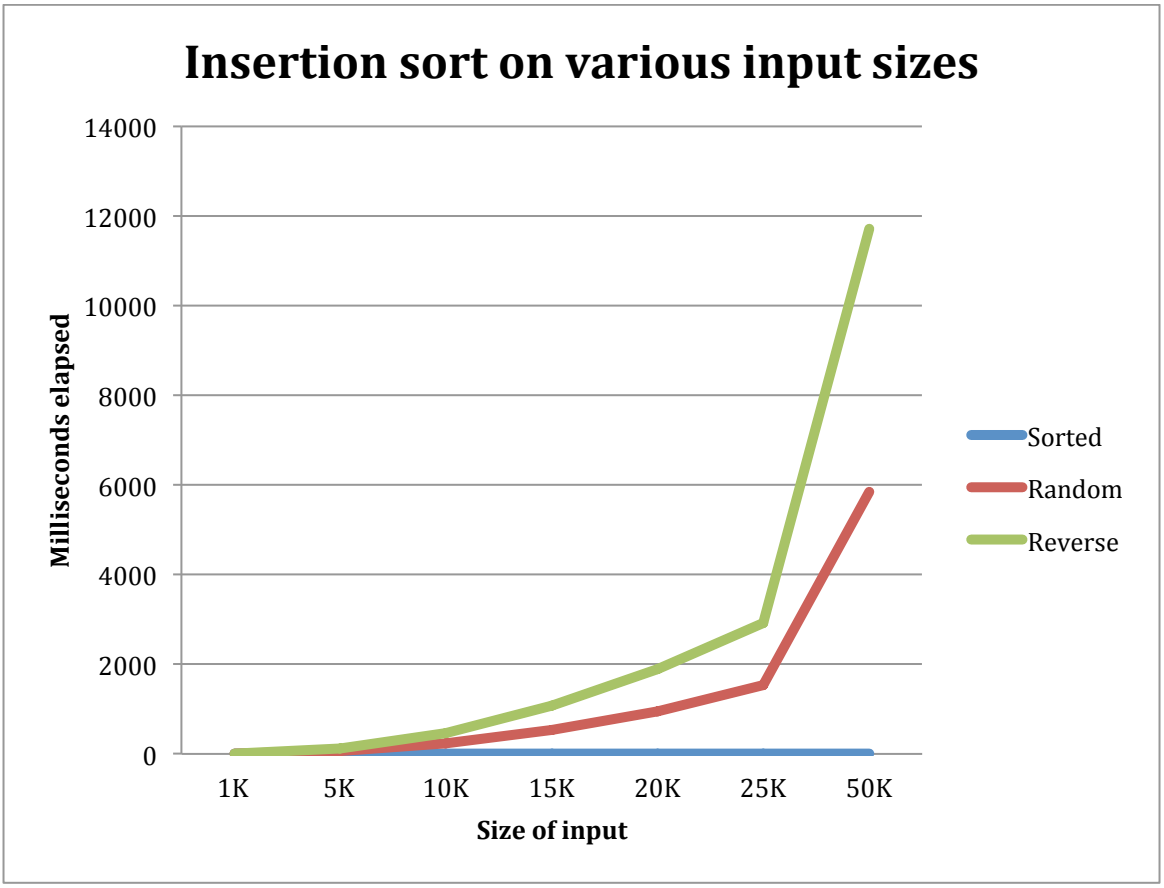
Insertion Sort Analysis

	1K	5K	10K	15K	20K	25K	50K
Sorted	0	0	0	0	0	0	1
Random	2	60	230	529	949	1539	5853
Reverse	5	118	467	1076	1895	2924	11715

