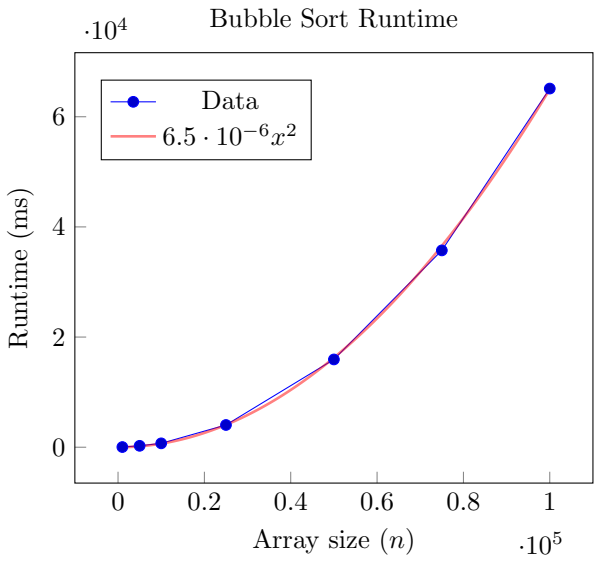
Bubble Sort

The expected runtime for bubble sort is $\mathcal{O}(n^2)$. The plot in blue is the runtime for varying array sizes of bubble sort and the red curve is a quadratic function for reference. It shows that the measured runtime very closely matches the expected runtime and becomes closer as n gets larger.

Array Size (n)	Runtime (ms)	Runtime/n^2
1000	31	$3.1 \cdot 10^{-5}$
5000	255	$1.0 \cdot 10^{-5}$
10000	702	$7.0 \cdot 10^{-6}$
25000	4036	$6.5 \cdot 10^{-6}$
50000	15936	$6.4 \cdot 10^{-6}$
75000	35735	$6.4 \cdot 10^{-6}$
100000	65120	$6.5 \cdot 10^{-6}$

Table 1: Bubble Sort Runtime

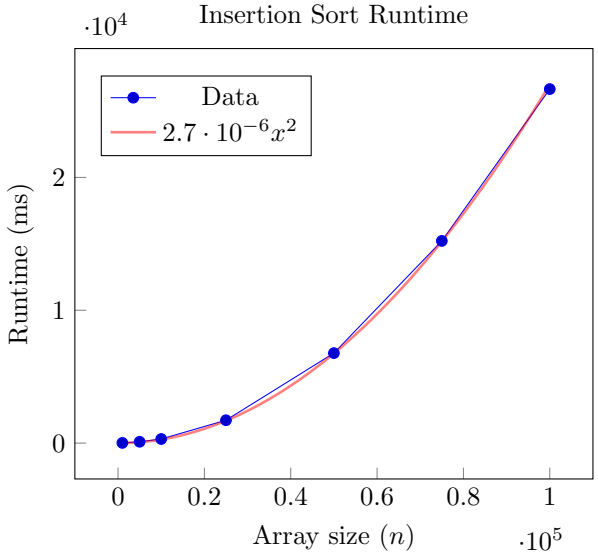


Insertion Sort

Like bubble sort, insertion sort has expected runtime of $\mathcal{O}(n^2)$. Once again, the data very closely matches the expected quadratic runtime, albeit somewhat faster than bubble sort.

Array Size (n)	Runtime (ms)	Runtime/n^2
1000	15	$1.5 \cdot 10^{-5}$
5000	99	$4.0 \cdot 10^{-6}$
10000	307	$3.0 \cdot 10^{-6}$
25000	1721	$2.8 \cdot 10^{-6}$
50000	6775	$2.7 \cdot 10^{-6}$
75000	15220	$2.7 \cdot 10^{-6}$
100000	26651	$2.7 \cdot 10^{-6}$

