

Explanation of Task 2

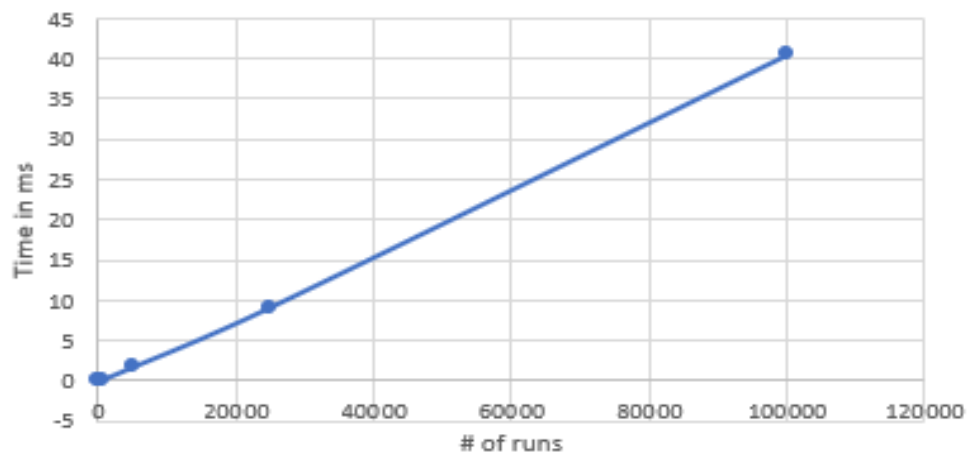
Screenshots: - times are in Nanoseconds

Time in Nanoseconds	10	100	500	5000	25000	100000
Bubble Sort:	650	38930	751850	83775660	1810408170	34421942950
Insert Sort:	2100	11040	165820	19297310	452549440	7133319750
Quick Sort:	2680	11520	68370	1199360	4322410	18274880
Merge Sort:	4600	57670	232790	3203370	20404480	79219800
Count Sort:	1170	6680	15130	146100	772900	3044810
Radix Sort:	3470	15110	74390	921480	5106380	22313770
Heap Sort:	880	16710	112360	1708950	9167840	40580920

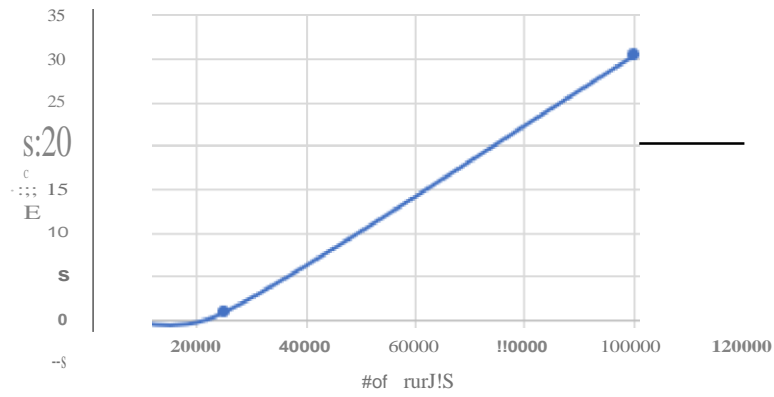
Table in MS:

# of runs	10	100	500	5000	25000	100000
Bubble	0.00065	0.03893	0.75185	83.77566	1810.40817	34421.94295
Insert	0.0021	0.01104	0.16582	19.29731	452.54944	7133.31975
Quick	0.00268	0.1152	0.06837	1.19936	4.32241	18.27488
Merge	0.0046	0.05767	0.23279	3.20337	20.40448	79.2198
Count	0.00117	0.00668	0.01513	0.1461	0.7729	30.4481
Radix	0.00347	0.01511	0.07439	0.92148	5.10638	22.31377
Heap	0.00088	0.01671	0.11236	1.70895	9.16784	40.58092
Time in ms						