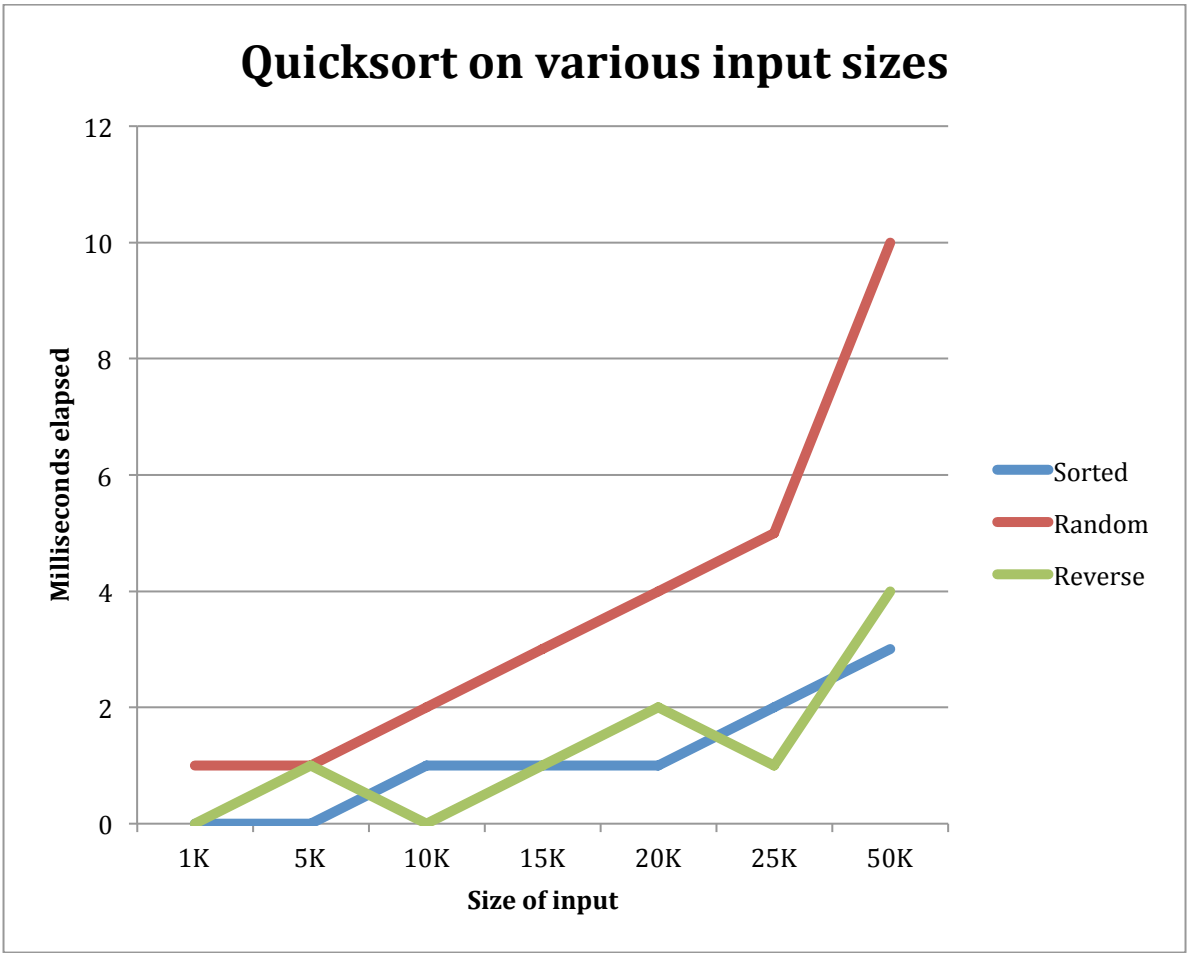


Clearly from this chart, we see that sorted array trivially takes no time while an array of random inputs which would be the average case takes roughly half the time of reversed list and sorted list.

Reversed list being the worst case time, does indeed take the most time. We can really start seeing the difference in time when the input size jumps from 25K to 50K. Furthermore the curve is skewed left.

Quicksort Analysis

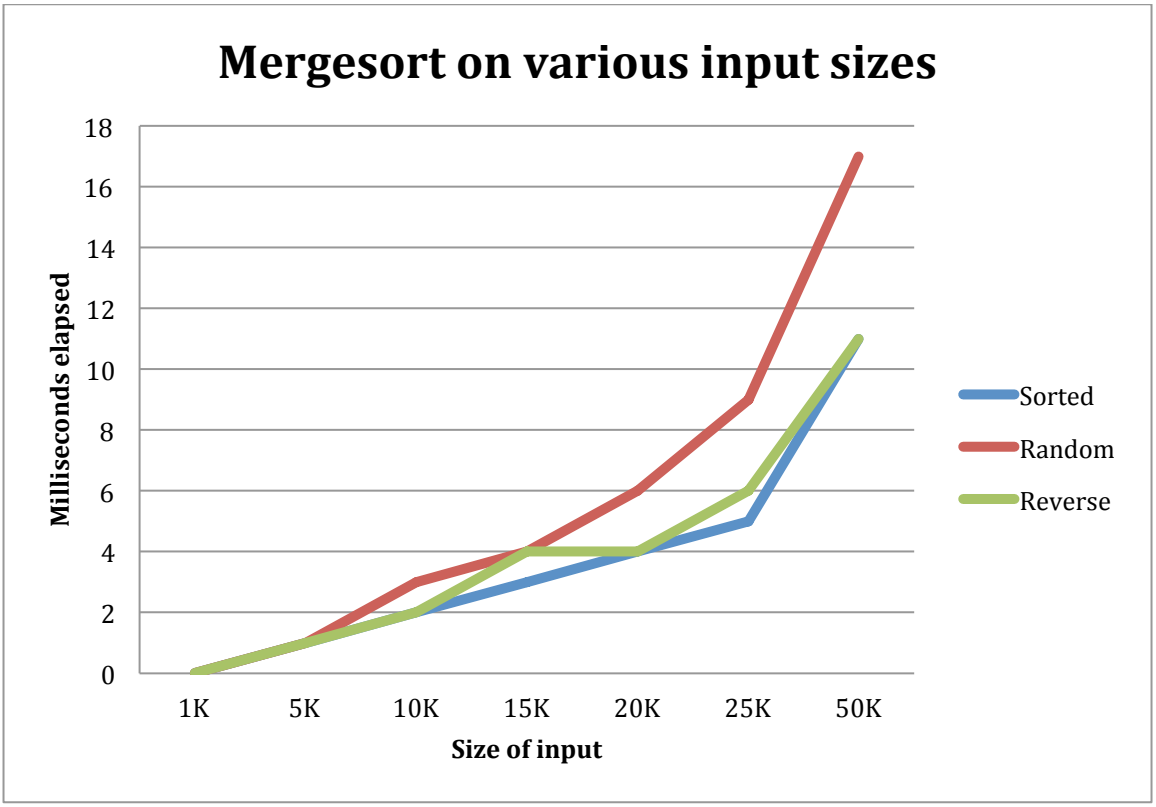
	1K	5K	10K	15K	20K	25K	50K
Sorted	0	0	1	1	1	2	3
Random	1	1	2	3	4	5	10
Reverse	0	1	0	1	2	1	4



