CCDSALG Term 3, AY 2019 – 2020

Project 1 – Comparing Sorting Algorithms

Section	Names	Task 1	Task 2	Task 3	Task 4
S13	Dela Cruz, Sabrina Mykel	X		X	X
	Mandadero, Clarissa Mae	X	X		X
	Sanson, Ralph Matthew	X	X	X	X

LIST OF SORTING ALGORITHMS

Sorting Algorithm	Author (if available)	Downloaded From
1. Bubble sort	n/a	https://www.programmingsimplified.com/c/source-code/c-program-bubble-sort
2. Insertion sort	n/a	
3. Selection sort	n/a	https://www.geeksforgeeks.org/selection-sort/
4. Merge sort	n/a	
5. Heap sort	n/a	https://www.geeksforgeeks.org/heap-sort/
6. Quicksort	n/a	https://www.programming9.com/programs/c-programs/234-c-program-for-shell-sort

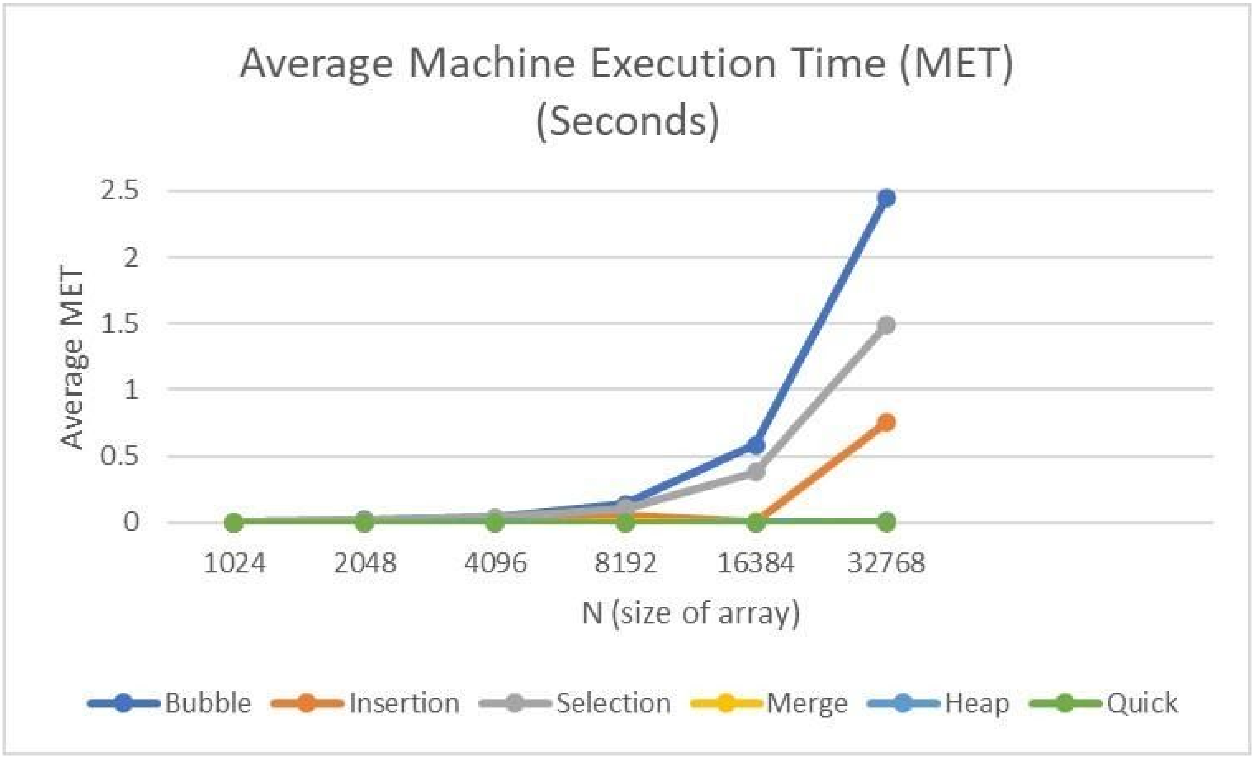
COMPARISON TABLE

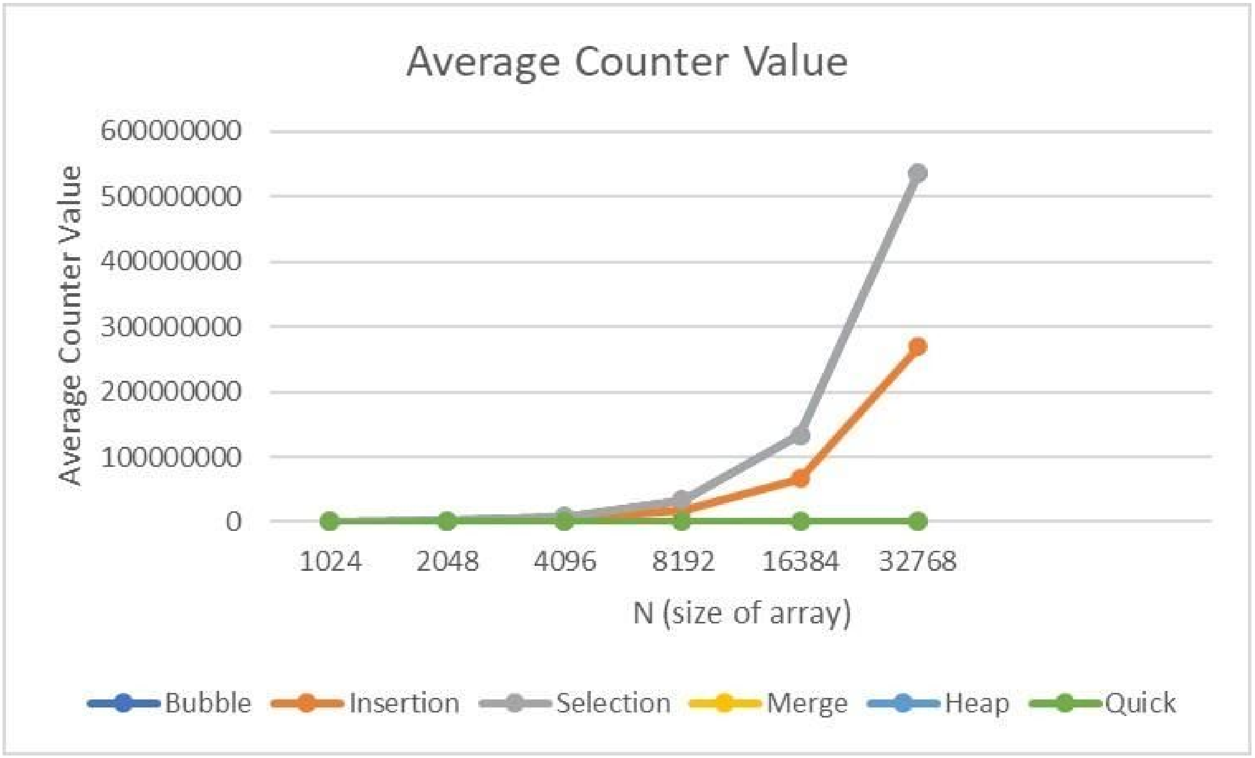
M = (10)

