**Lab 5 Report: Sorting and Efficiency**

Lab Group: 8

Name(s): Keerthi Sekar and Grayson Cox

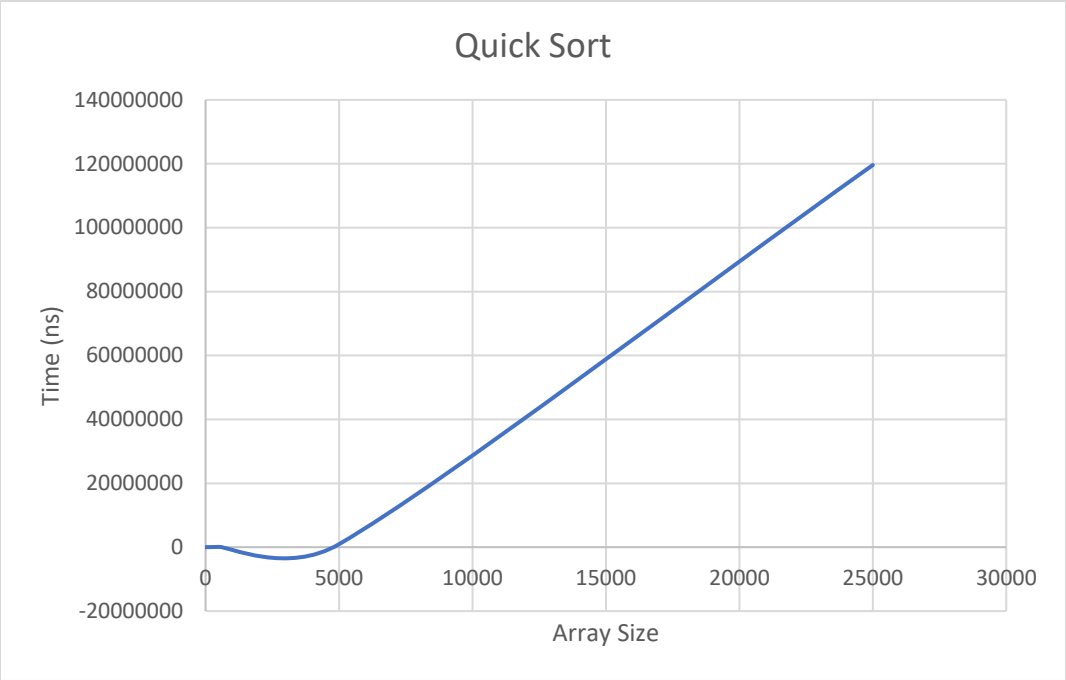
The following table explains the times that were run on both Keerthi and Grayson's laptops.

