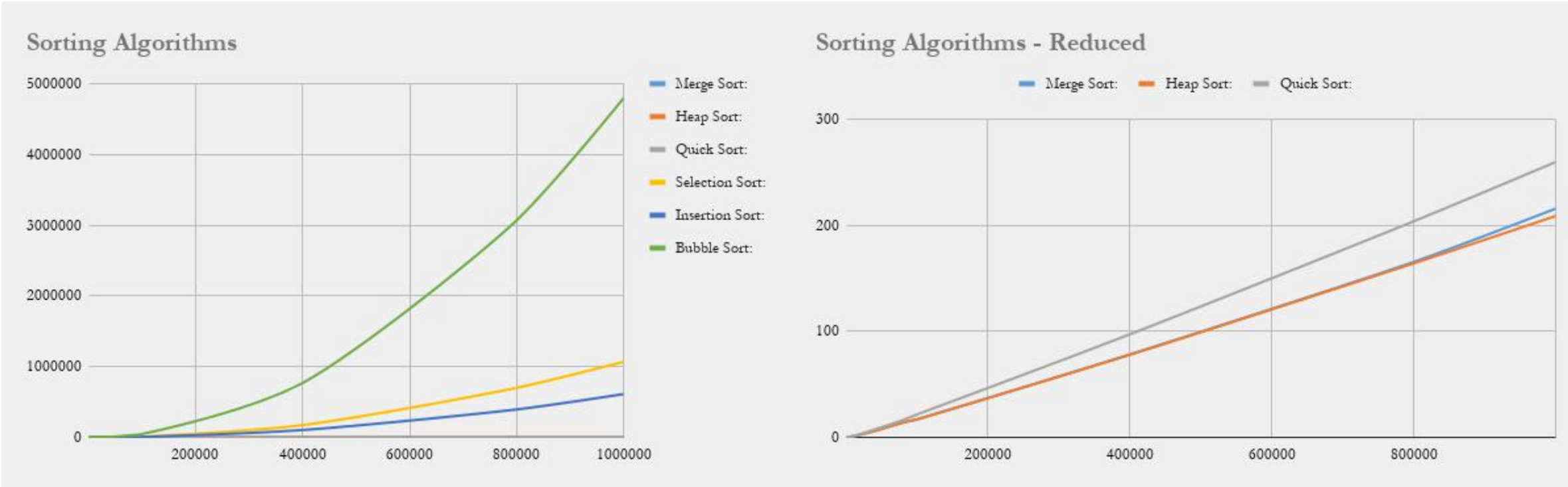

The graph has been shown in two different perspectives to fully represent the results. Since [Bubble, Insertion, Selection] sorts take a much longer time when it comes to larger datasets than the other [Merge, Heap, Quick] sorts, the latter ones get burried.

Notes and Observations

The program's stack frame vs physical memory

When handling large datasets/arrays, there are two restrictions. The stack frame and physical memory(heap). Since we are dealing with large datasets, it's safer to use arrays on the heap to avoid a stack overflow.
In the case of Merge Sort, we create temporary arrays to “Divide and Conquer”, each containing half the size of the array to sort (± 1). These arrays should be on the heap as well since the stack frame may not hold half of the size of the original array.

Testing Environment

Although the initial testing was done in one sitting, it is not safe to guarantee that other factors may have affected the results. Background processes and/or other processes may have had different load balances affecting the final runtime results.

Theory Testing

Referring to the table below, we can see that some algorithms share the same **best-case time complexity**, but the implementation may change the results.

Merge Sort vs Heap Sort

As to note, Merge Sort **should** be faster than Heap Sort for larger datasets, but in our results this did not reflect at the time of initial testing. This may be due to the reasons pointed out in the **Testing Environment** section, since re-testing in similar conditions later on resulted in the opposite and expected outcome.

Insertion Sort vs Selection Sort

At the time of sorting the larger datasets is when we can see the clear difference between these two algorithms. Selection sort's worst case is when the array is not partially sorted, since it will have to read the entire unsorted portion of the array. While Insertion sort will read the sorted part of the array until it finds the correct place, making it more stable and efficient for larger datasets.

Bubble Sort

Bubble sort showed to be the slowest algorithm by a great margin - expectedly .- It will read and swap two elements every step until the array is sorted. This sort is enough for smaller datasets but when sorting large datasets it just becomes unbearably slow.

Time complexity

	Best Case
Selection Sort	Ω(n^2)
Bubble Sort	Ω(n)
Insertion Sort	Ω(n)
Heap Sort	Ω(n log(n))
Quick Sort	Ω(n log(n))
Merge Sort	Ω(n log(n))