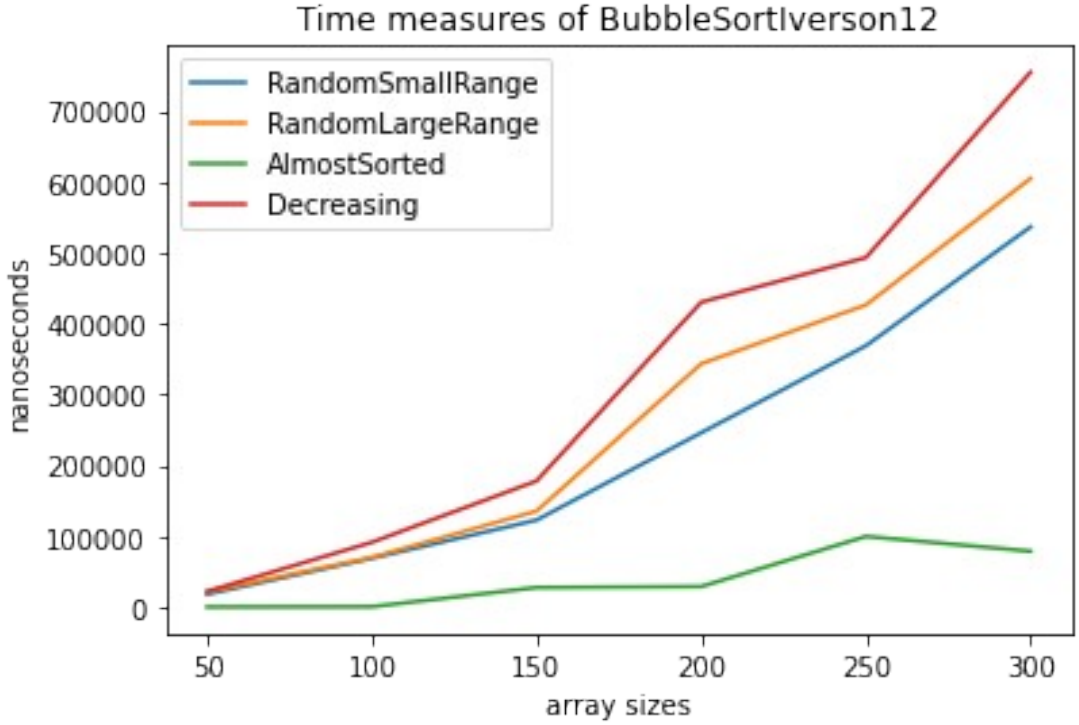
```
<Figure size 5000x5000 with 0 Axes>
```