Table 2: Insertion Sort Runtime

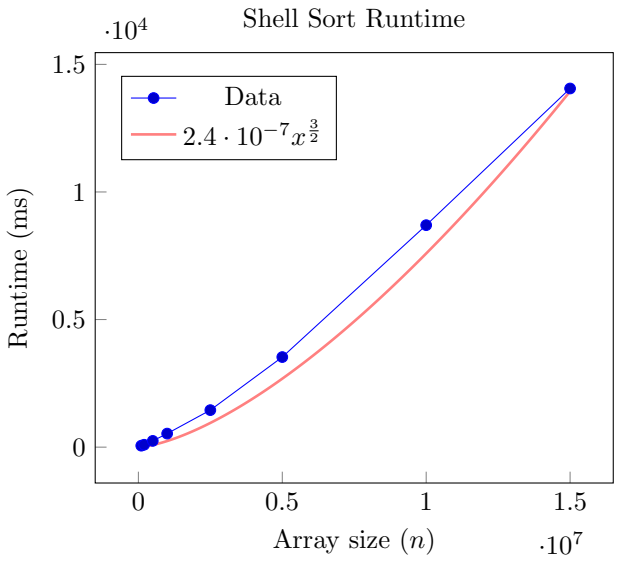


Shell Sort

Shell sort's expected runtime is $\mathcal{O}(n^{\frac{3}{2}})$. The plot shows that the measured runtime is close to the expected, yet not as close as with insertion sort or bubble sort. Overall, it ran significantly faster and could handle larger arrays.

Array Size (n)	Runtime (ms)	Runtime/$n^{\frac{3}{2}}$
100000	57	$1.8 \cdot 10^{-6}$
200000	94	$1.1 \cdot 10^{-6}$
500000	245	$6.9 \cdot 10^{-7}$
1000000	531	$5.3 \cdot 10^{-7}$
2500000	1451	$3.7 \cdot 10^{-7}$
5000000	3530	$3.2 \cdot 10^{-7}$
10000000	8699	$2.8 \cdot 10^{-7}$
15000000	14055	$2.4 \cdot 10^{-7}$

Table 3: Shell Sort Runtime

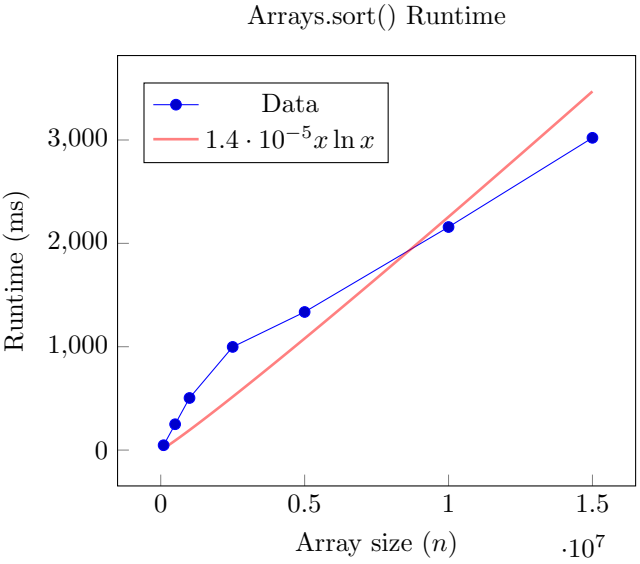


Arrays.sort()

Arrays.sort() showed dramatic improvement upon shell sort's runtime, suggesting its runtime is something smaller than $n^{\frac{3}{2}}$. I believe that this method has $n \log n$ runtime because the data was somewhat correlated to a plot of $n \log n$.

Array Size (n)	Runtime (ms)
100000	47
500000	250
1000000	504
2500000	998
5000000	1336
10000000	2158
15000000	3021

Table 4: Arrays.sort() Runtime



Quick Sort

Quick sort's expected runtime is $\mathcal{O}(n \log n)$, which in this case did closely match up with the data. Quick sort ran faster than either bubble, insertion, or shell sort, but slightly slower than Arrays.sort().

Array Size (n)	Runtime (ms)	Runtime/$n \log n$
100000	42	$8.4 \cdot 10^{-5}$
500000	400	$1.4 \cdot 10^{-4}$
1000000	464	$7.7 \cdot 10^{-5}$
2500000	764	$4.8 \cdot 10^{-5}$
5000000	1363	$4.1 \cdot 10^{-5}$
10000000	2462	$3.5 \cdot 10^{-5}$
15000000	3472	$3.2 \cdot 10^{-5}$

Table 5: Quick Sort Runtime