Size	Average Machine Execution Time (seconds)					
	Bubble *$O(n^2)$*	**Insertion** *$O(n^2)$*	**Selection** *$O(n^2)$*	**Merge** *$O(nlogn)$*	**Heap** *$O(nlogn)$*	**Quick** *$O(n^2)$*
1024	0.003125	0.001563	0.003125	0	0	0
2048	0.017188	0.004687	0.010937	0	0	0
4096	0.037500	0.025000	0.040625	0	0.001563	0
8192	0.139063	0.054688	0.104688	0.001563	0	0
16384	0.585938	0.185938	0.378125	0.004687	0.003125	0
32768	2.445313	0.753125	1.487500	0.007813	0.004687	0.003125

Size	Average Counter Value					
	Bubble *$O(n^2)$*	**Insertion** *$O(n^2)$*	**Selection** *$O(n^2)$*	**Merge** *$O(nlogn)$*	**Heap** *$O(nlogn)$*	**Quick** *$O(n^2)$*
1024	523776	258903	523776	8961	18122	10925
2048	2096128	1060519	2096128	19954	40378	25886
4096	8386560	4223427	8386560	43998	88906	56774
8192	33550336	16797271	33550336	96179	194316	123408
16384	134209536	66830684	134209536	208656	421504	280585
32768	536854528	268924019	536854528	450160	908218	606249

GRAPHS


Average Machine Execution Time (MET) (Seconds)
Average MET
2.5
2
1.5
1
0.5
0
1024
2048
4096
8192
16384
32768
N (size of array)
Bubble
Insertion
Selection
Merge
Heap
Quick


Average Counter Value
Average Counter Value
600000000
500000000
400000000
300000000
200000000
100000000
0
1024
2048
4096
8192
16384
32768
N (size of array)
Bubble
Insertion
Selection
Merge
Heap
Quick

DISCUSSION

The average counter value per sorting algorithm shows the number of comparisons each sorting algorithm undergoes to sort a given array, so the higher the average counter value, the more tasks one sorting algorithm must perform. Looking at the average counter value, we have observed 3 things. First, Selection and Bubble Sort have the same average counter value in every value of N, and the average counter value in every run with the same N stays the same even if the data being sorted is different. This means that, for Selection and Bubble Sort, the values being sorted does not matter, it will still compare all values to one another in every run, resulting to a large number of comparisons. Second, both Selection and Bubble Sort are not recommended for sorting an array with a large size. As the array size (N) becomes larger, the average counter value in the stated sorting algorithms become much larger compared to the other sorting algorithms that we have tested, making it inefficient for sorting arrays with a large size. Third, we observed that both Merge and Quick Sort are ideal for sorting an array with a large size because its average counter value when sorting a large N is far lesser than the other sorting algorithms that we have tested.

Machine Execution Time (MET) refers to the time it takes for a machine to execute a certain program which means that the lower the MET, the faster the machine executes the program. One of the few things that we have noticed is that Bubble Sort's Machine Execution Time rises exponentially with each rising value of N, but only becomes noticeable once it breaks that 1 second barrier on sample sizes around >= 32678 (more or less). As is also seen with Insertion and Selection Sort since the three share the same Big Oh, but Insertion and Selection Sort have significantly lower MET compared to Bubble Sort because of how the code of each sorting algorithm was written. Looking at the last 3 algorithms on the other hand, seem to have very efficient algorithms to the point that none of the MET for the last 3 have gone over 0.01 second.

To summarize, after taking into consideration the average counter value and MET of each sorting algorithm. We found out that the most efficient sorting algorithms, out of the algorithms that we have tested, are Quick and Merge Sort. In the average counter value, both sorting algorithms have the least amount of average counter value, and as the N becomes larger, its average counter value does not increase as exponentially as the other sorting algorithms. For the average Machine Execution Time (MET), both algorithms are also the ones with the least amount of MET, with it being very close to 0 seconds. We have also found out that sorting algorithms that use recursions perform much faster than other sorting algorithms which do not use recursions. The sorting algorithms that use recursions are Heap, Merge, and Quick and, as seen from the graphs, they are the three most efficient sorting algorithms, outperforming the other algorithms - in terms of average counter value and MET - which only used simple loop statements in sorting.