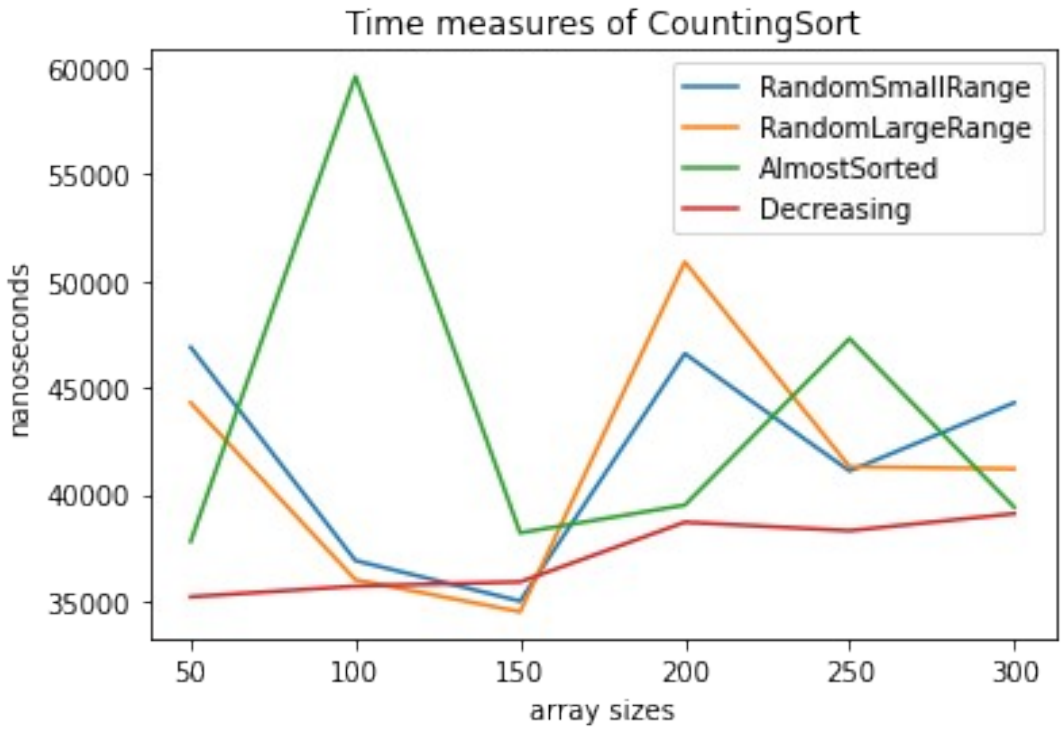
<Figure size 5000x5000 with 0 Axes>



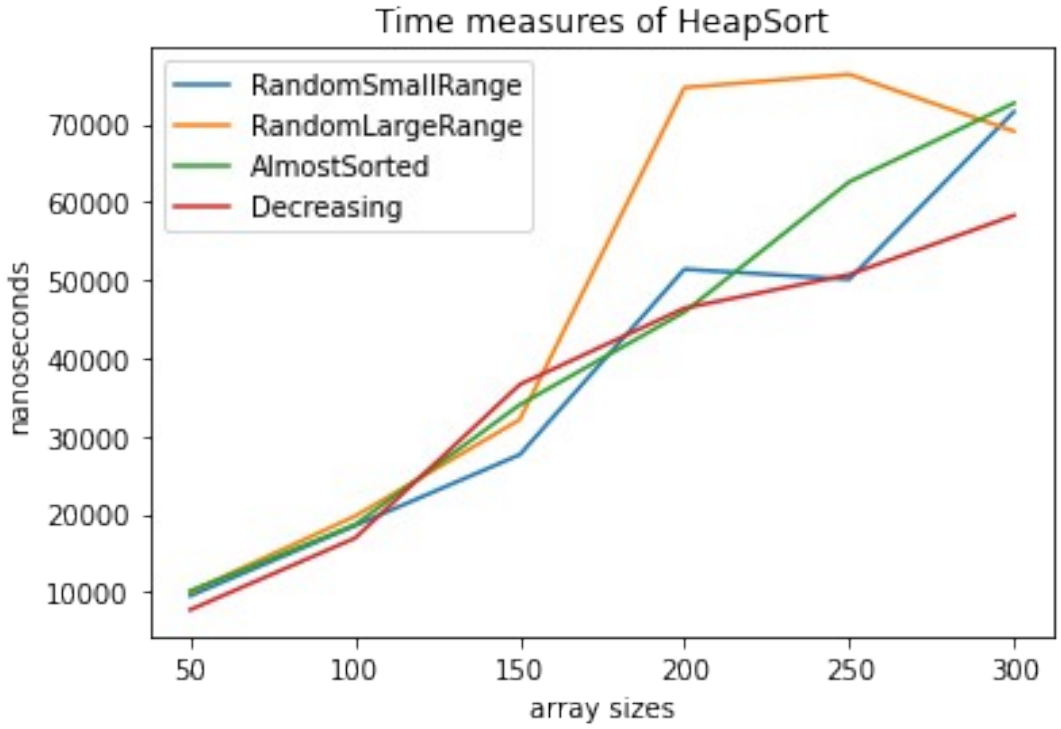
<Figure size 5000x5000 with 0 Axes>



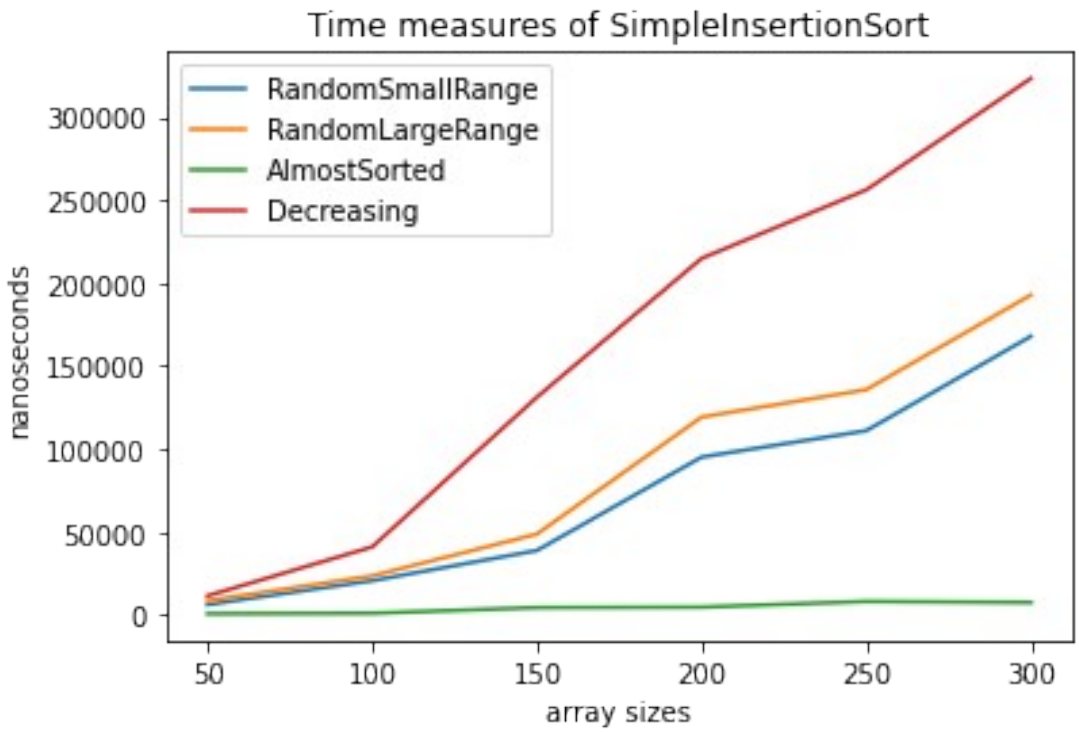
<Figure size 5000x5000 with 0 Axes>



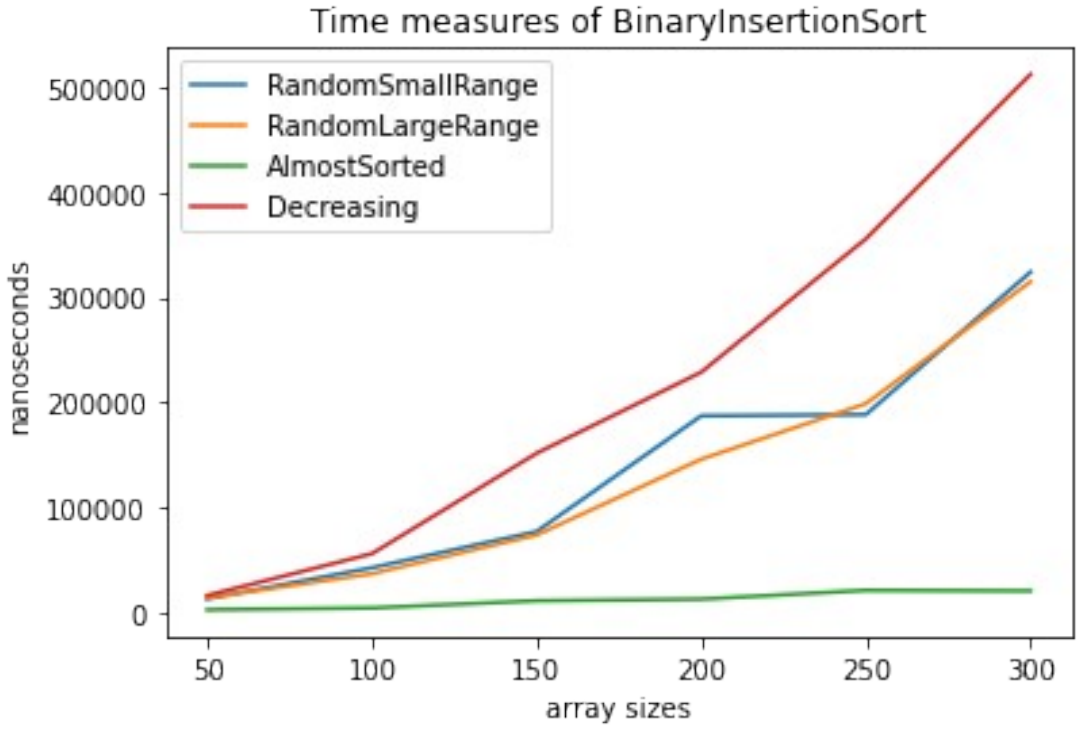
<Figure size 5000x5000 with 0 Axes>



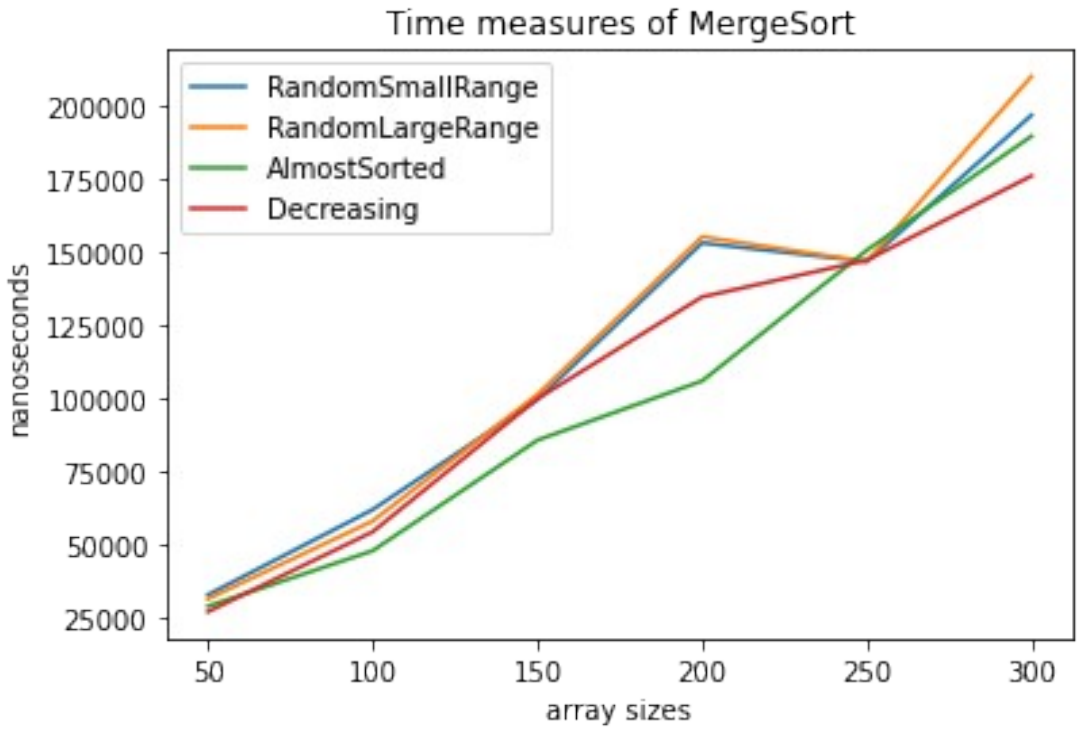
<Figure size 5000x5000 with 0 Axes>



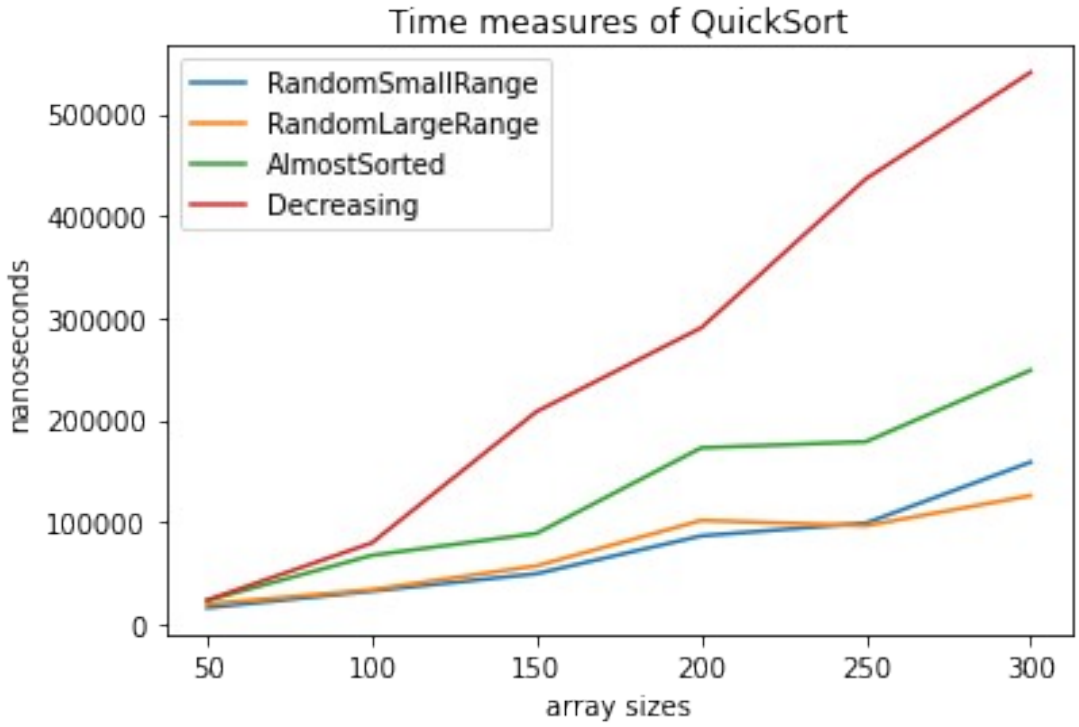
<Figure size 5000x5000 with 0 Axes>



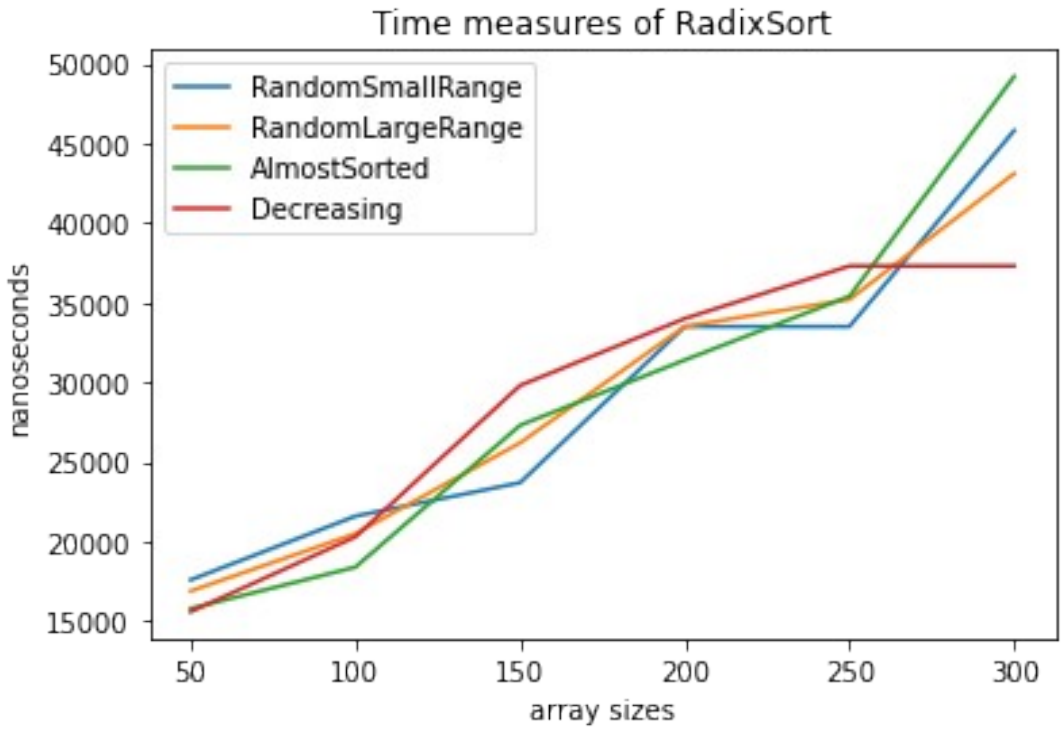
<Figure size 5000x5000 with 0 Axes>



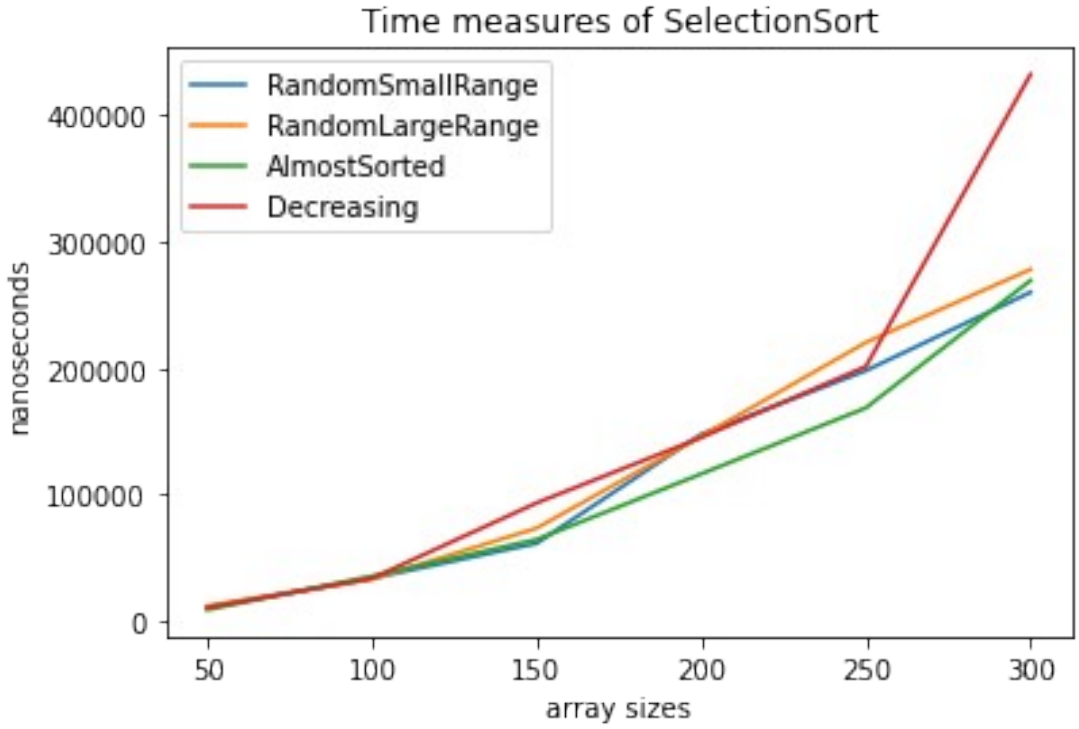
<Figure size 5000x5000 with 0 Axes>



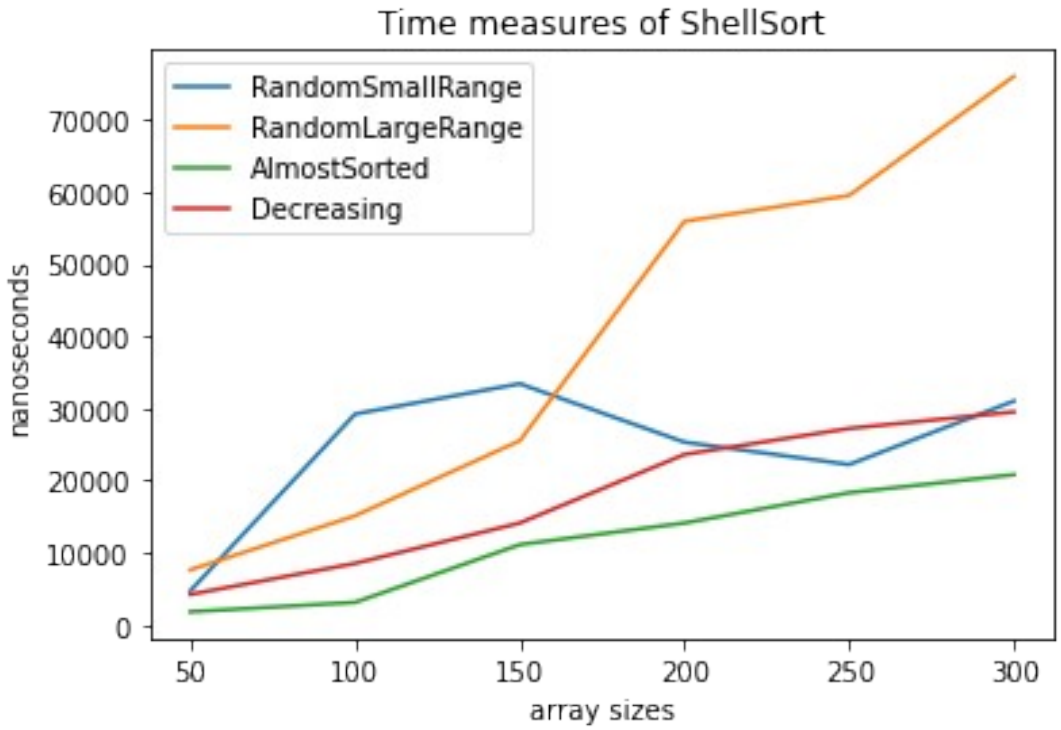
<Figure size 5000x5000 with 0 Axes>



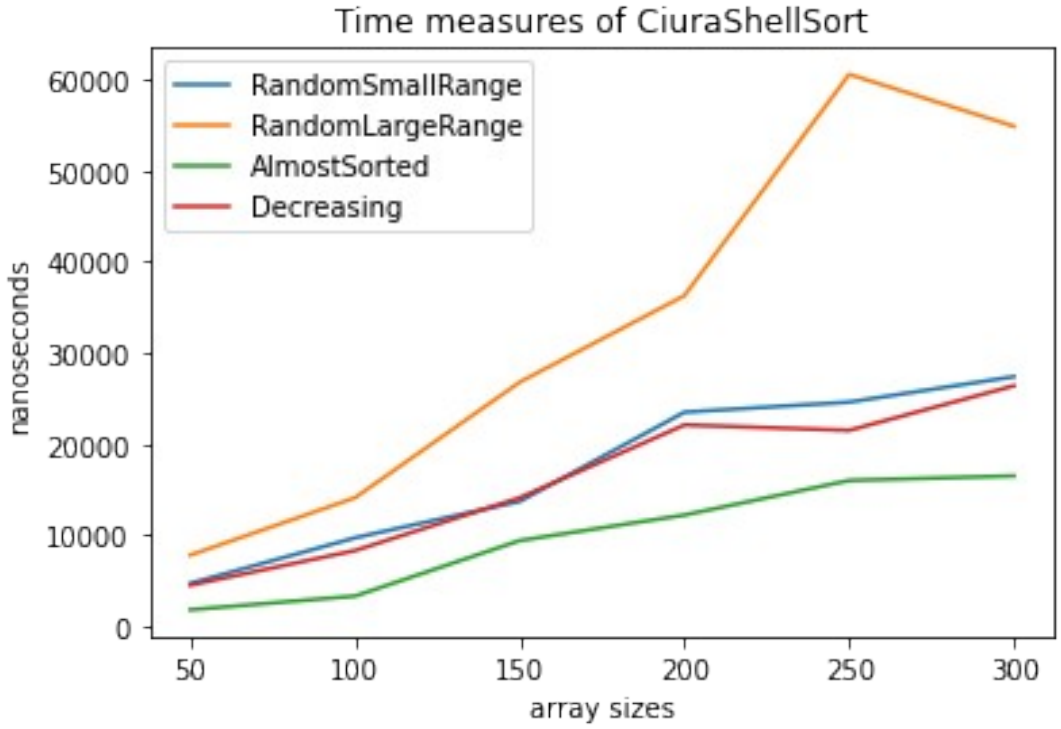
<Figure size 5000x5000 with 0 Axes>



<Figure size 5000x5000 with 0 Axes>



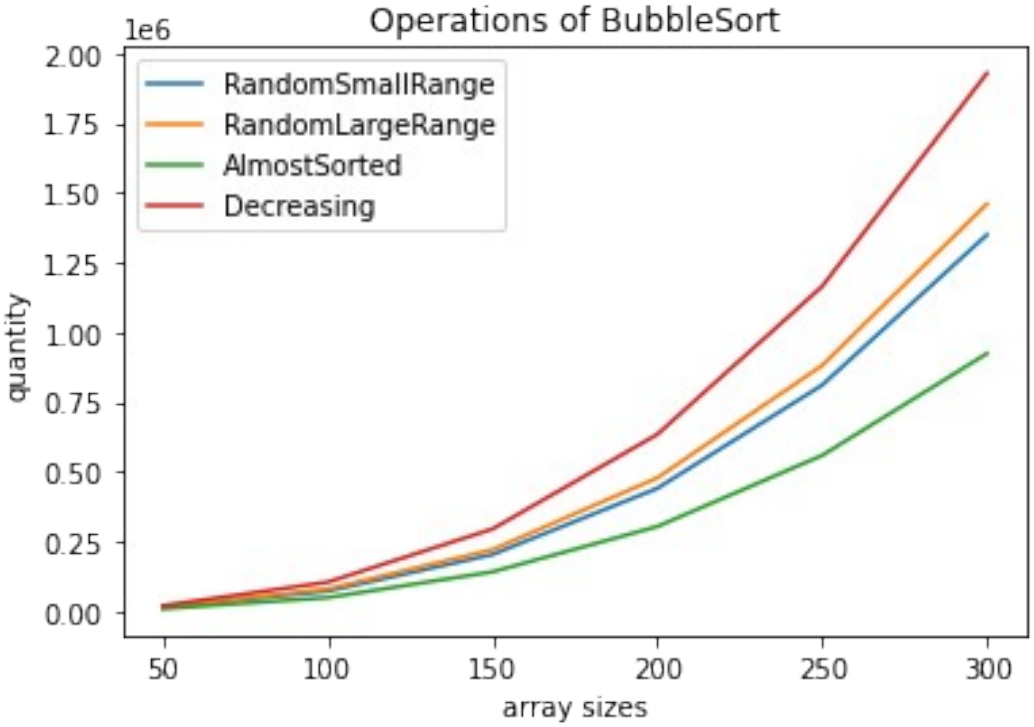
<Figure size 5000x5000 with 0 Axes>



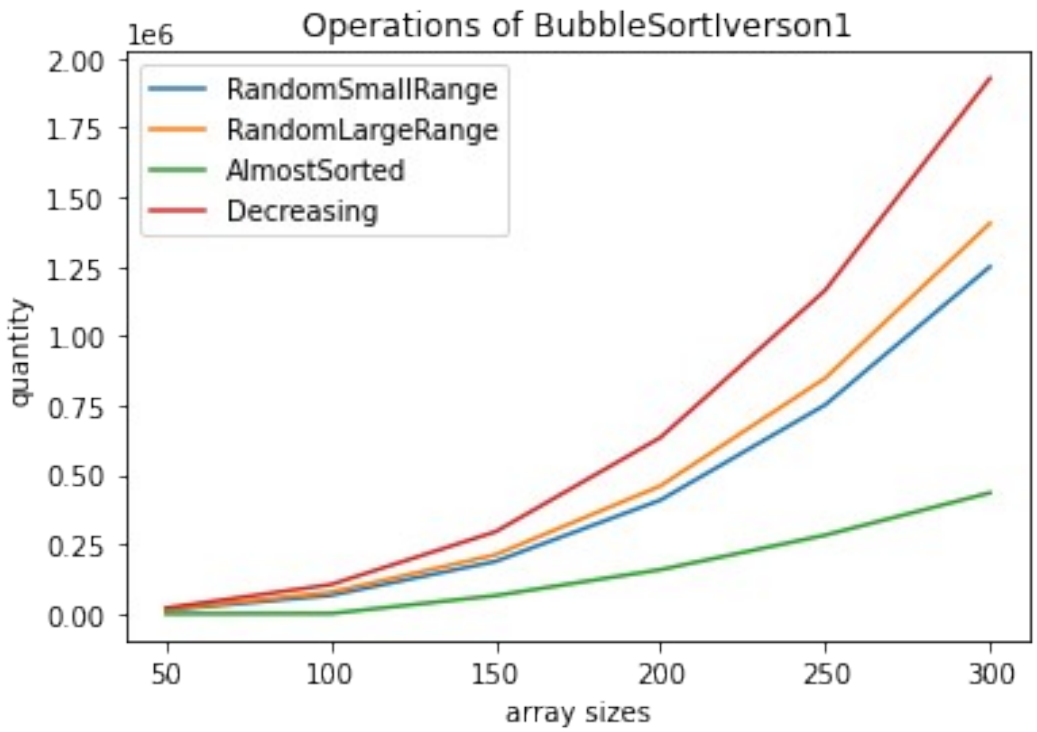

```
#@title Графики количества операций по всем генерациям для конкретной сортировки на маленьких массивах
#small size range, for all sorts, operations
```

```
for curr in sorts:
  plt.figure(figsize=(10, 10),dpi=500)
  fig, ax = plt.subplots()

  for gen in gens:
      plt.plot(oper_small["Array sizes"], oper_small[gen + " & " +
curr], label=gen)
  ax.set(xlabel='array sizes', ylabel='quantity',
         title='Operations of ' + curr)

  fig.savefig(curr + "_time.png")
  plt.legend()
  plt.show()
```

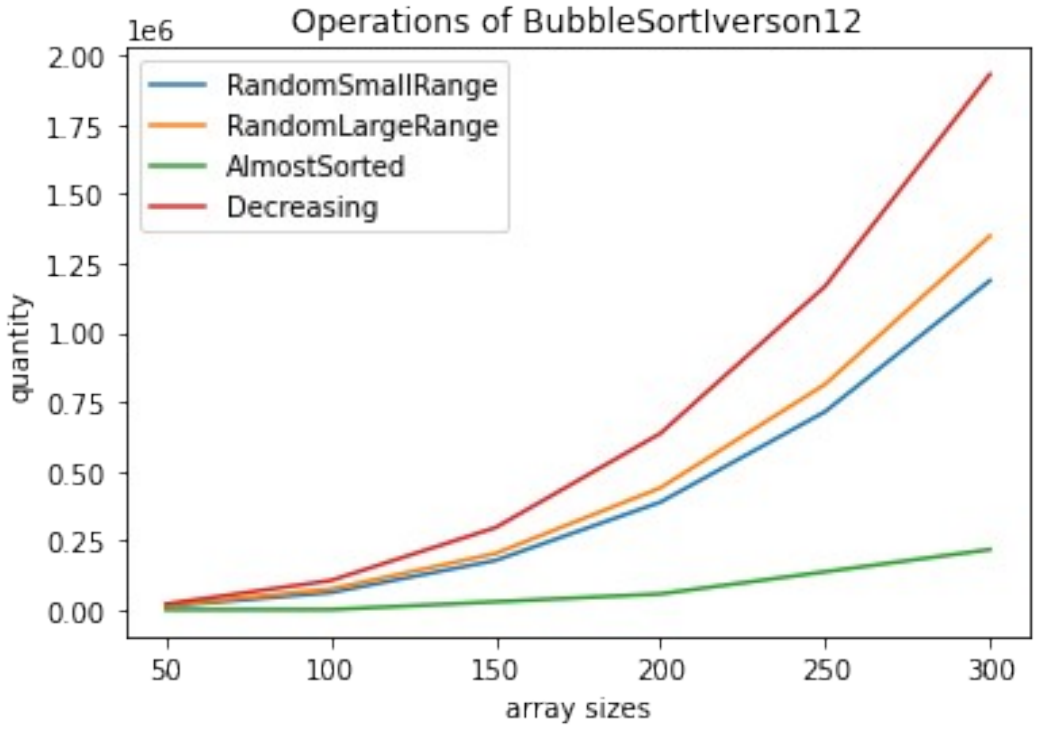
```
<Figure size 5000x5000 with 0 Axes>
```



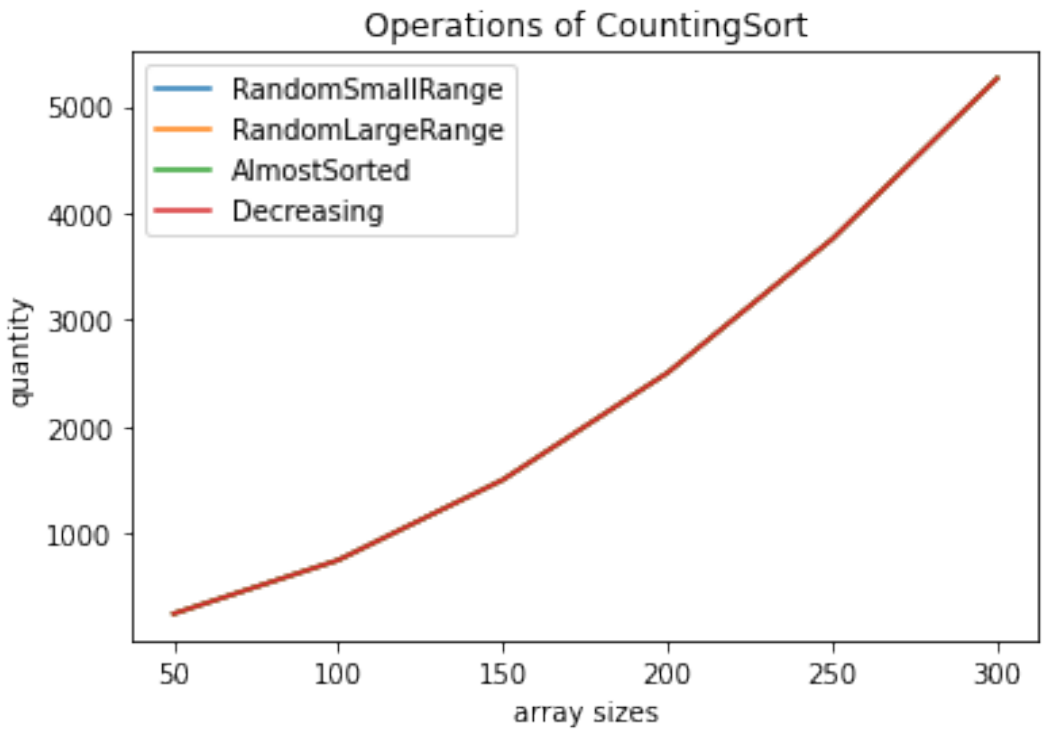

```
<Figure size 5000x5000 with 0 Axes>
```