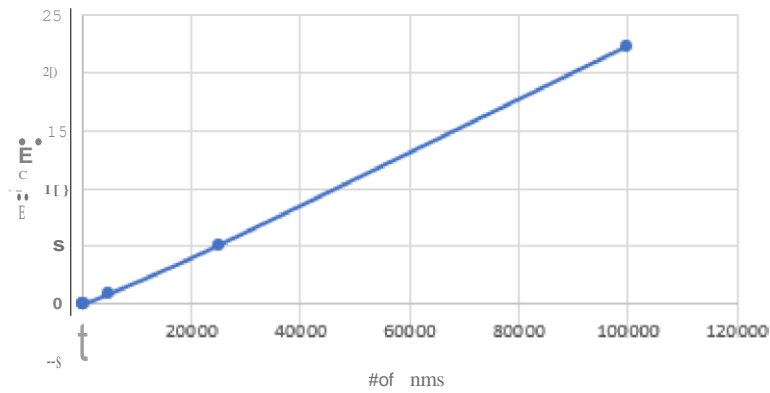
of runs vs Heap Sort



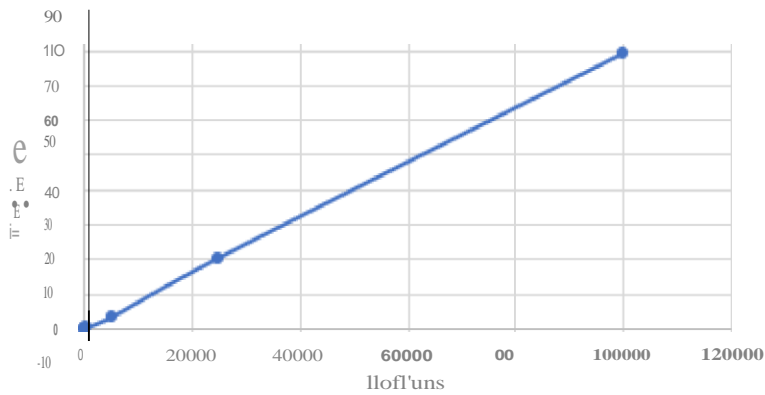
of runs vs Count Sort

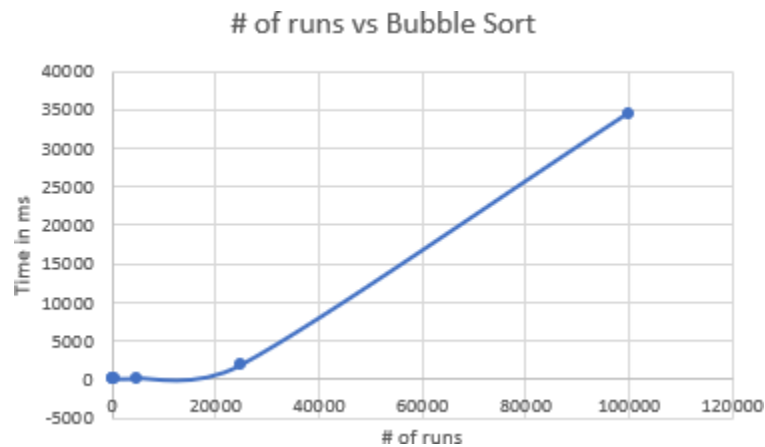
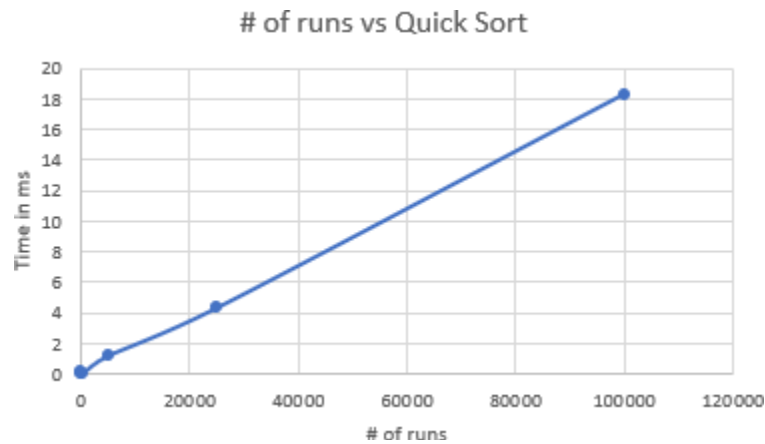


of runs vs Radix Sort



of runs vs Merge Sort





1) Explain the results based on their big N notation

a) Bubble sort: Average/Worst Case Performance - $O(N^2)$

- i) From the graph above we can see that as we added more elements to the sorting algorithm, the larger the graph looks like a parabolic curve representing N^2 . The results look a little higher than expected as the algorithm added more elements the slower the time became.

- b) Insertion Sort: Average case $O(n^2)$.
 - i) As we increase the amount in the sort the time is about 2-4 times faster than the bubble sort but it still has around the same shape. Performed as expected.
 - c) Quick Sort: Performance - $O(n \log n)$
 - i) For this it was actually less than expected as the graph is basically a flat line as perform faster than bubble sort and insertion sort.
 - d) Merge sort: Performance - $O(n \log n)$
 - i) The merge sort performed as expected and while less than what we discussed in class I believe that a computer's hardware does affect the final outcome.
 - e) Heap sort: Performance - $O(n \log n)$
 - i) As the number of tests becomes greater the line starts to increase almost exponentially but still follows the predicted path. Performed as expected
 - f) Counting Sort: Performance - $O(2n+k)$
 - i) As the value we input into counting sort is random we cannot predict the outcome however as there may be duplicates so the number could be smaller than the expected.
 - g) Radix-Sort: performance $O(d*n)$
 - i) The result is very linear and thus meets the required parameters for the class. Meets the expected performance for the class.
- 2) What did we expect for time for each of the array sizes based on the results for array size of 10?

Time in ms	10	100 (estimate)	500 (estimate)	5000 (estimate)	25000 (estimate)	100000 (estimate)
Bubble	0.00065	0.01	0.025	25	625	10000
Insertion	0.0021	0.01	0.025	25	625	10000
Merge	0.0046	0.0002	0.0013	0.01849	0.1099	0.5
Quick	0.00268	0.0002	0.0013	0.01849	0.1099	0.5
Counting	0.00117	0.0004	0.002	0.02	0.1	0.4
Radix	0.00347	0.01	0.25	25	625	10000
Heap	0.00088	0.0002	0.0013	0.01849	0.1099	0.5

Calculations above were done with respective time complexities.