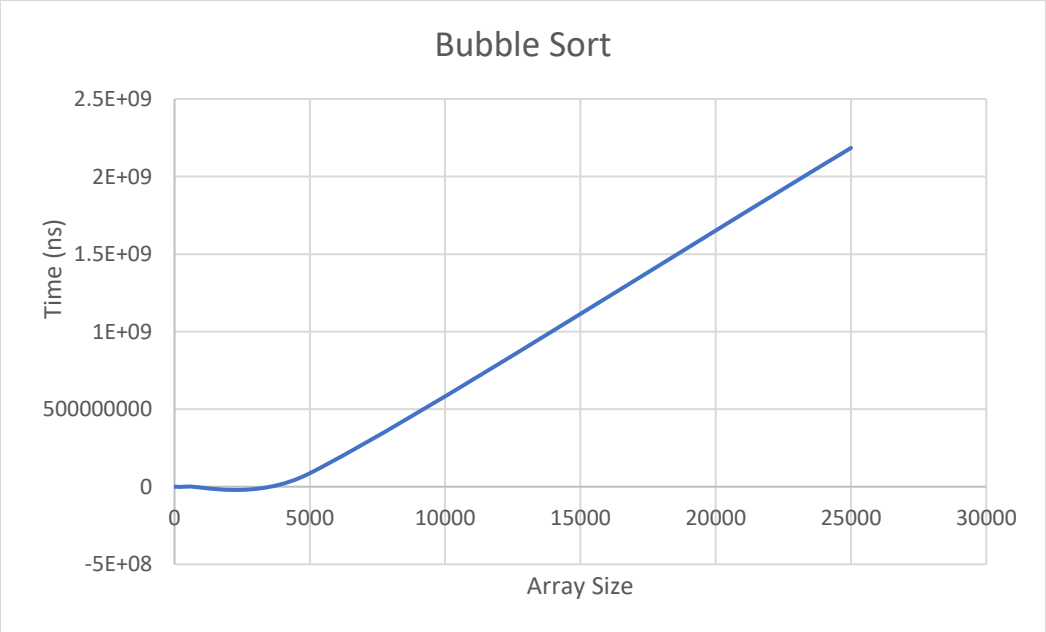
| Sorting Method | **10 (ns)** | **100 (ns)** | **500 (ns)** | **5000 (ns)** | **25000 (ns)** |
|---|---|---|---|---|---|
| Bubble Sort | 1000 | 45700 | 974800 | 86638500 | 2184535900 |
| Insertion Sort | 300 | 23700 | 270300 | 20498900 | 434034800 |
| Merge Sort | 1200 | 10600 | 77200 | 763100 | 3940900 |
| Quick Sort | 1200 | 29100 | 91200 | 933800 | 119614400 |
| Radix Sort | 900 | 28200 | 34700 | 321400 | 1469700 |

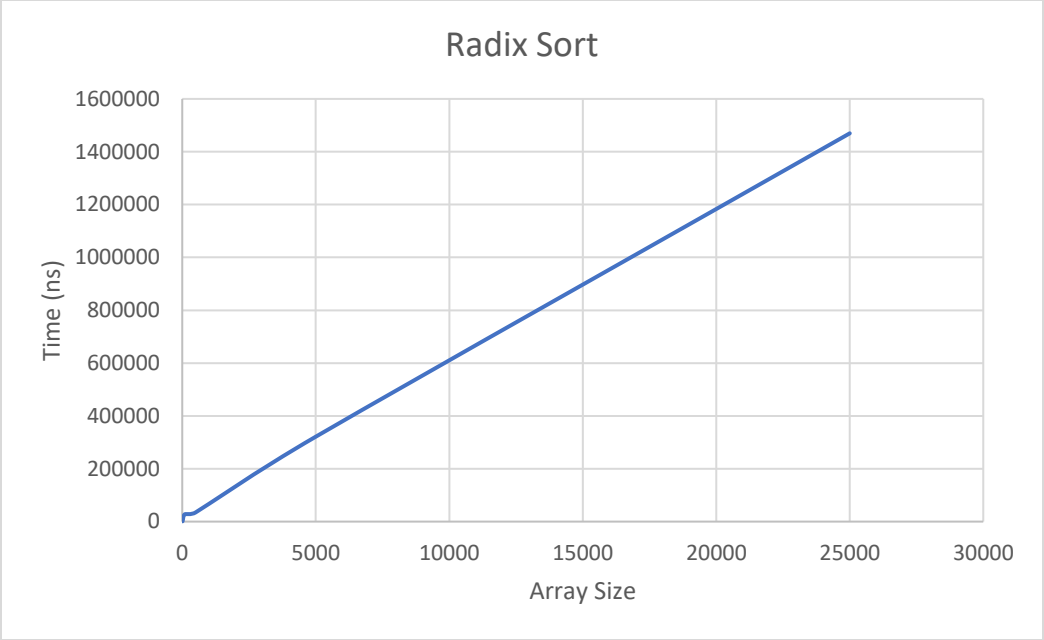
We expected the growth order to be how it is in the table, bubble being the worst case for the larger datasets to Radix being the quickest. Since bubble has a $n^2$ growth rate it will take the longest like seen in the graph. Radix's growth rate is nk (k = maximum possible value). So, it obviously will be a lot quicker that bubble and the other remaining sorting methods. The values increasing in the magnitudes that it did in nanoseconds, seen in the table, was under our expectations as we assumed the growth of time would match the size of the datasets. (25,000 is a huge array so the time being in the $10^9$ is right.)