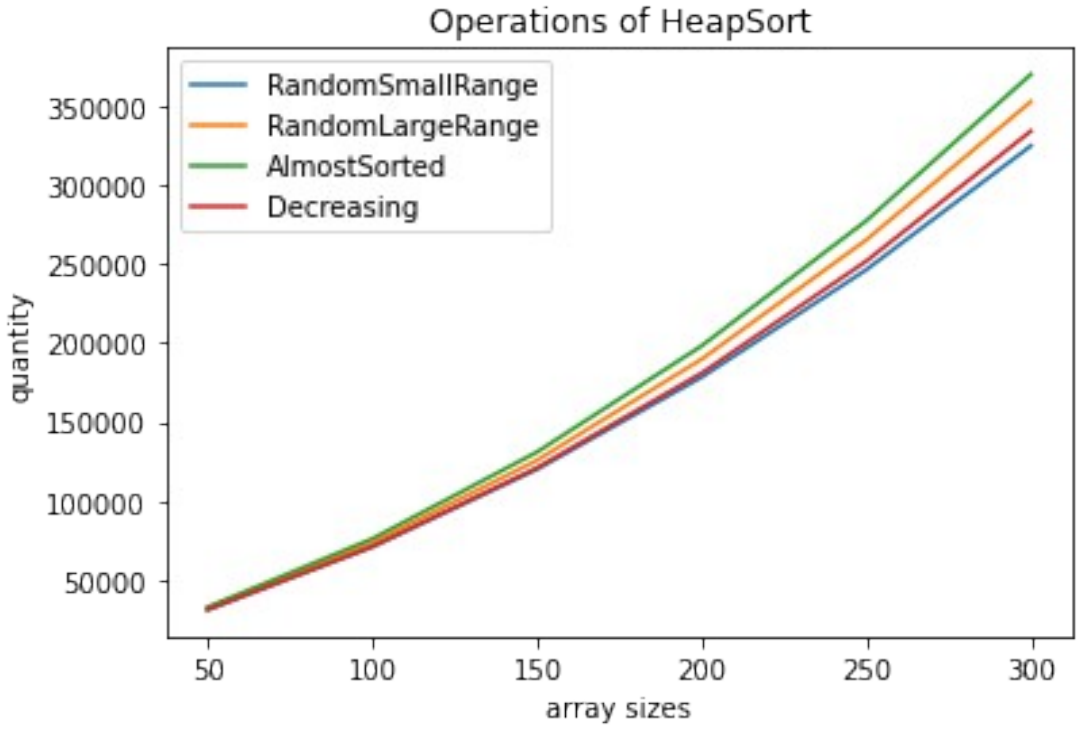
<Figure size 5000x5000 with 0 Axes>



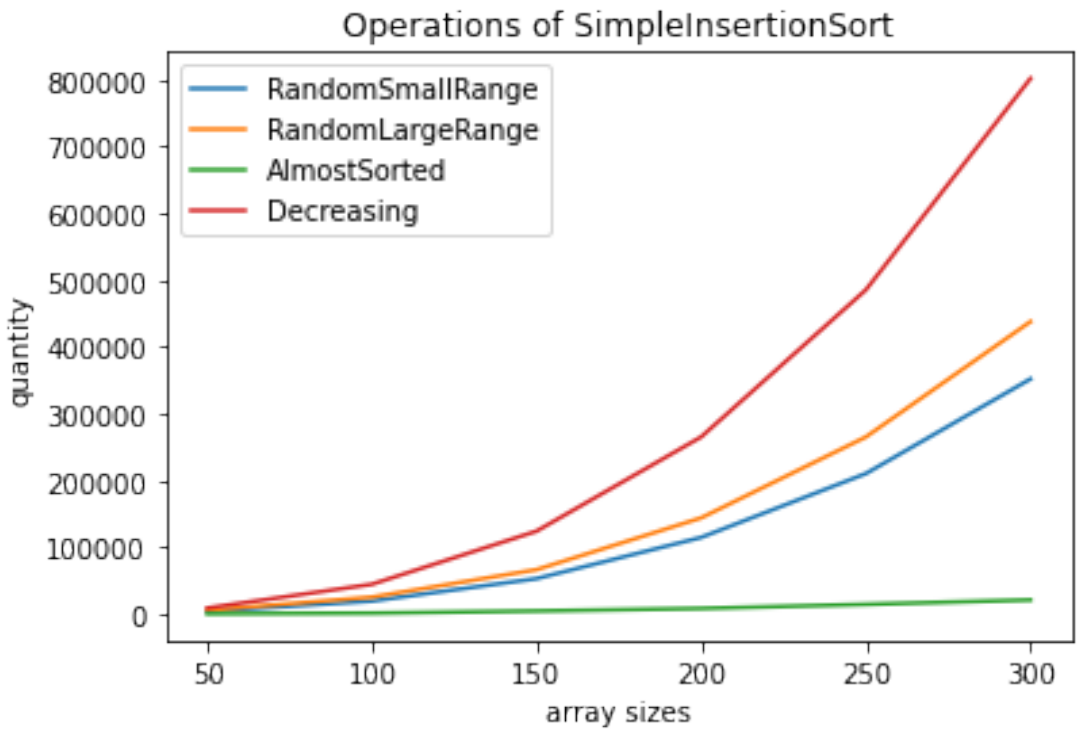
<Figure size 5000x5000 with 0 Axes>



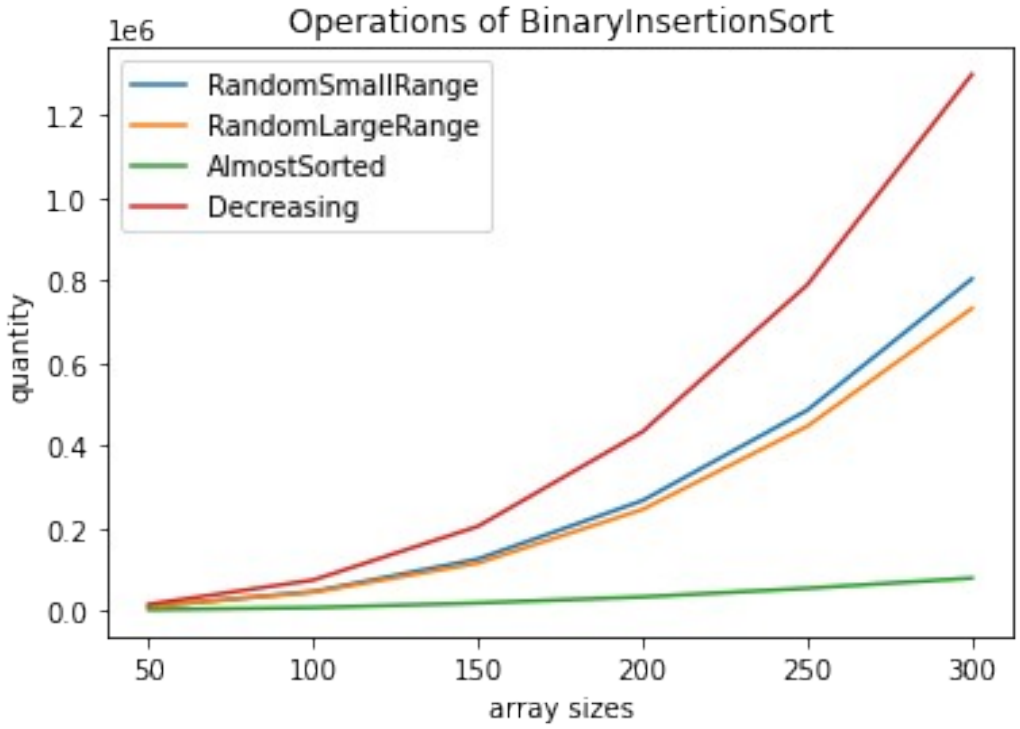
<Figure size 5000x5000 with 0 Axes>



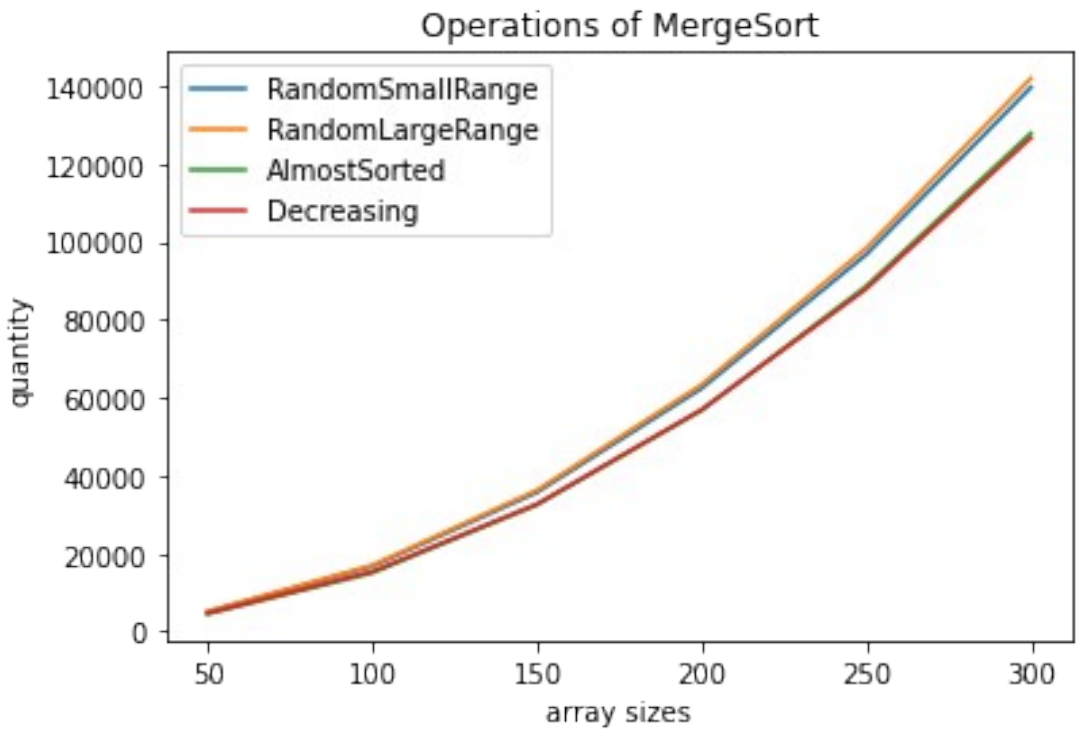
<Figure size 5000x5000 with 0 Axes>



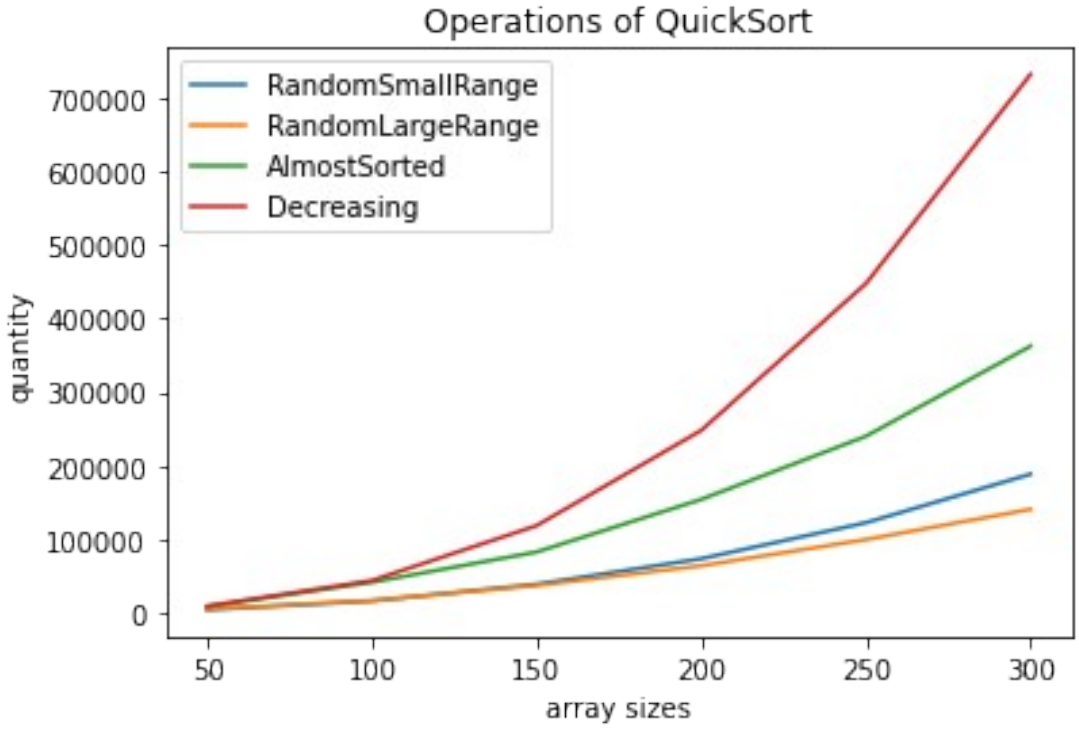
<Figure size 5000x5000 with 0 Axes>



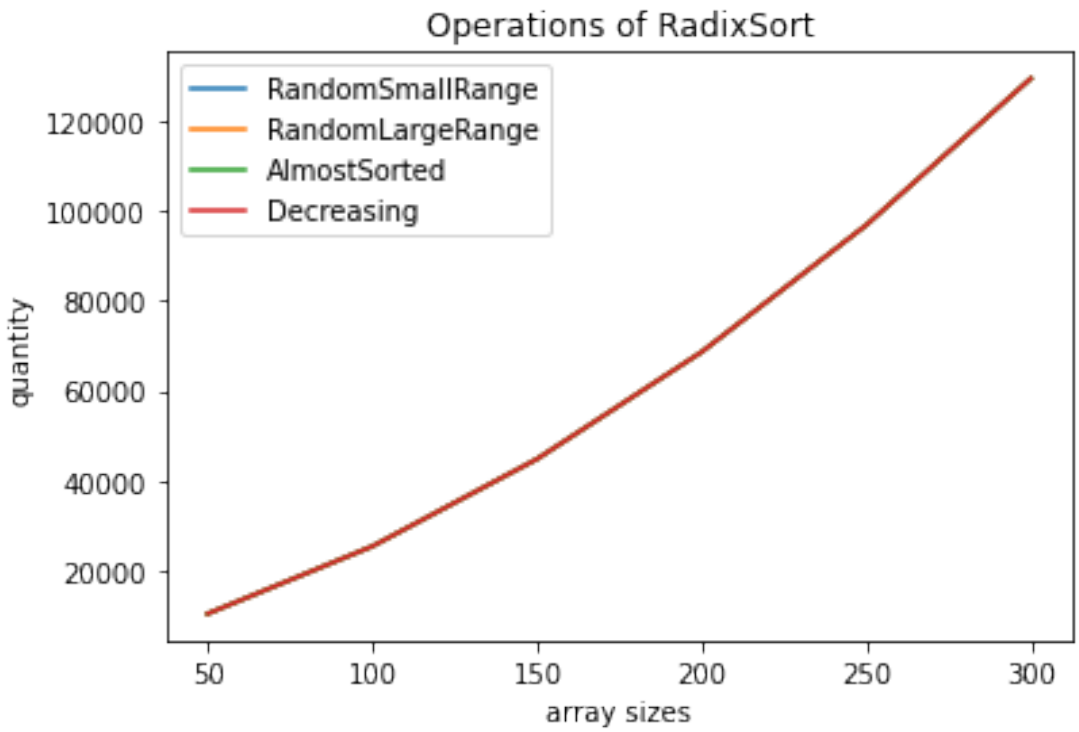
<Figure size 5000x5000 with 0 Axes>



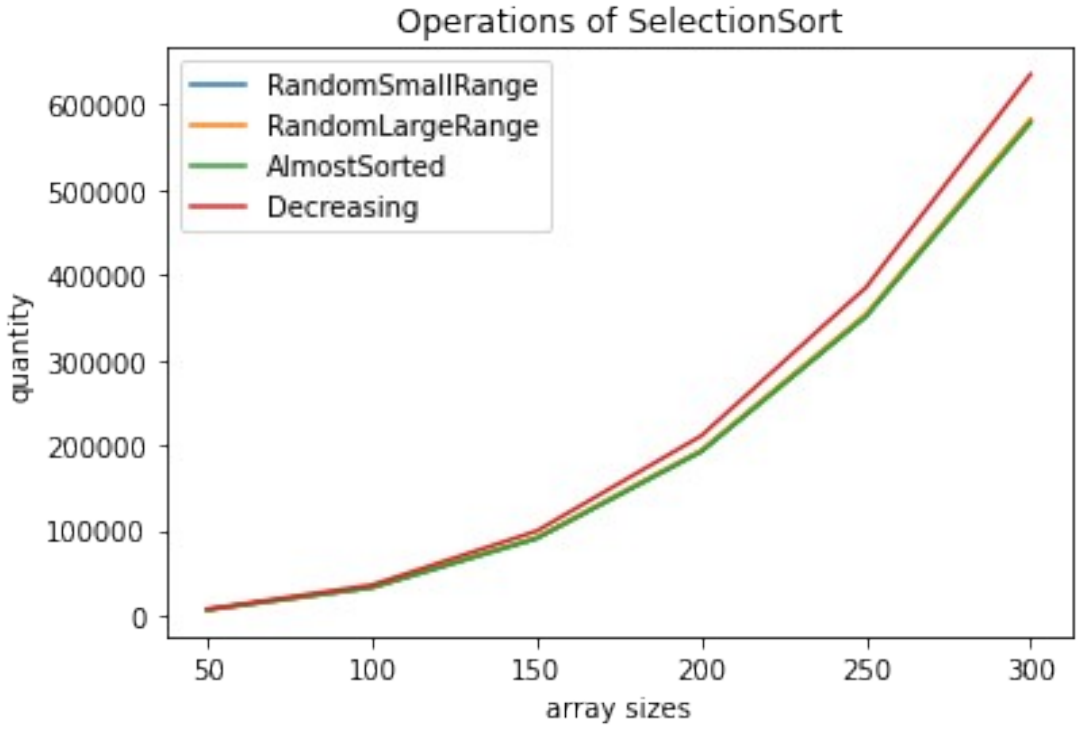
<Figure size 5000x5000 with 0 Axes>



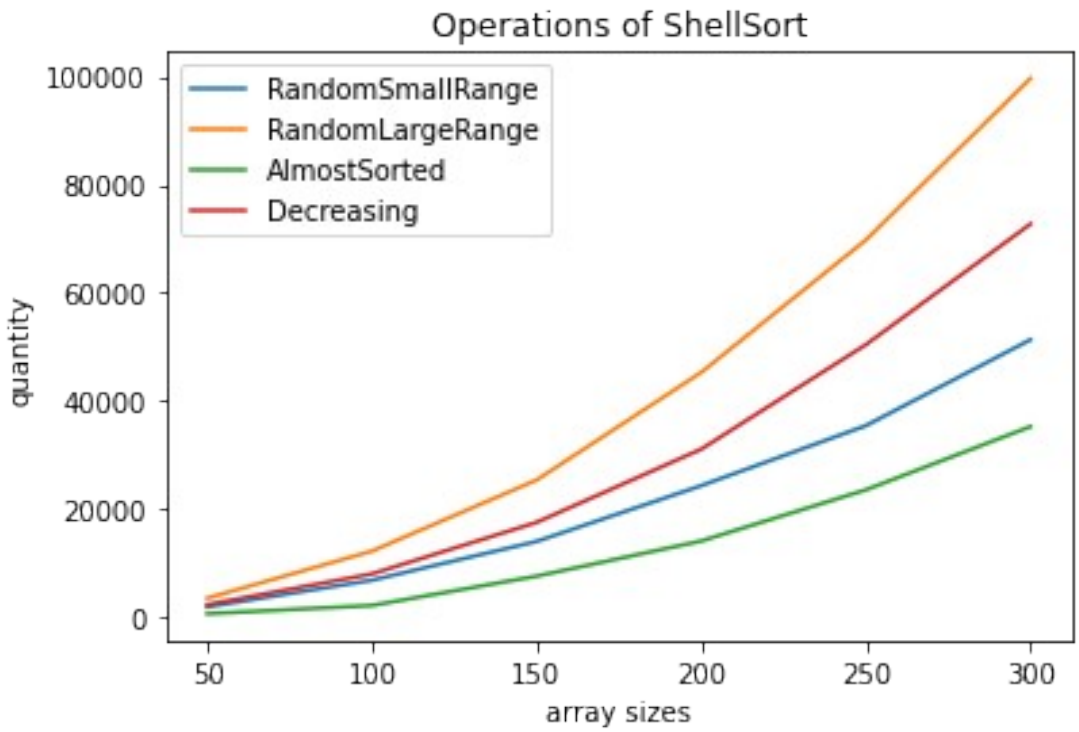
<Figure size 5000x5000 with 0 Axes>



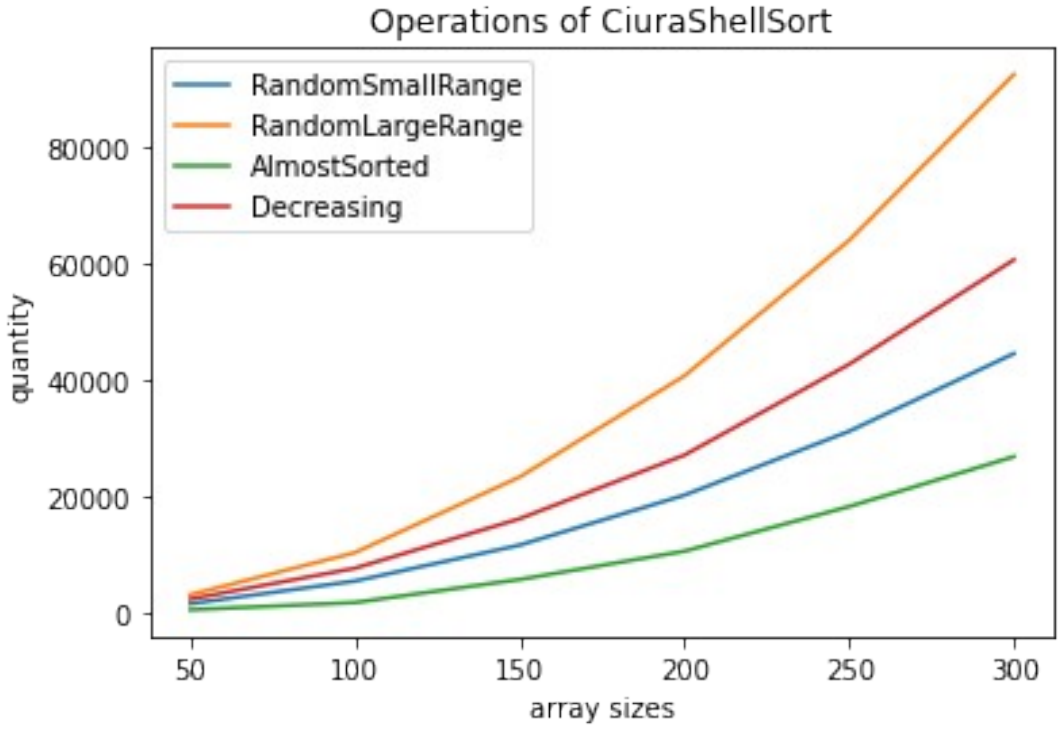
<Figure size 5000x5000 with 0 Axes>



<Figure size 5000x5000 with 0 Axes>



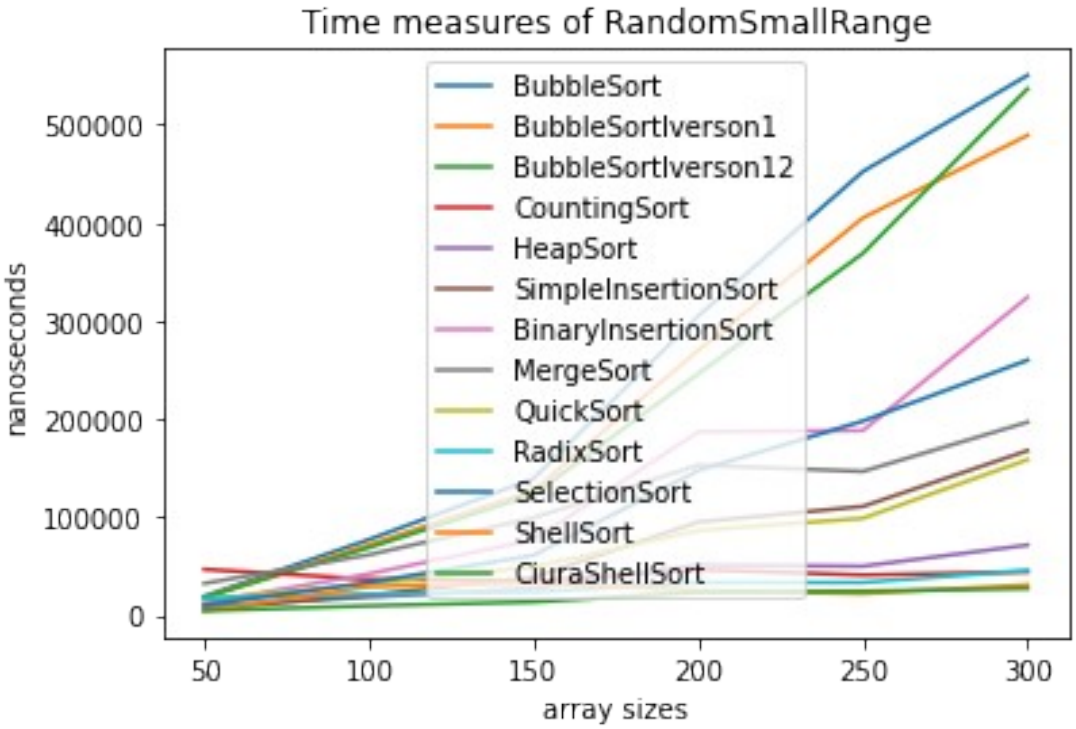
<Figure size 5000x5000 with 0 Axes>



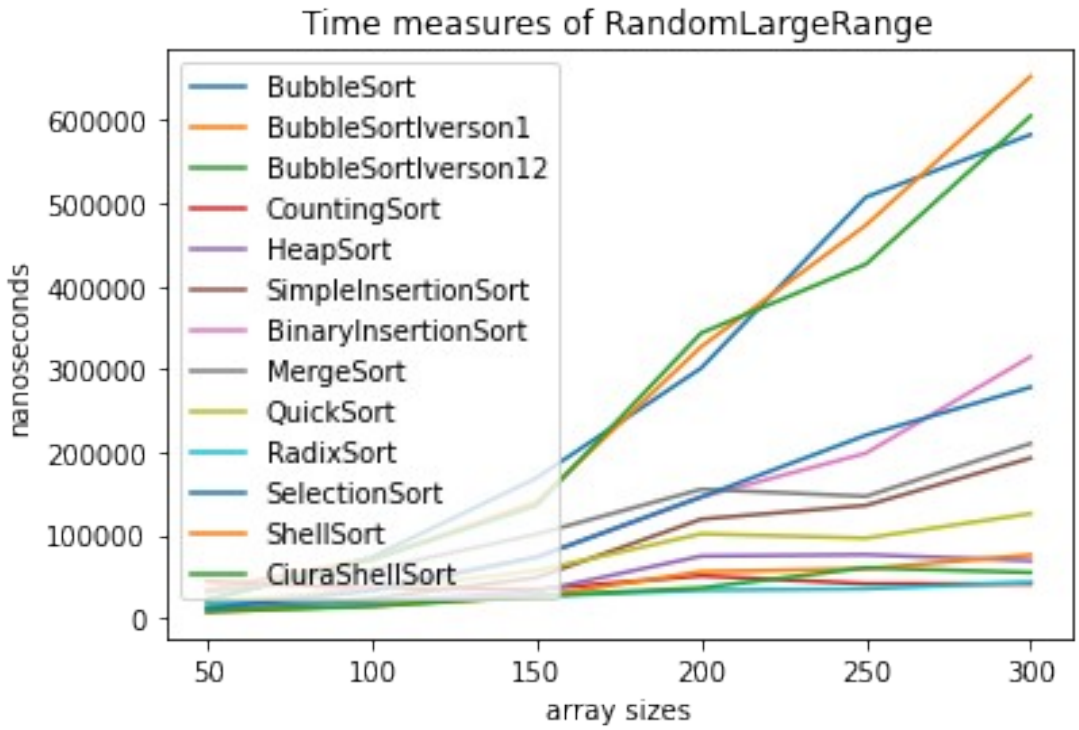

```
#@title Графики измеренного времени по всем сортировкам для конкретной генерации на маленьких массивах
# small size range, for all generations, time
```

```
for curr in gens:
  plt.figure(figsize=(10, 10),dpi=500)
  fig, ax = plt.subplots()

  for sort in sorts:
      plt.plot(times_small["Array sizes"], times_small[curr + " & " +
sort], label=sort)
  ax.set(xlabel='array sizes', ylabel='nanoseconds',
        title='Time measures of ' + curr)

  fig.savefig(curr + "_time.png")
  plt.legend()
  plt.show()
```

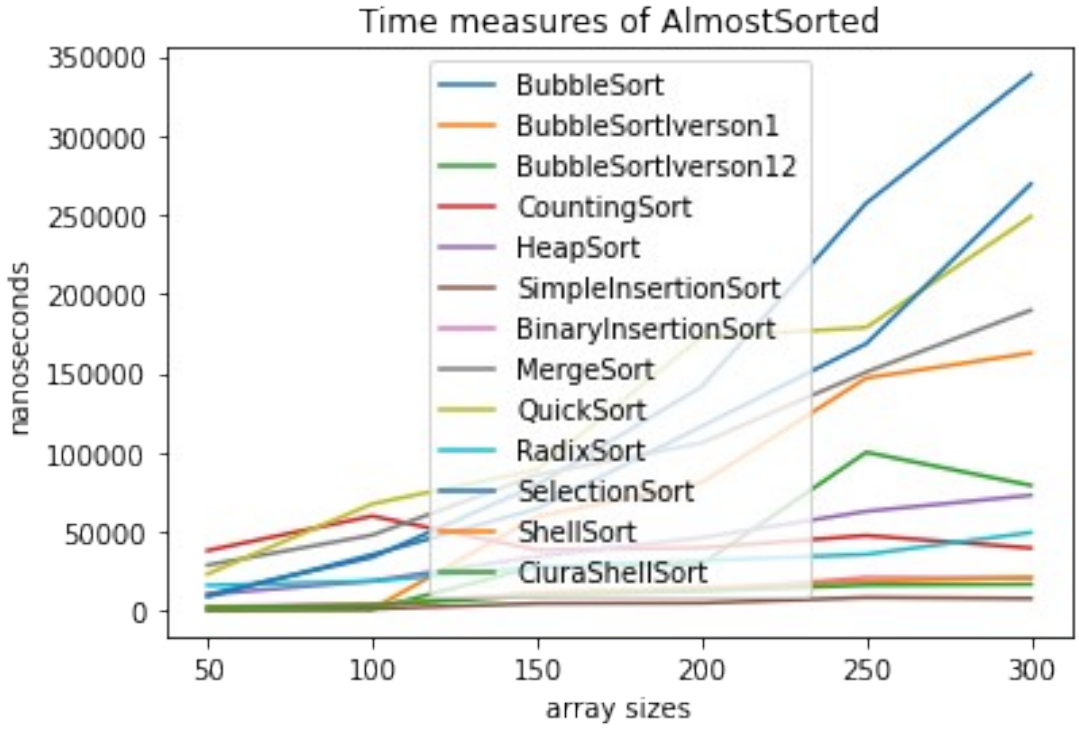
```
<Figure size 5000x5000 with 0 Axes>
```



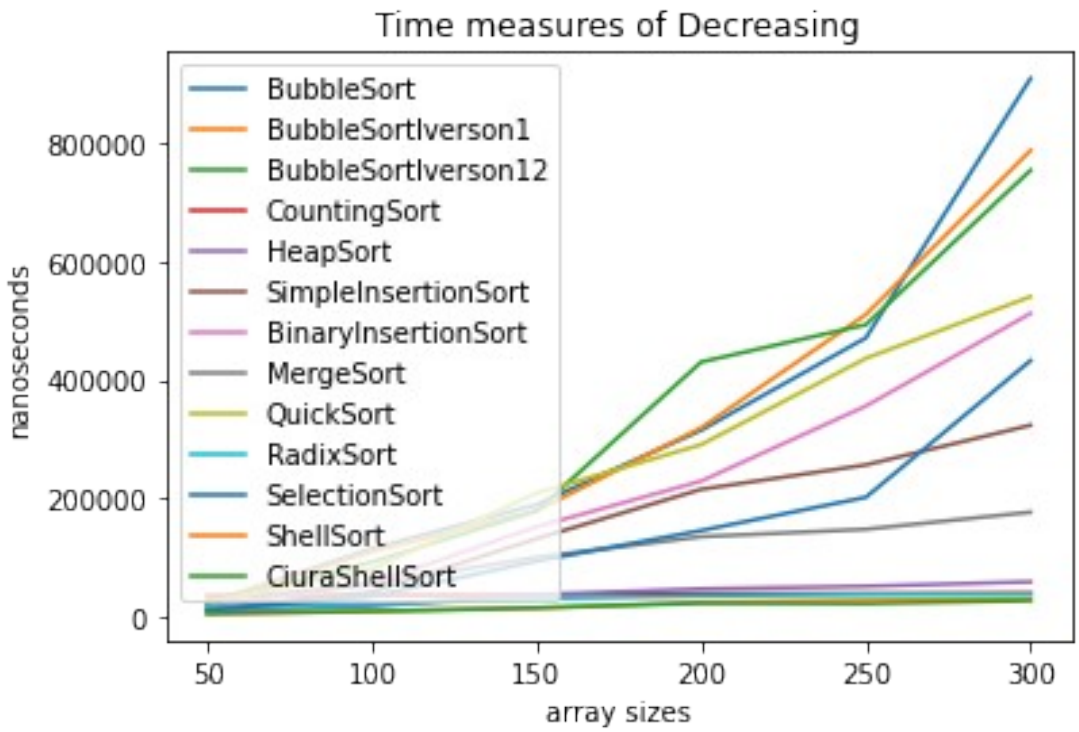

```
<Figure size 5000x5000 with 0 Axes>
```



<Figure size 5000x5000 with 0 Axes>

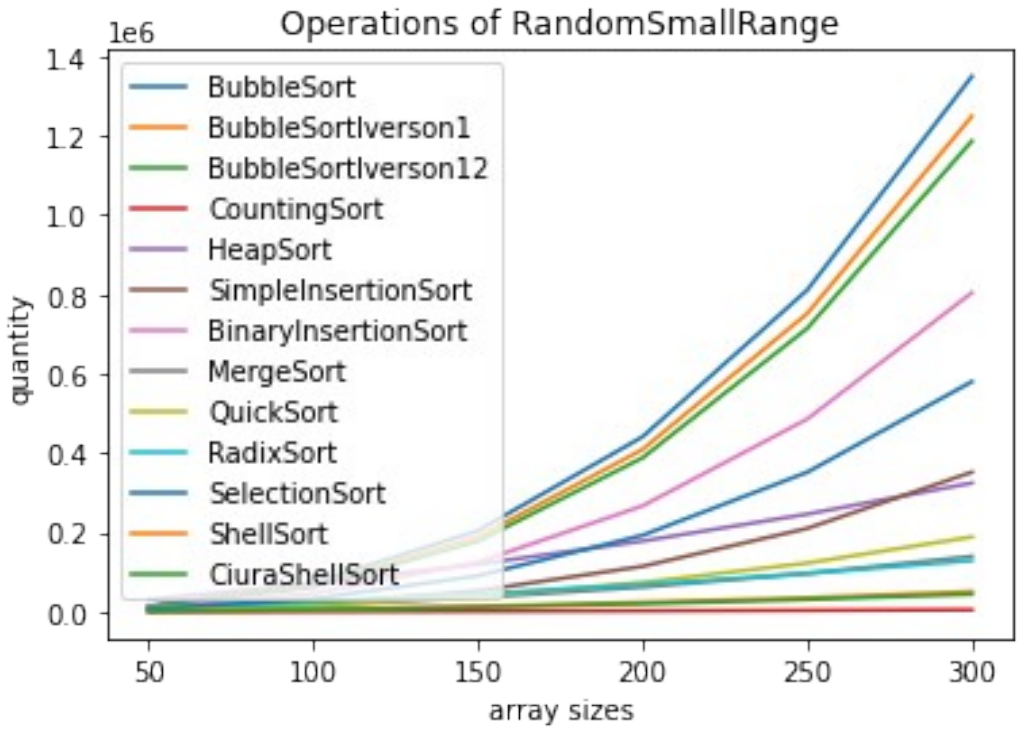


<Figure size 5000x5000 with 0 Axes>

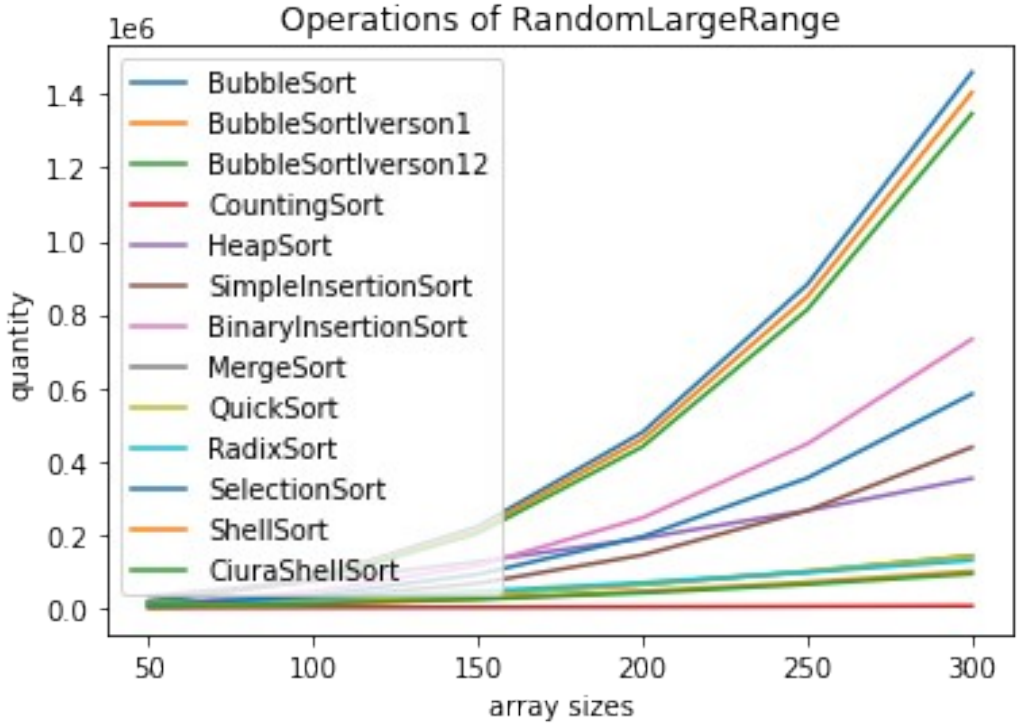



```
#@title Графики количества операций по всем сортировкам для конкретной генерации на маленьких массивах
# small size range, for all generations, operations
for curr in gens:
  plt.figure(figsize=(10, 10),dpi=500)
  fig, ax = plt.subplots()

  for sort in sorts:
      plt.plot(oper_small["Array sizes"], oper_small[curr + " & " + sort], label=sort)
  ax.set(xlabel='array sizes', ylabel='quantity',
        title='Operations of ' + curr)

  fig.savefig(curr + "_time.png")
  plt.legend()
  plt.show()
```

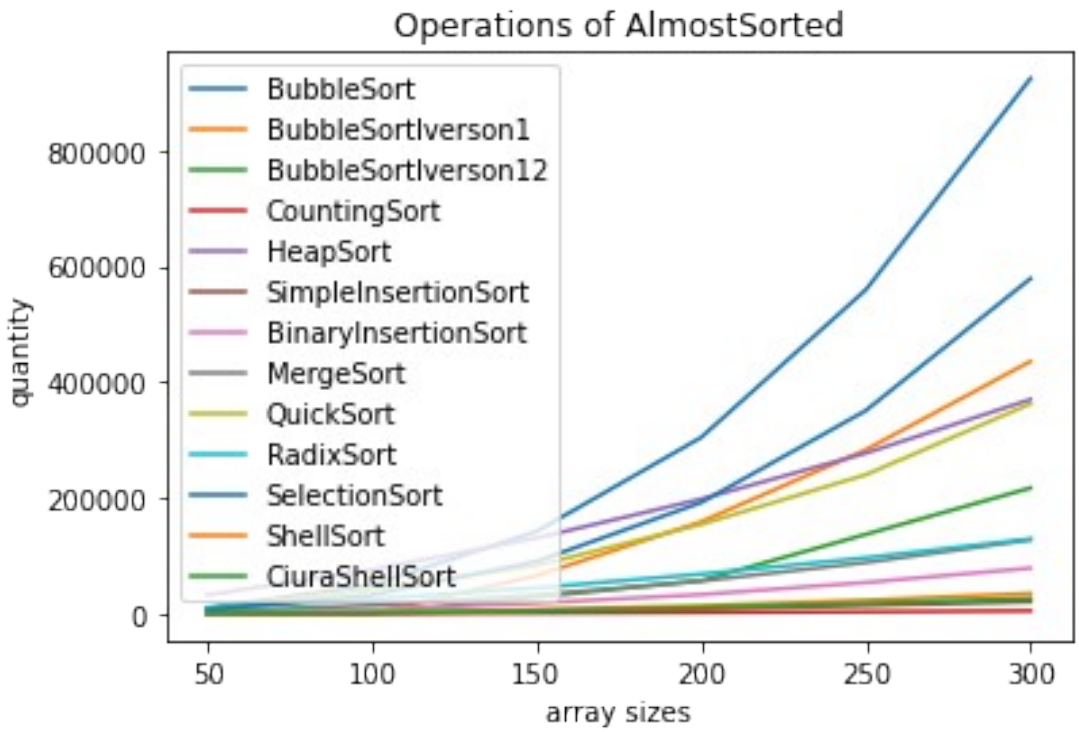
```
<Figure size 5000x5000 with 0 Axes>
```