We did not run into any issues of memory or functions running longer than 4 seconds at the most. We think that the main reason this is not an issue for us is that we both have a powerful processor (i7) capable of computing these sorting algorithms at an efficient rate. From the table it looks like the longest running one is bubble and the shortest method is radix. Radix and Merge both have a relatively linear runtime in comparison to the others.

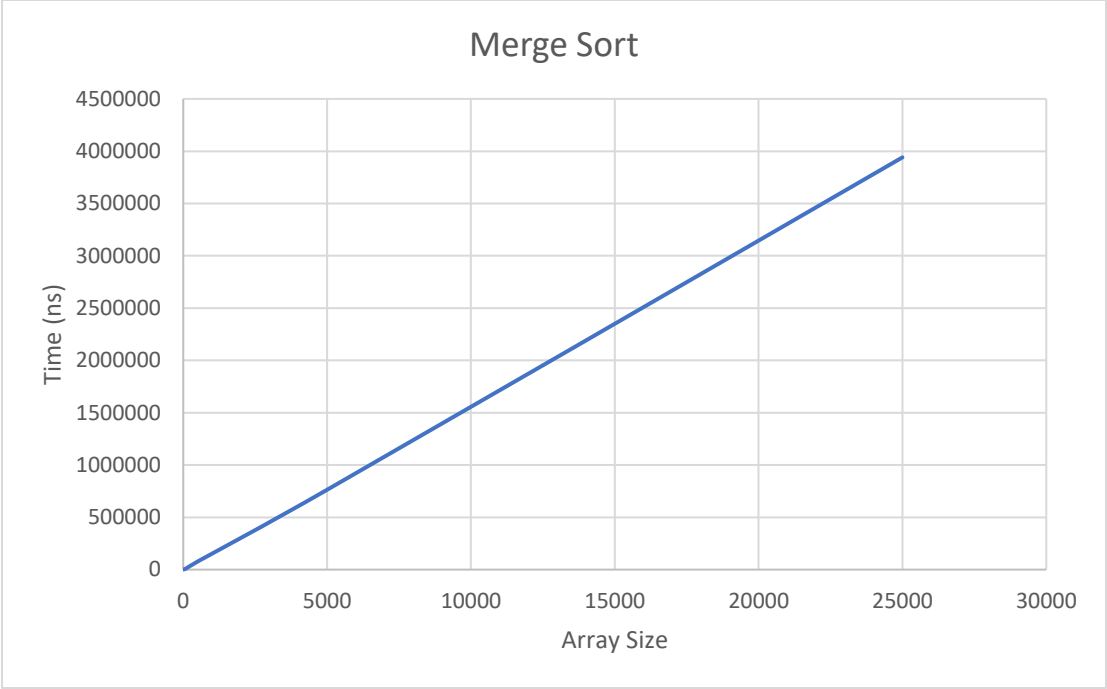
In graphical form for each sorting algorithm sorting analysis:




Radix Sort
1600000
1400000
1200000
1000000
800000
600000
400000
200000
0
Time (ns)
0
5000
10000
15000
20000
25000
30000
Array Size


Insertion Sort
500000000
450000000
400000000
350000000
300000000
250000000
200000000
150000000
100000000
50000000
0
-50000000
Time (ns)
0
5000
10000
15000
20000
25000
30000
Array Size


Merge Sort
4500000
4000000
3500000
3000000
2500000
2000000
1500000
1000000
500000
0
Time (ns)
0
5000
10000
15000
20000
25000
30000
Array Size