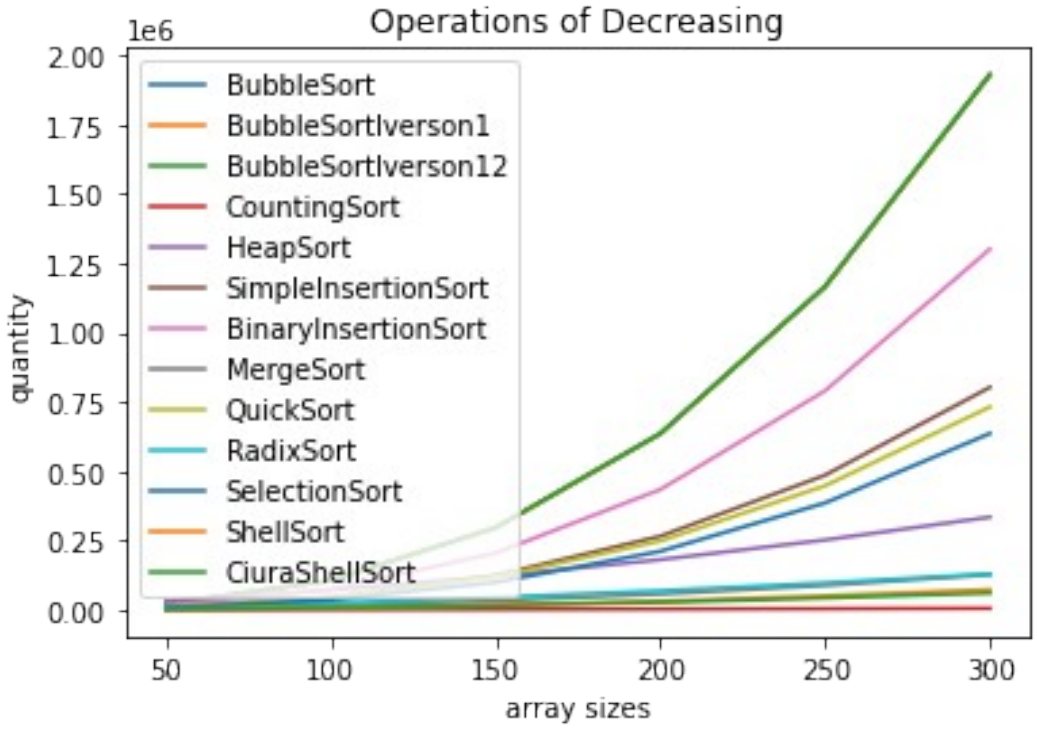
<Figure size 5000x5000 with 0 Axes>



<Figure size 5000x5000 with 0 Axes>



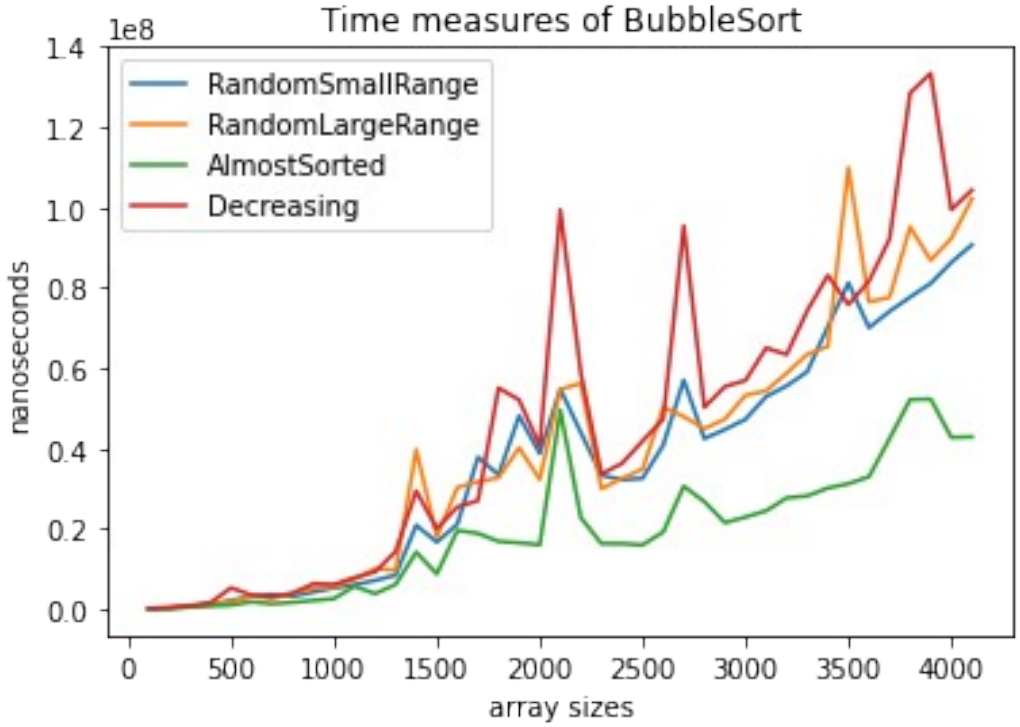

```
<Figure size 5000x5000 with 0 Axes>
```



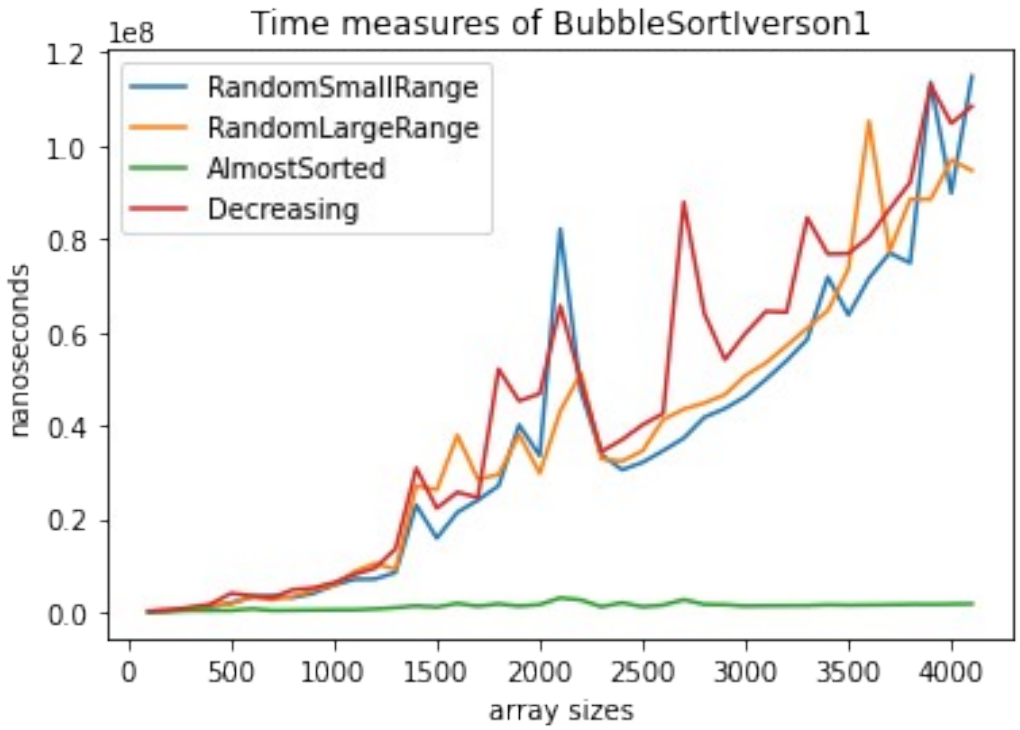

```
#@title Графики измеренного времени по всем генерациям для конкретной сортировки на больших массивах
#small size range, for all sorts, time
```

```
for curr in sorts:
  plt.figure(figsize=(10, 10),dpi=500)
  fig, ax = plt.subplots()

  for gen in gens:
      plt.plot(times_big["Array sizes"], times_big[gen + " & " +
curr], label=gen)
  ax.set(xlabel='array sizes', ylabel='nanoseconds',
        title='Time measures of ' + curr)

  fig.savefig(curr + "_time.png")
  plt.legend()
  plt.show()
```

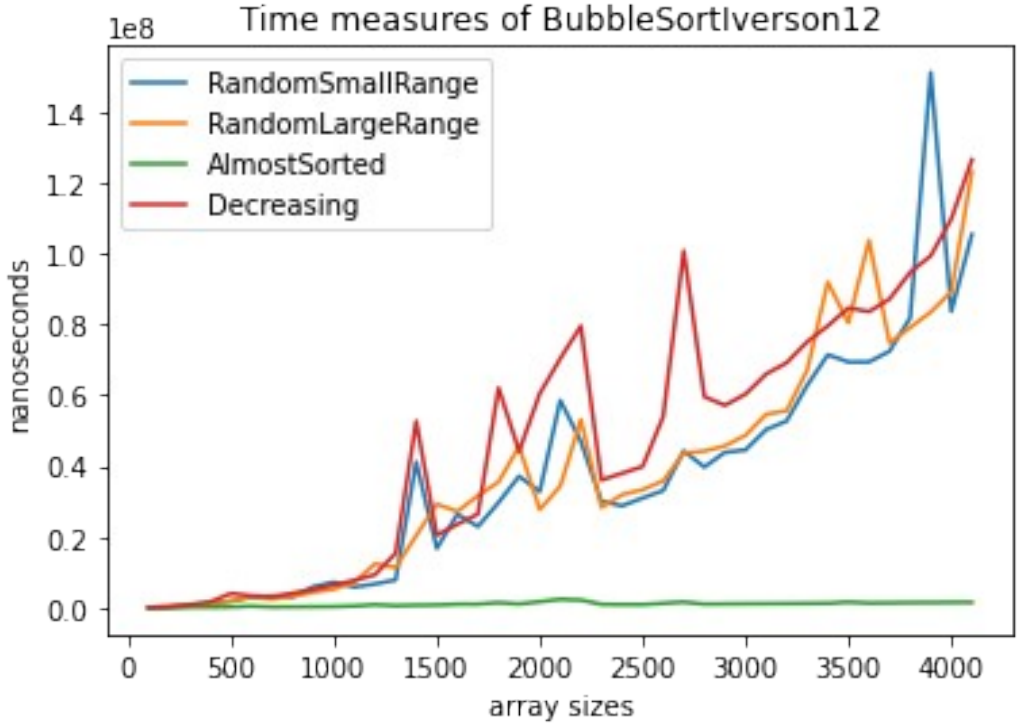
```
<Figure size 5000x5000 with 0 Axes>
```



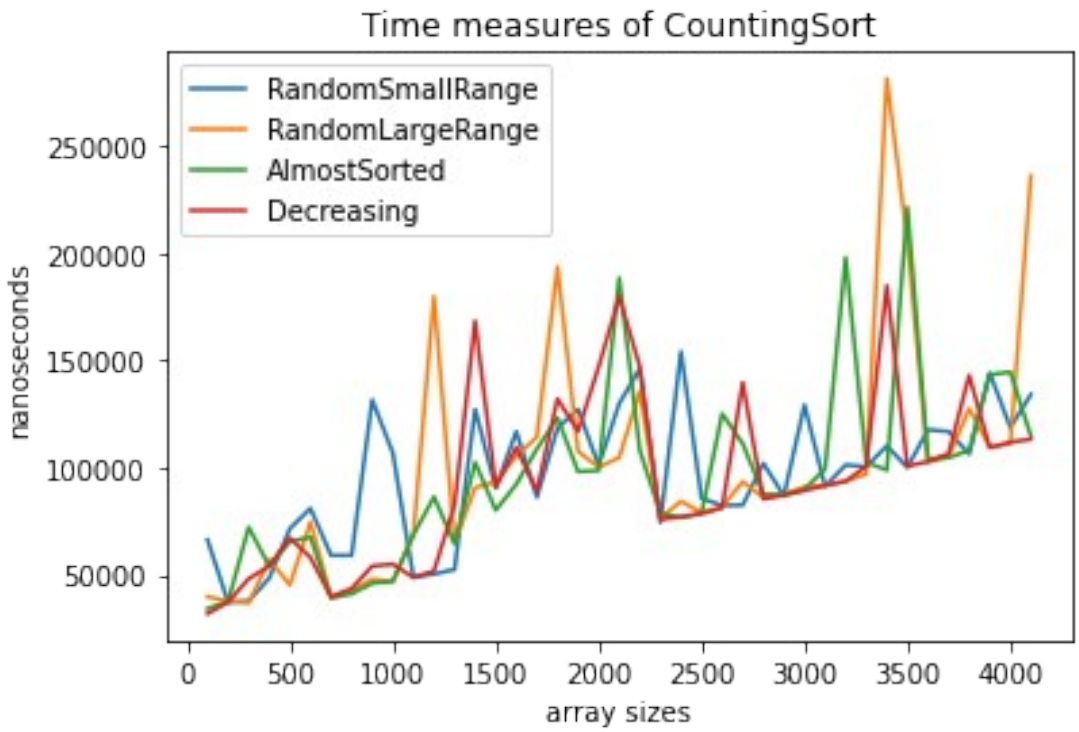

```
<Figure size 5000x5000 with 0 Axes>
```



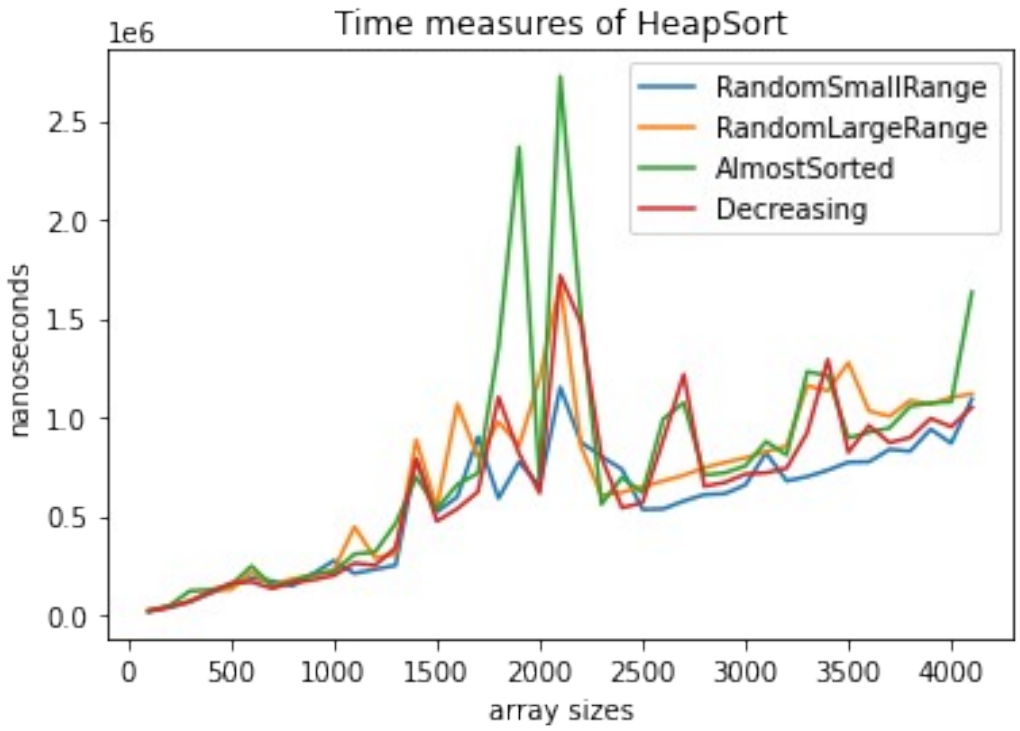
<Figure size 5000x5000 with 0 Axes>



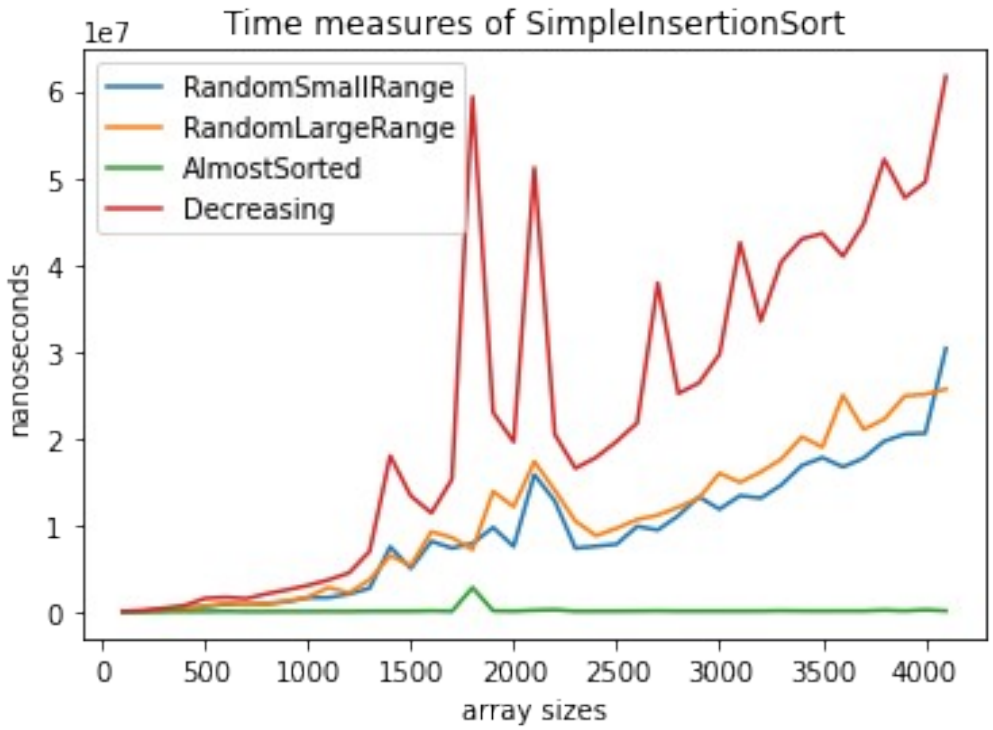
<Figure size 5000x5000 with 0 Axes>



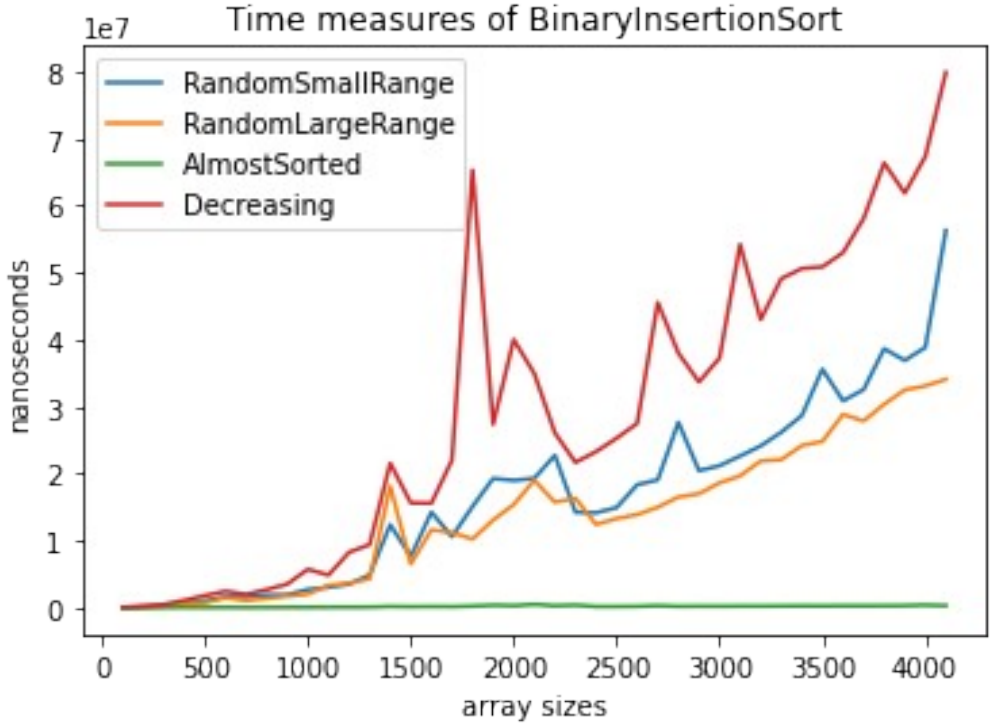
<Figure size 5000x5000 with 0 Axes>



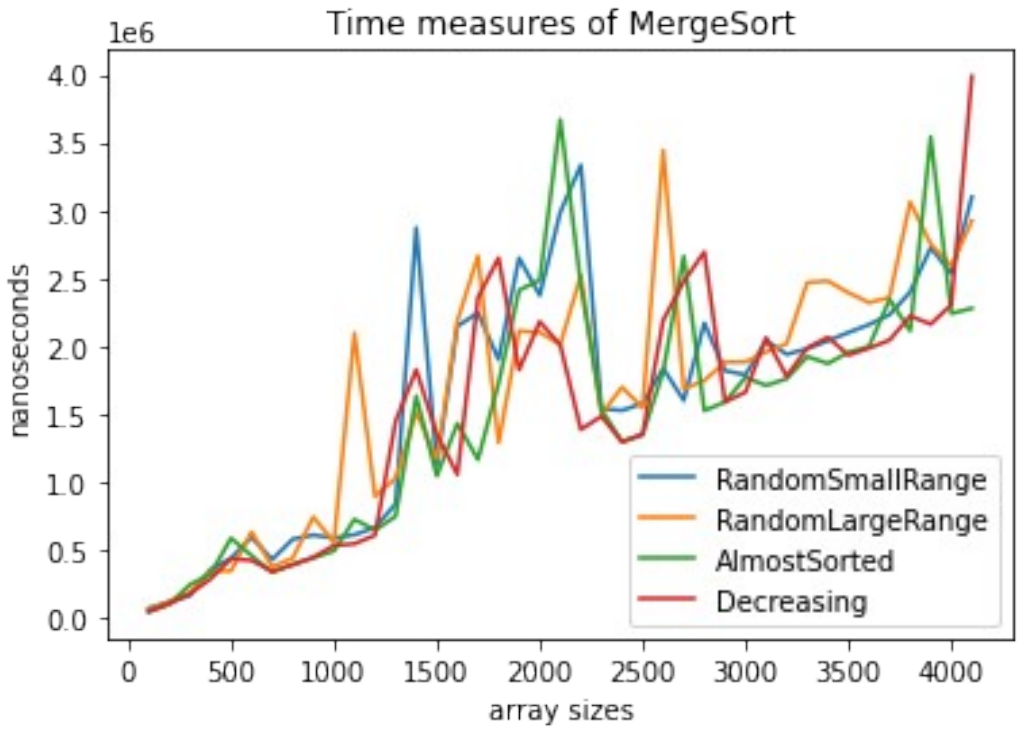
<Figure size 5000x5000 with 0 Axes>



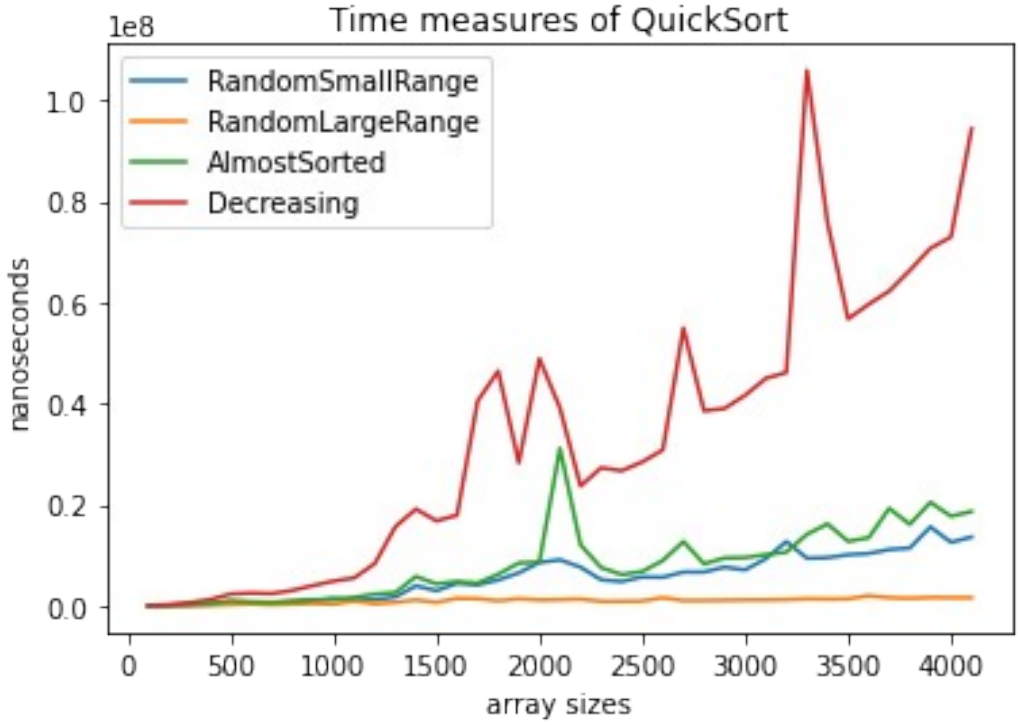
<Figure size 5000x5000 with 0 Axes>



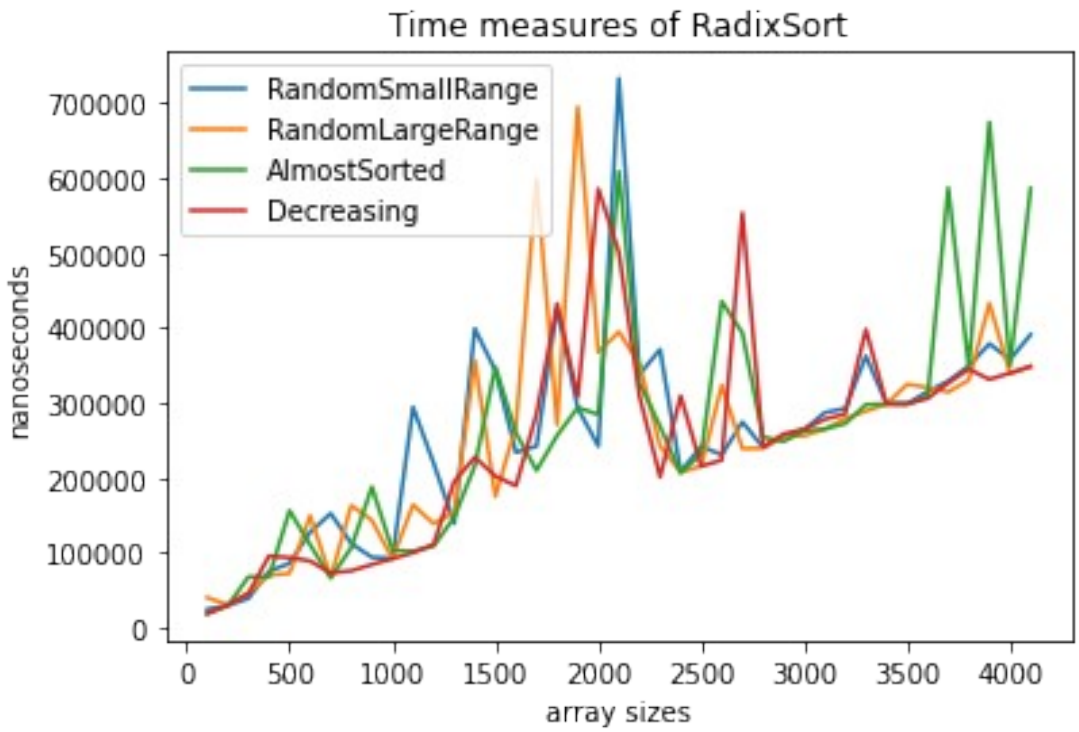
<Figure size 5000x5000 with 0 Axes>



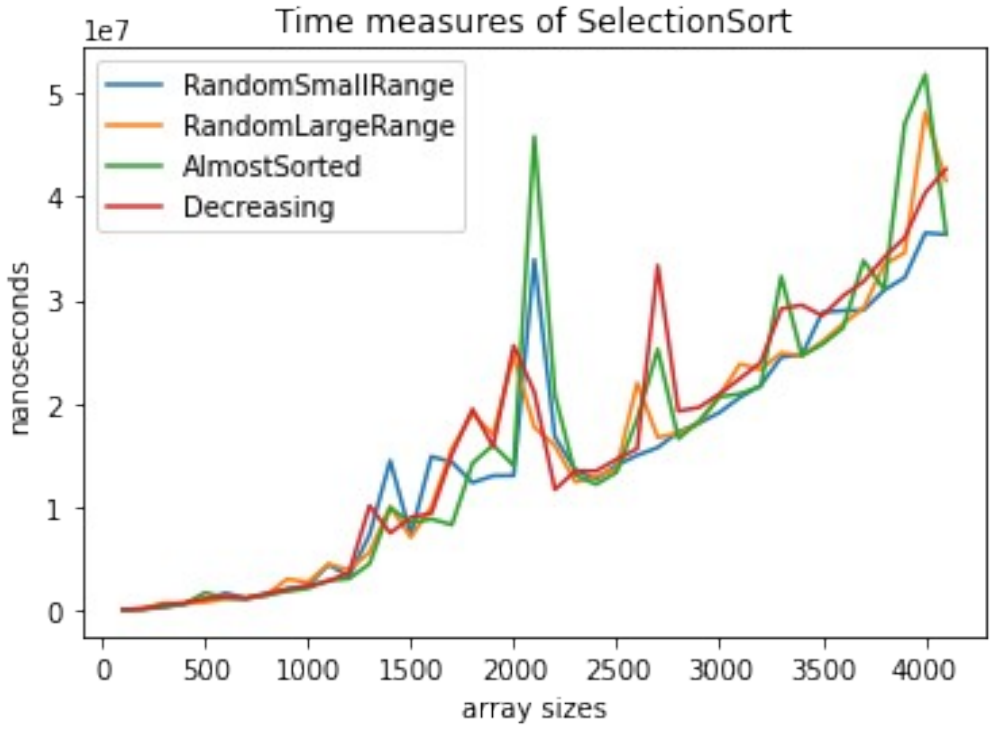
<Figure size 5000x5000 with 0 Axes>



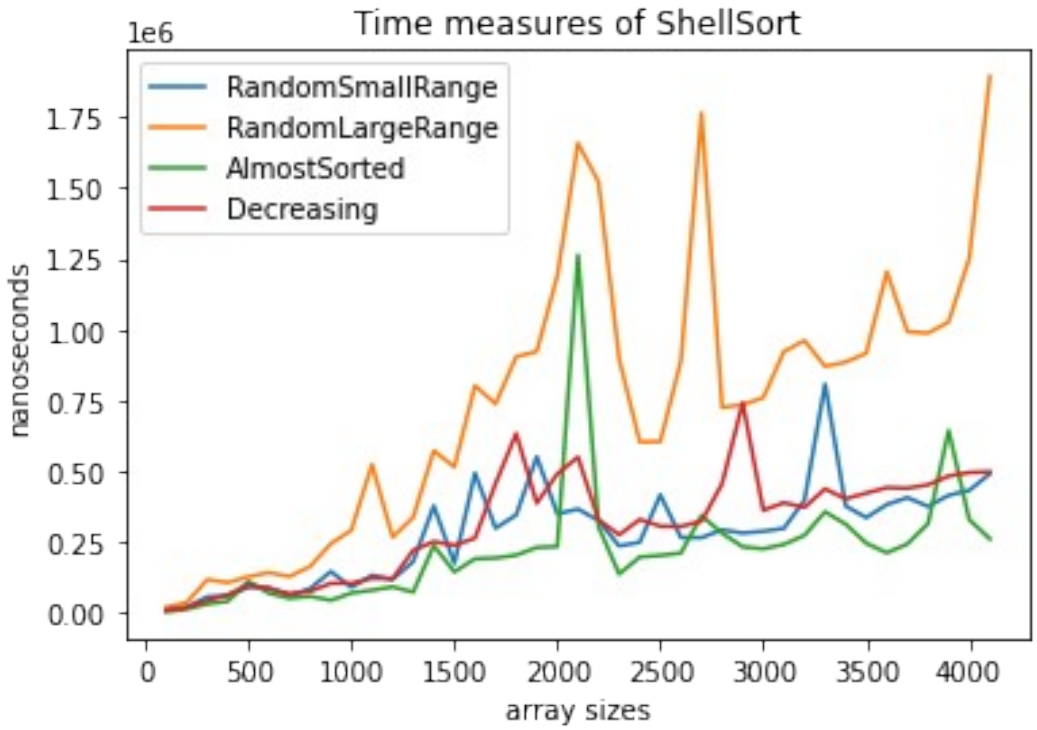
<Figure size 5000x5000 with 0 Axes>



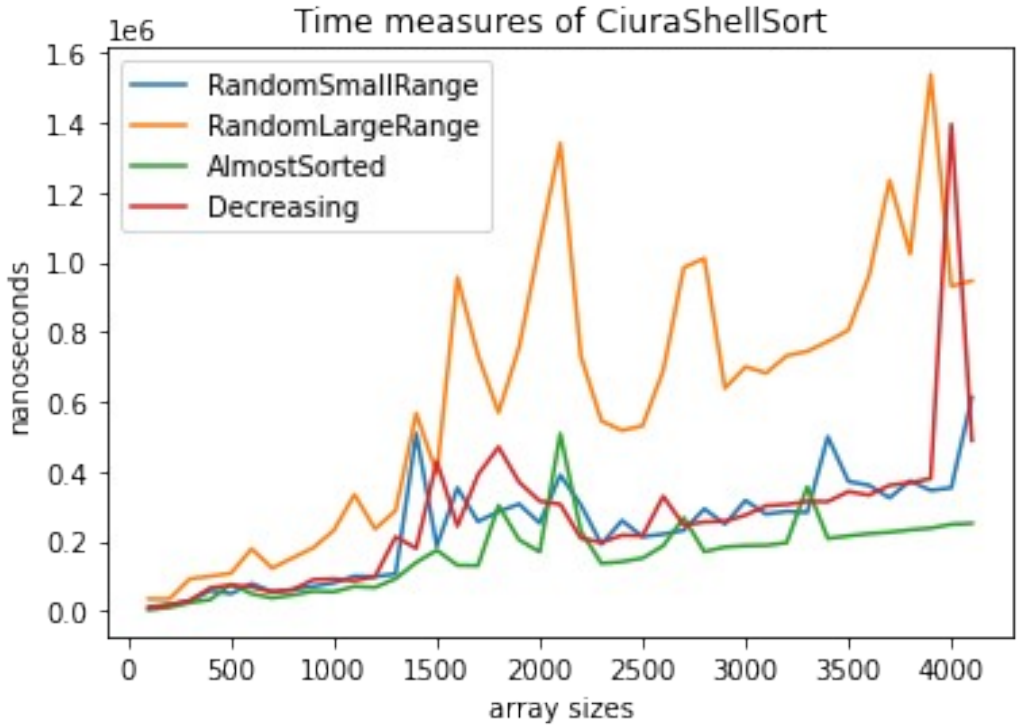
<Figure size 5000x5000 with 0 Axes>



<Figure size 5000x5000 with 0 Axes>



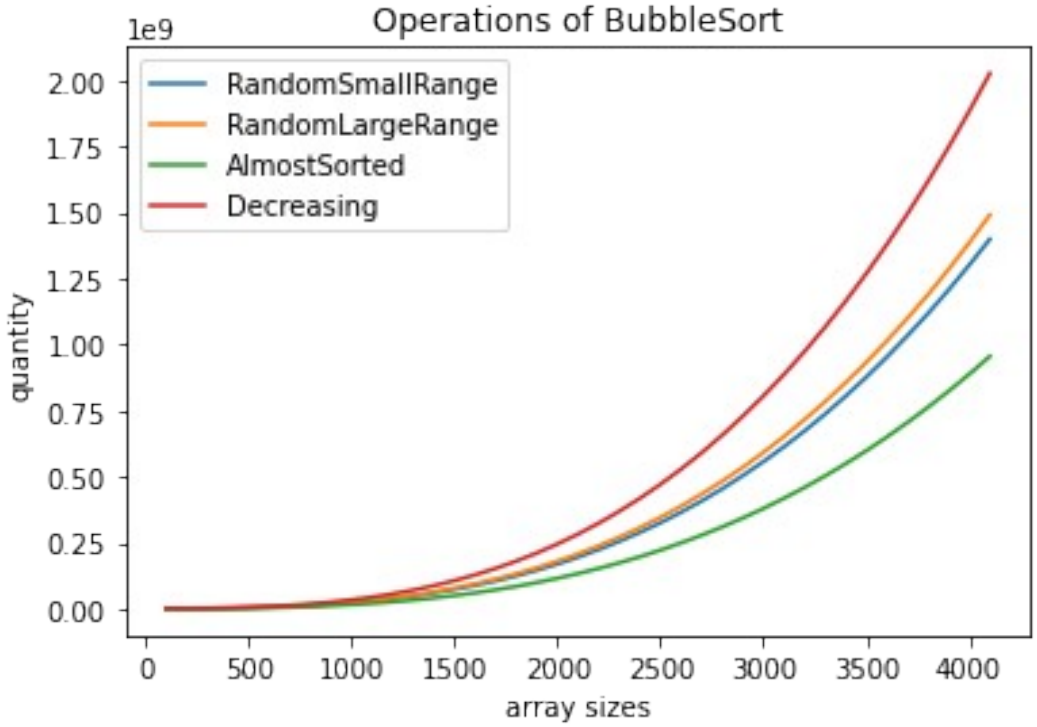

```
<Figure size 5000x5000 with 0 Axes>
```



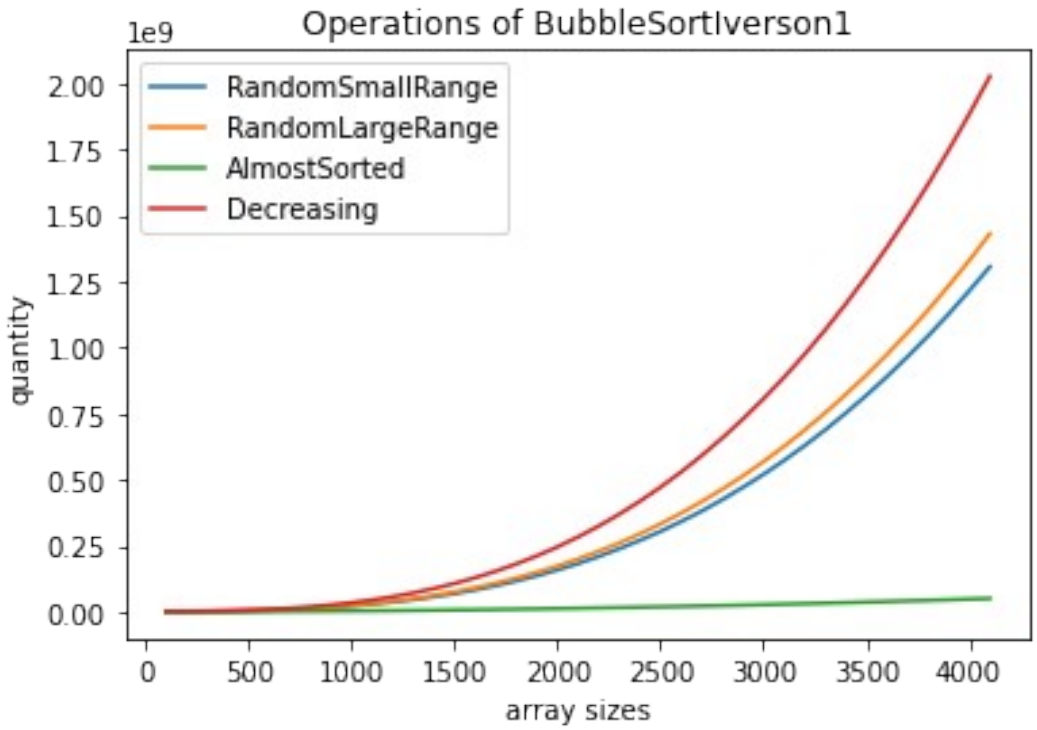

```
#@title Графики количества операций по всем генерациям для конкретной сортировки на больших массивах
#small size range, for all sorts, operations
```

```
for curr in sorts:
  plt.figure(figsize=(10, 10),dpi=500)
  fig, ax = plt.subplots()

  for gen in gens:
      plt.plot(oper_big["Array sizes"], oper_big[gen + " & " + curr],
label=gen)
  ax.set(xlabel='array sizes', ylabel='quantity',
        title='Operations of ' + curr)

  fig.savefig(curr + "_time.png")
  plt.legend()
  plt.show()
```

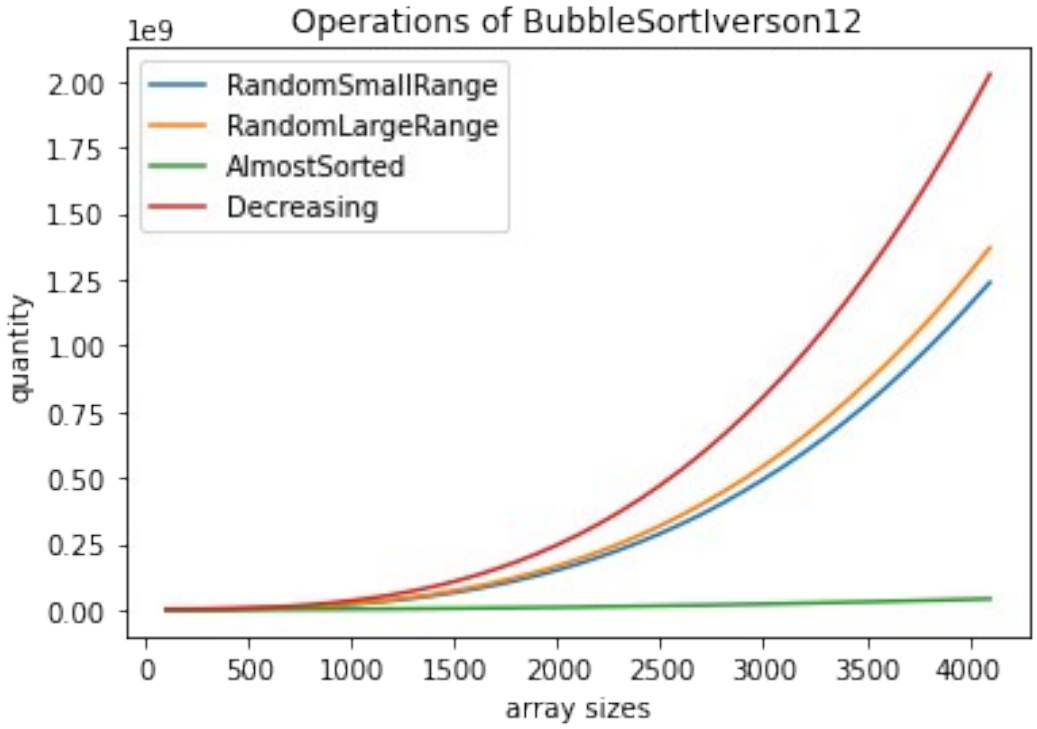
```
<Figure size 5000x5000 with 0 Axes>
```



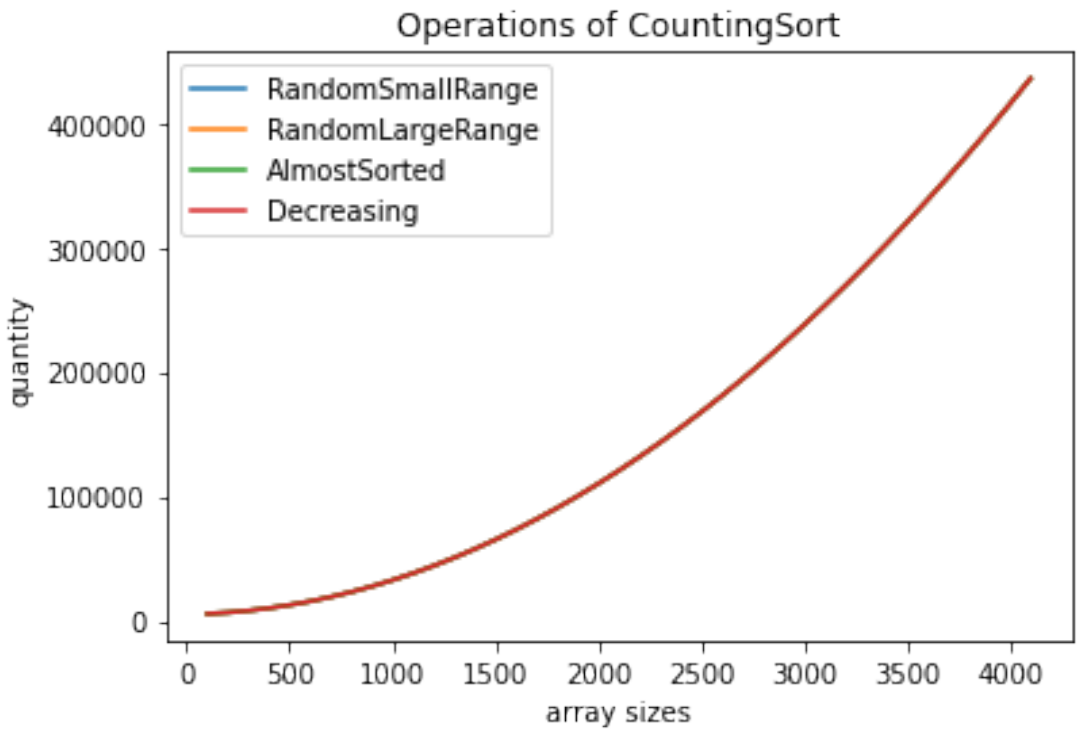

```
<Figure size 5000x5000 with 0 Axes>
```



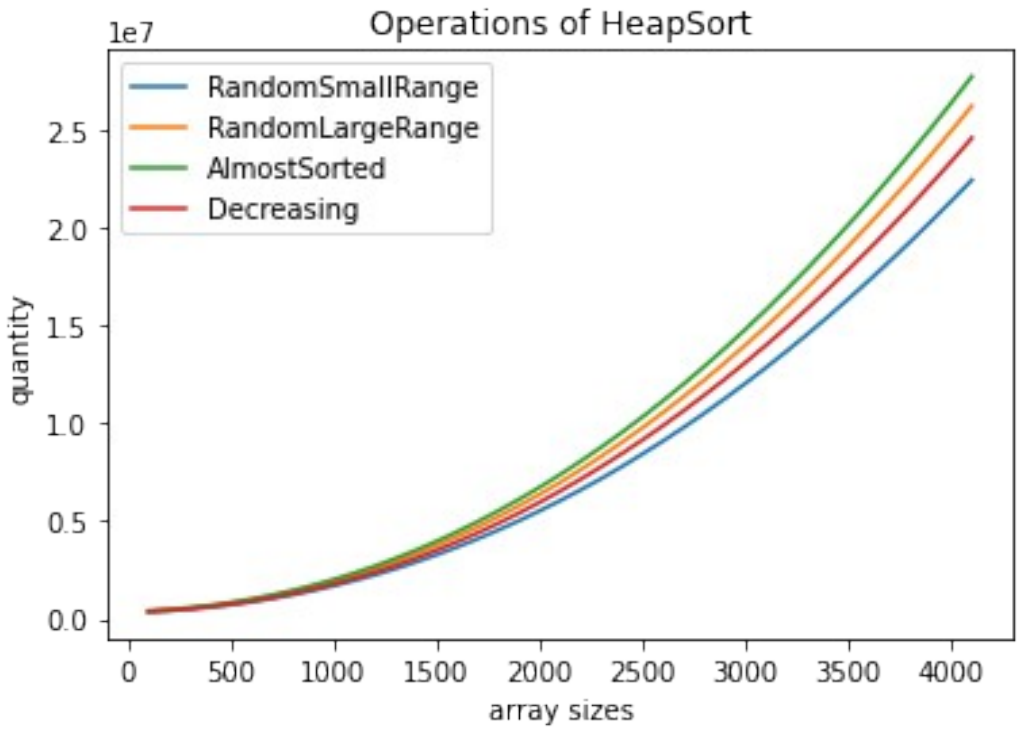
<Figure size 5000x5000 with 0 Axes>



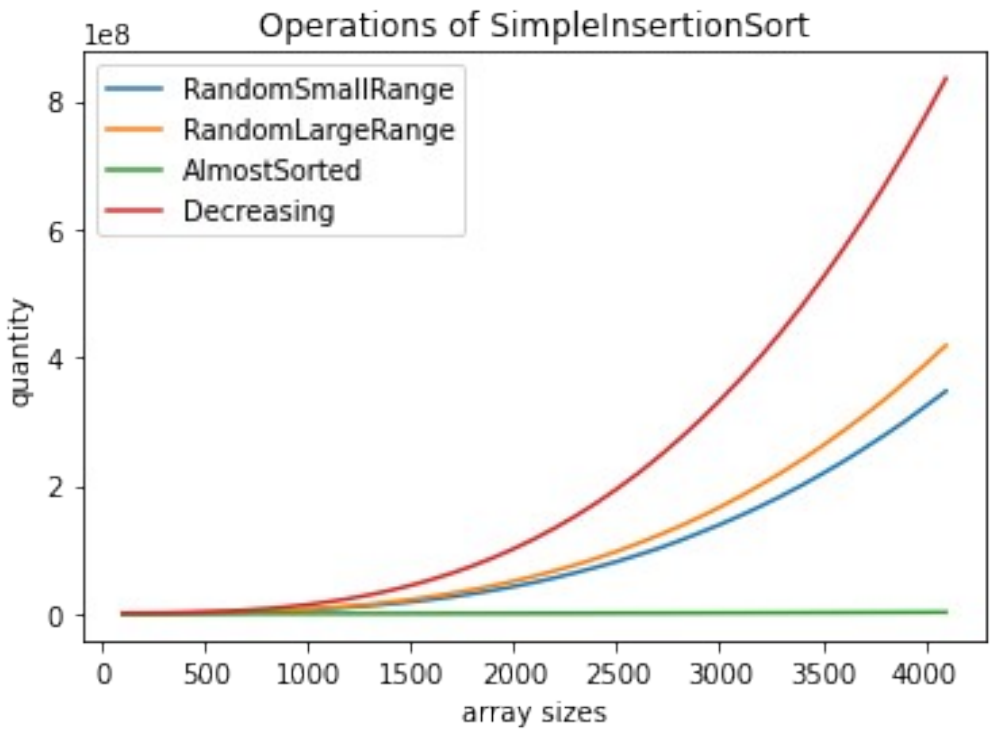
<Figure size 5000x5000 with 0 Axes>



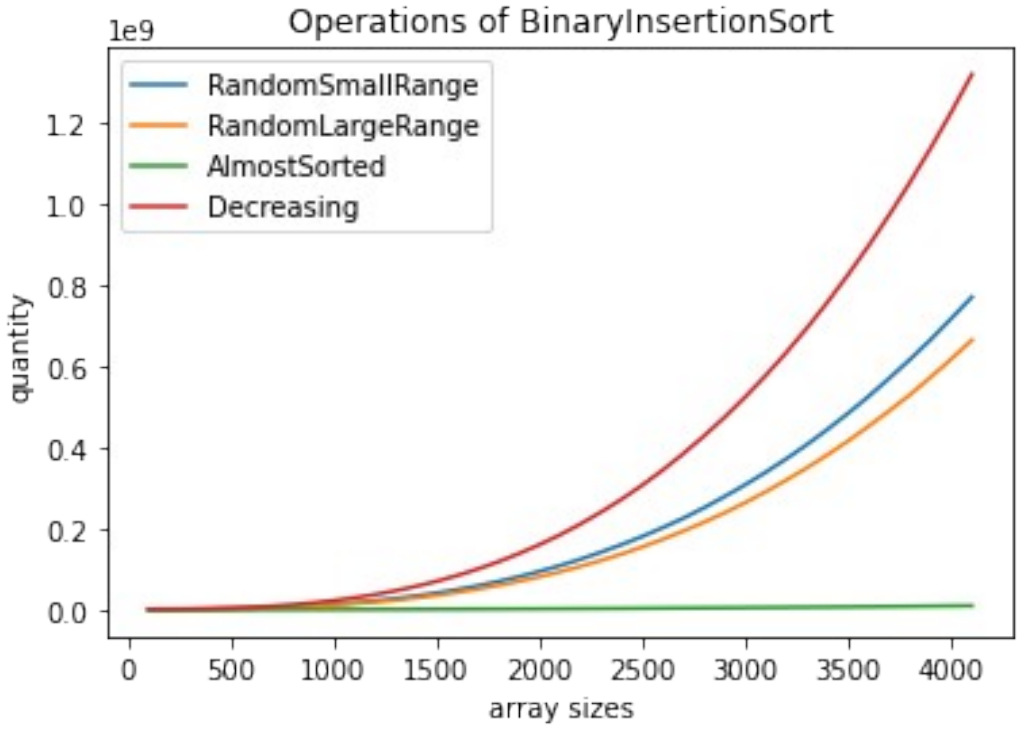
<Figure size 5000x5000 with 0 Axes>



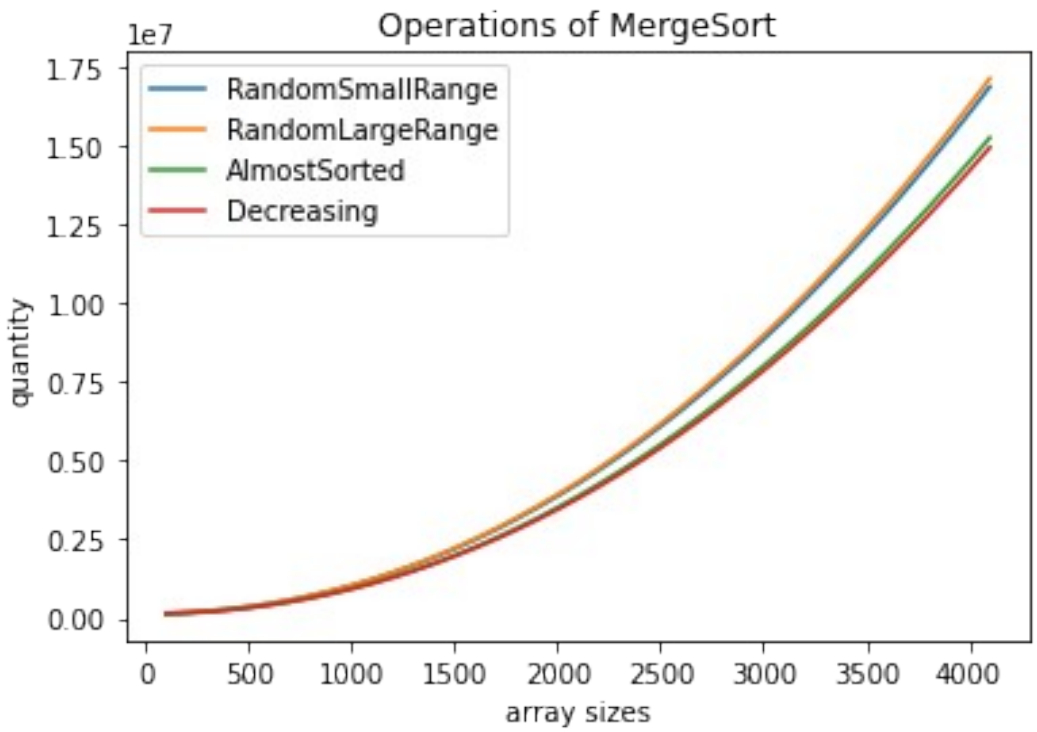
<Figure size 5000x5000 with 0 Axes>



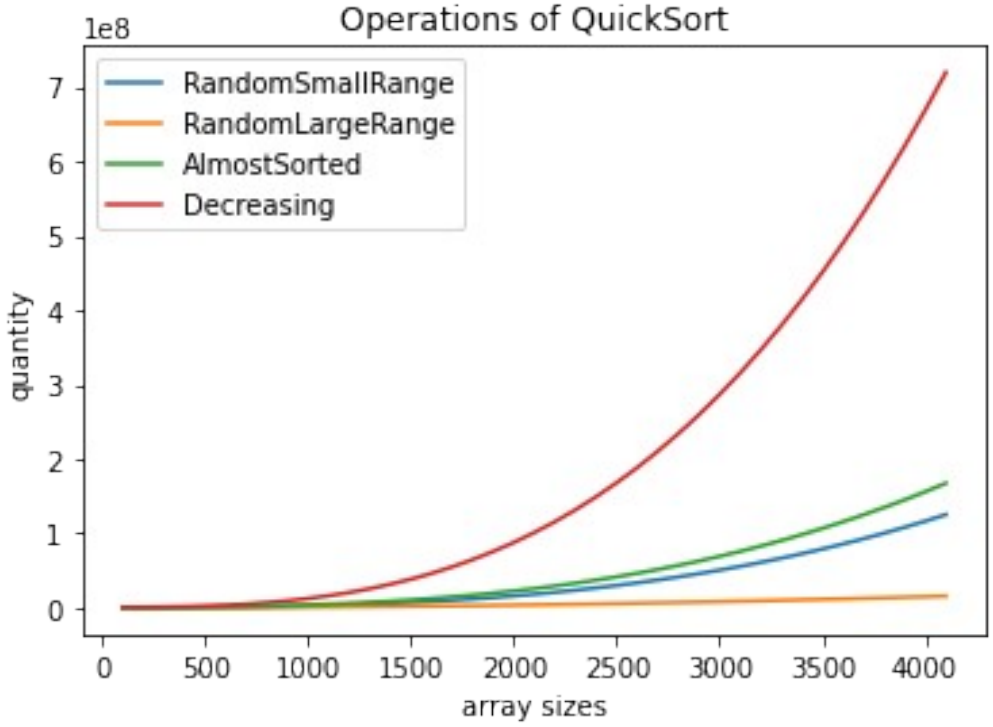
<Figure size 5000x5000 with 0 Axes>



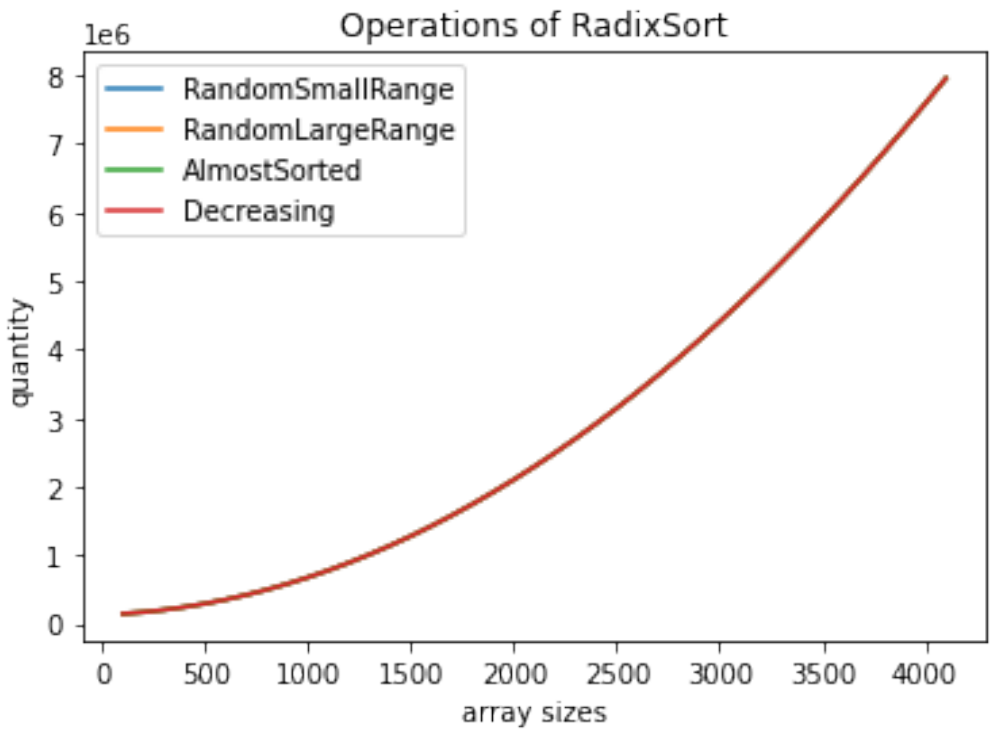
<Figure size 5000x5000 with 0 Axes>



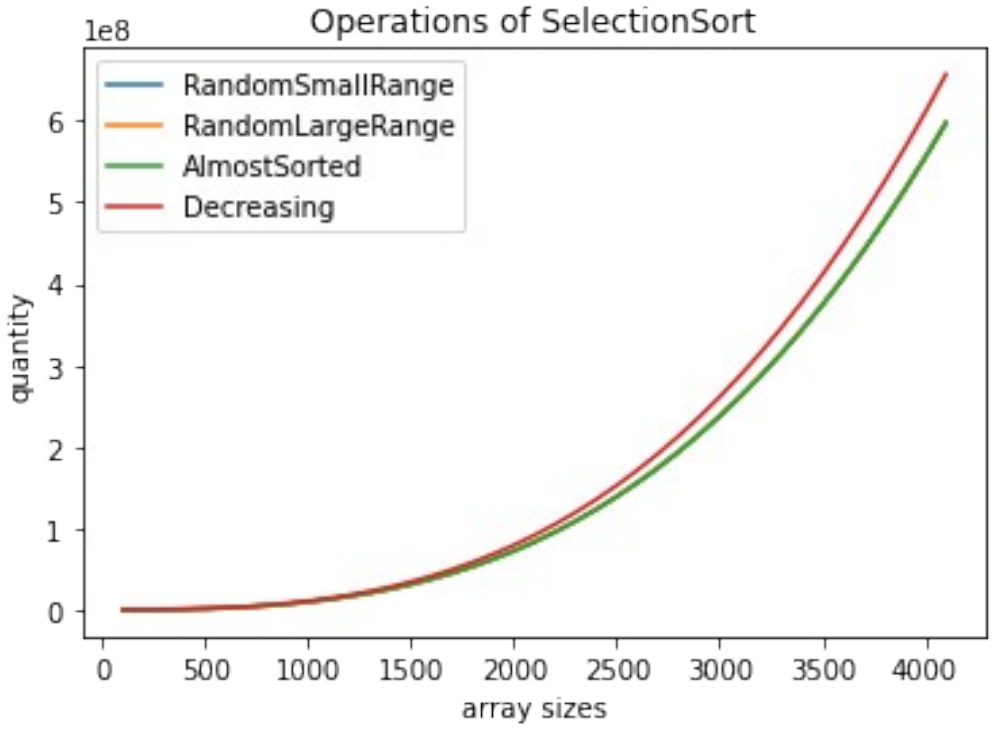
<Figure size 5000x5000 with 0 Axes>



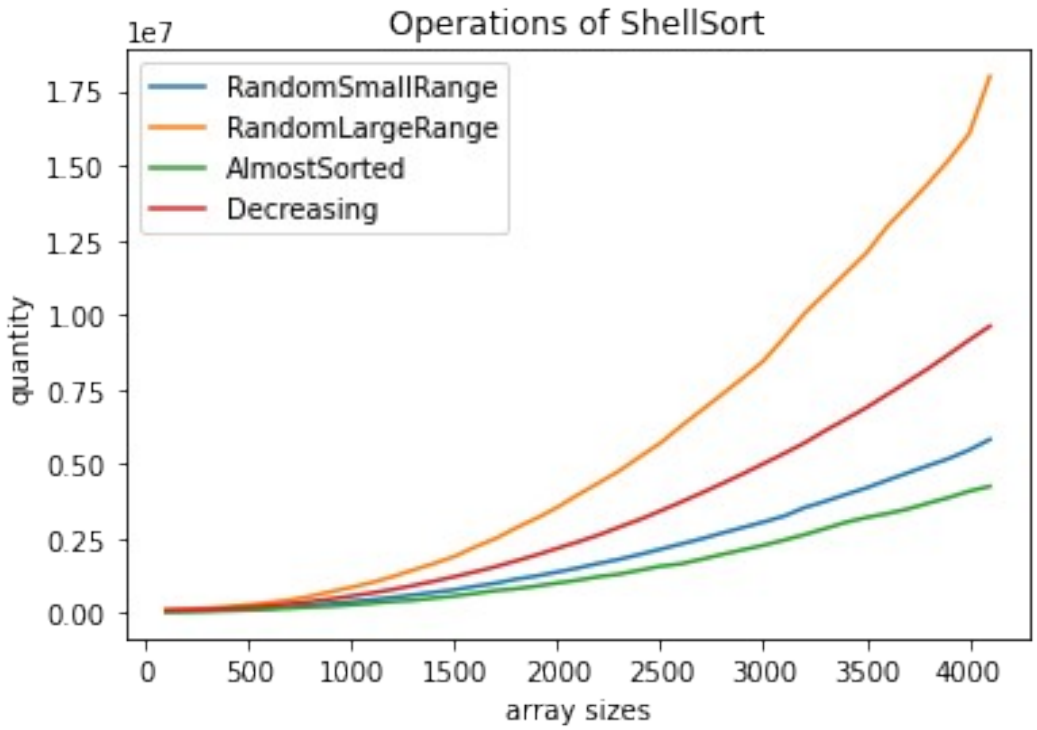
<Figure size 5000x5000 with 0 Axes>



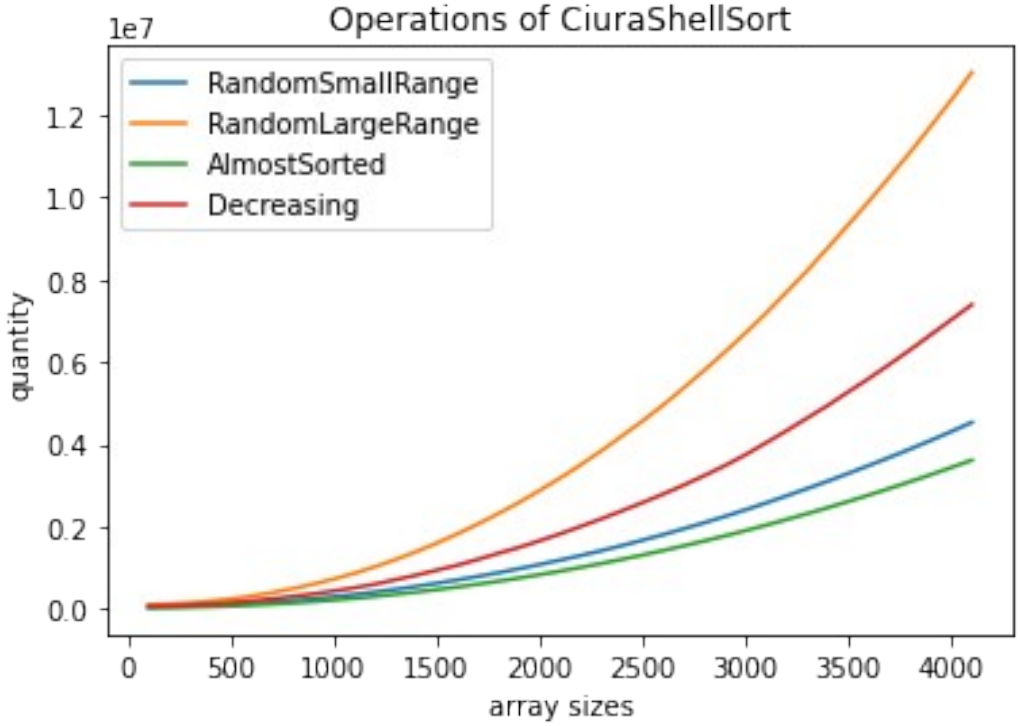
<Figure size 5000x5000 with 0 Axes>



<Figure size 5000x5000 with 0 Axes>



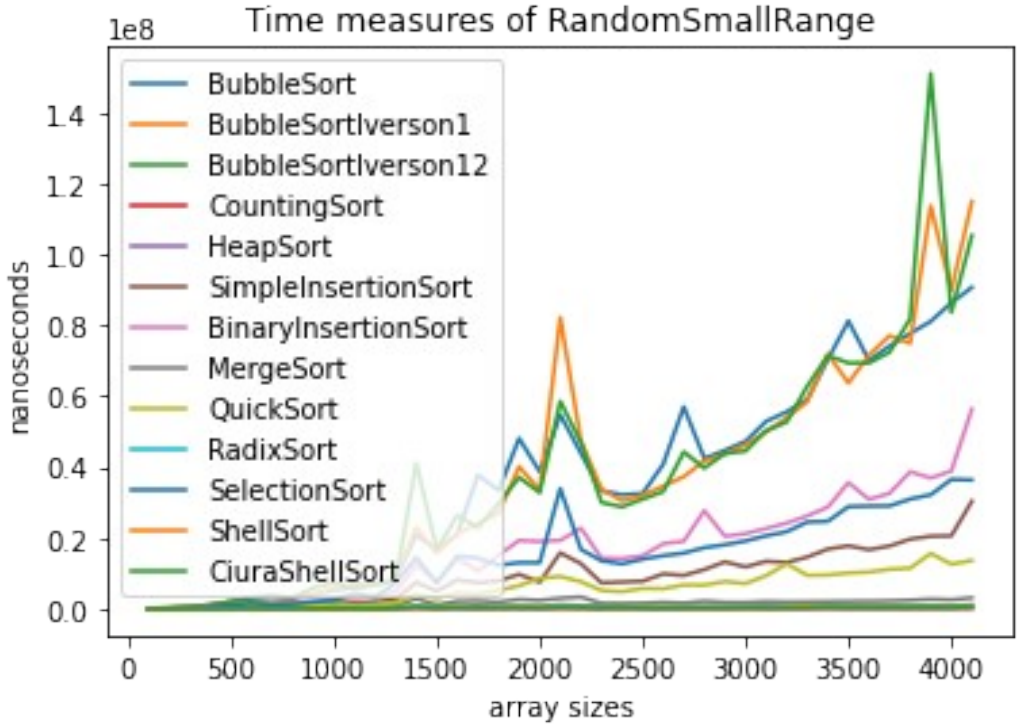

```
<Figure size 5000x5000 with 0 Axes>
```



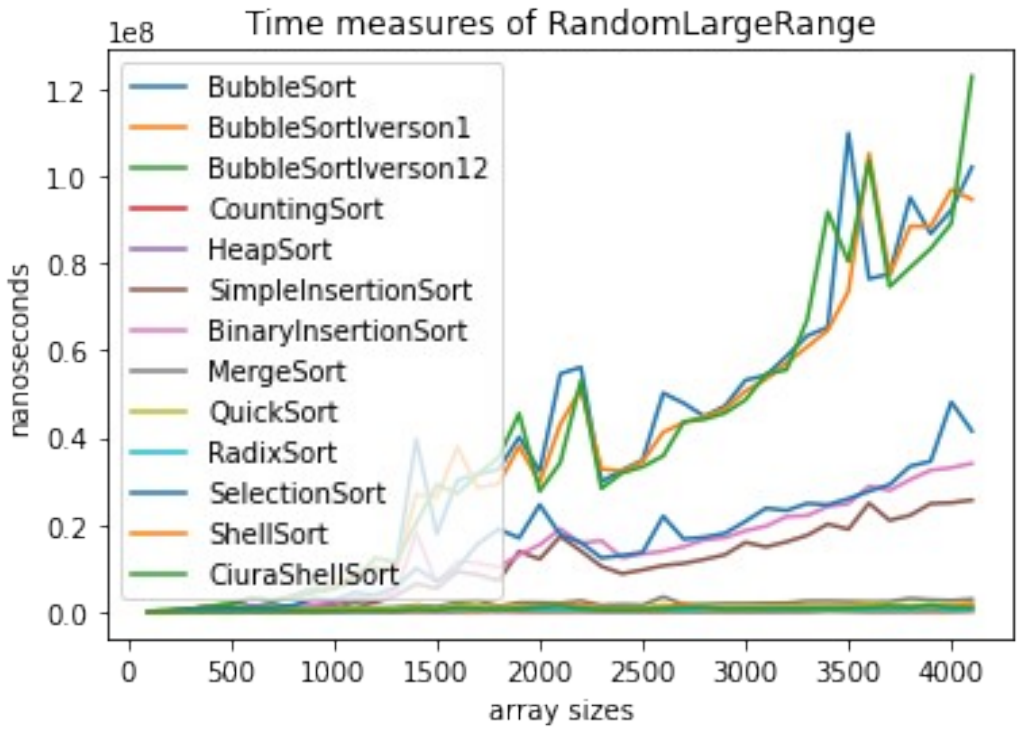

```
#@title Графики измеренного времени по всем сортировкам для конкретной генерации на больших массивах
# small size range, for all generations, time
```

```
for curr in gens:
  plt.figure(figsize=(10, 10),dpi=500)
  fig, ax = plt.subplots()

  for sort in sorts:
      plt.plot(times_big["Array sizes"], times_big[curr + " & " +
sort], label=sort)
  ax.set(xlabel='array sizes', ylabel='nanoseconds',
        title='Time measures of ' + curr)

  fig.savefig(curr + "_time.png")
  plt.legend()
  plt.show()
```

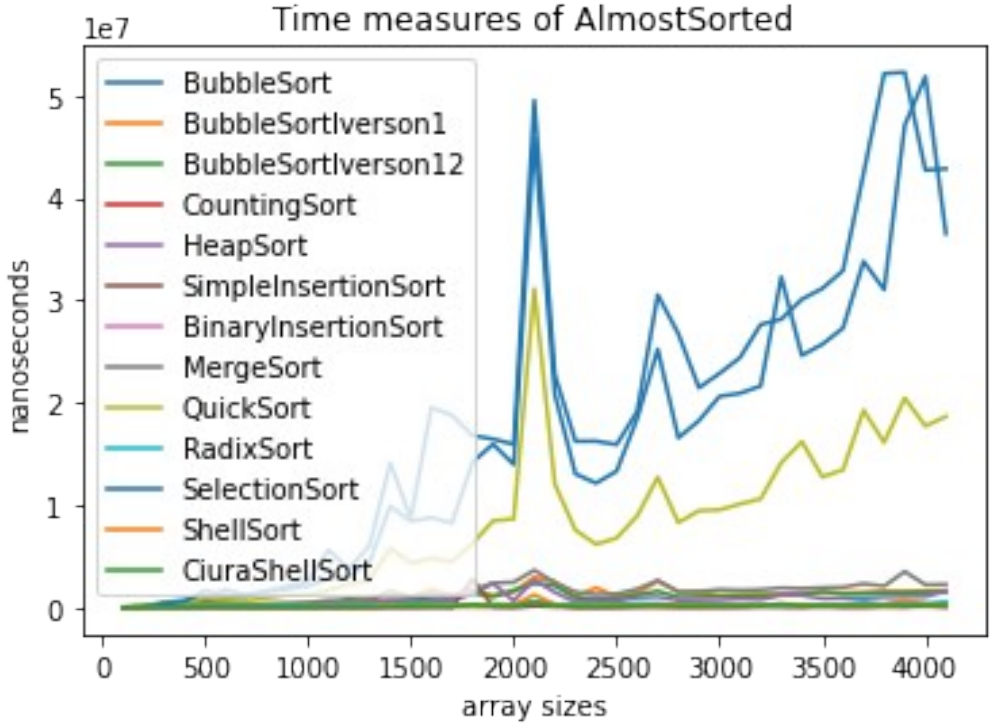
```
<Figure size 5000x5000 with 0 Axes>
```



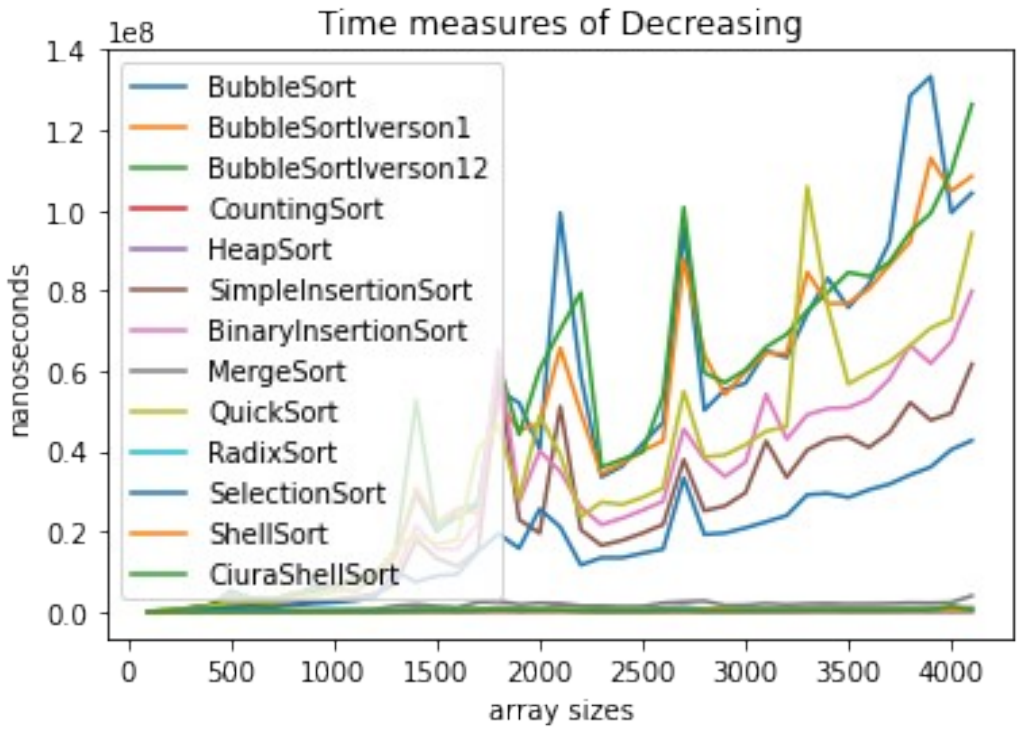

```
<Figure size 5000x5000 with 0 Axes>
```



<Figure size 5000x5000 with 0 Axes>

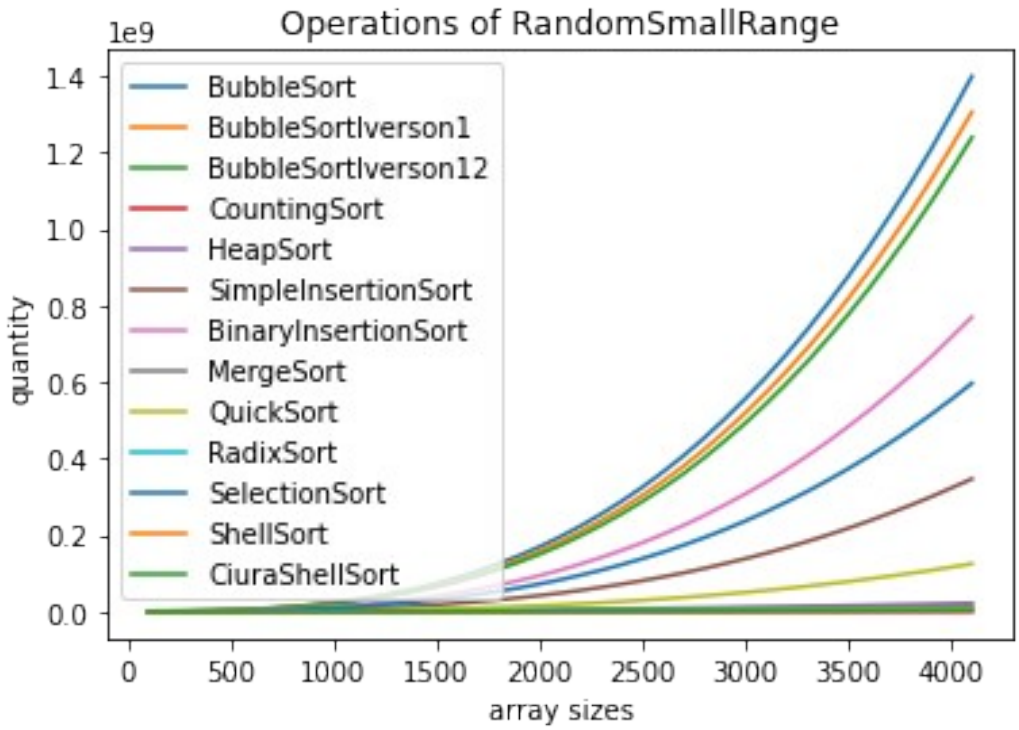


<Figure size 5000x5000 with 0 Axes>

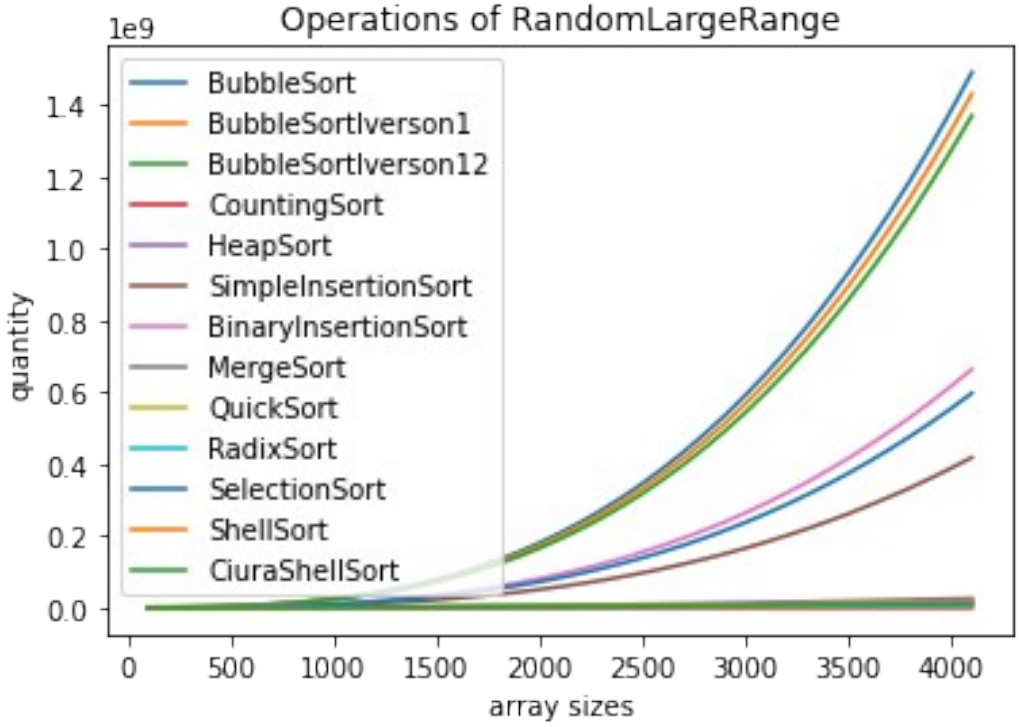



```
#@title Графики количества операций по всем сортировкам для конкретной генерации на больших массивах
# small size range, for all generations, operations
for curr in gens:
  plt.figure(figsize=(10, 10),dpi=500)
  fig, ax = plt.subplots()

  for sort in sorts:
      plt.plot(oper_big["Array sizes"], oper_big[curr + " & " + sort],
label=sort)
  ax.set(xlabel='array sizes', ylabel='quantity',
        title='Operations of ' + curr)

  fig.savefig(curr + "_time.png")
  plt.legend()
  plt.show()
```

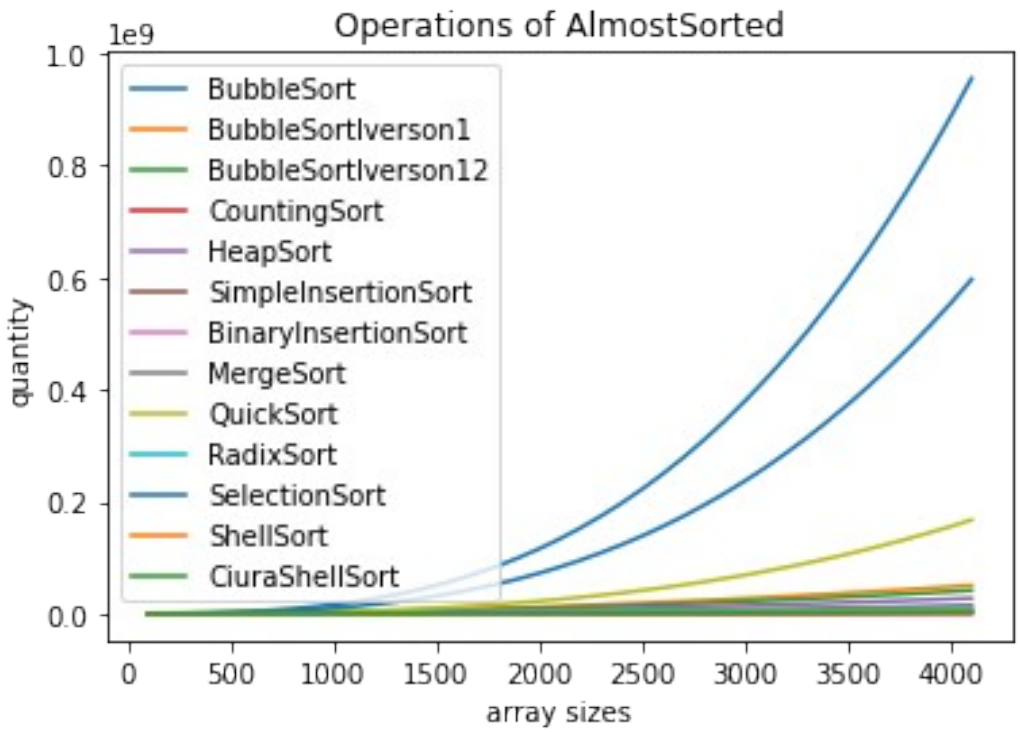
```
<Figure size 5000x5000 with 0 Axes>
```



<Figure size 5000x5000 with 0 Axes>



<Figure size 5000x5000 with 0 Axes>



<Figure size 5000x5000 with 0 Axes>