With this graph, we can see that quicksort performs dramatically better than most other sorting algorithms. Notice however that quicksort doesn't perform the worst when the array is completely reversed; instead it takes the most time when the input is random.

Mergesort Analysis

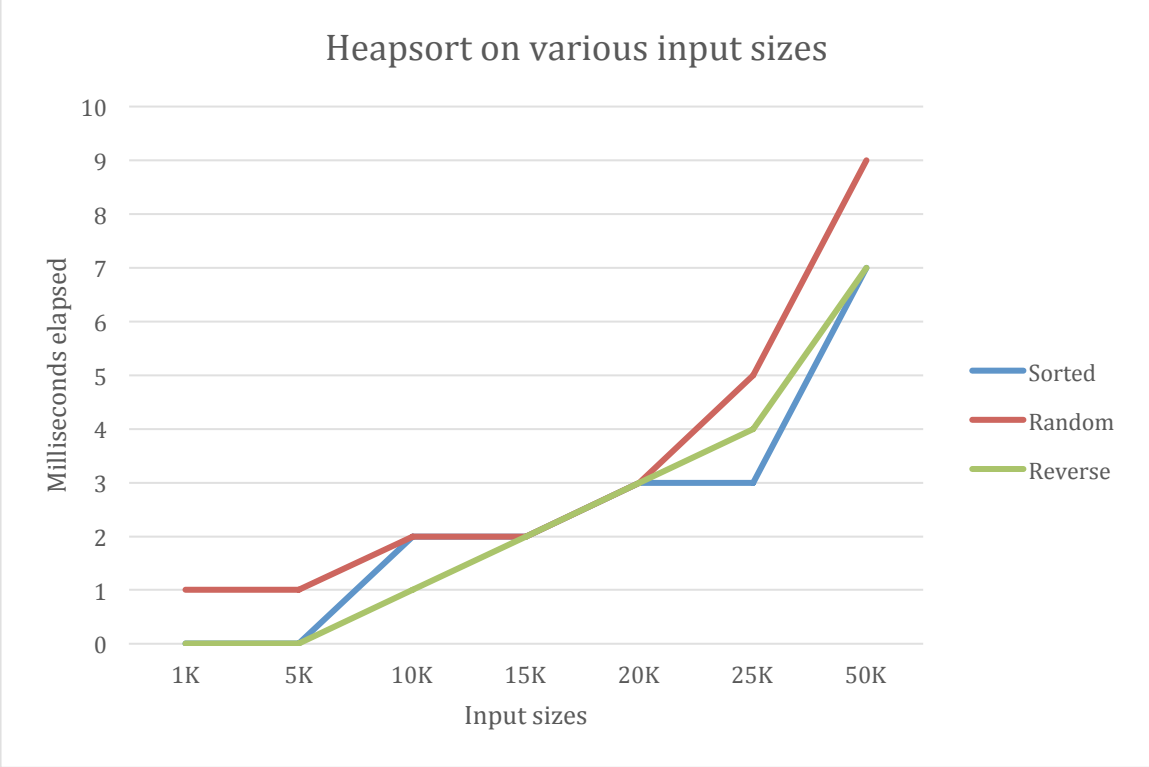
	1K	5K	10K	15K	20K	25K	50K
Sorted	0	1	2	3	4	5	11
Random	0	1	3	4	6	9	17
Reverse	0	1	2	4	4	6	11



Mergesort having the worse, best, and average time complexity of nlogn definitely seems appropriate according to this graph. We can see that both sorted and reverse input arrays performed roughly around the same time.

Heapsort Analysis

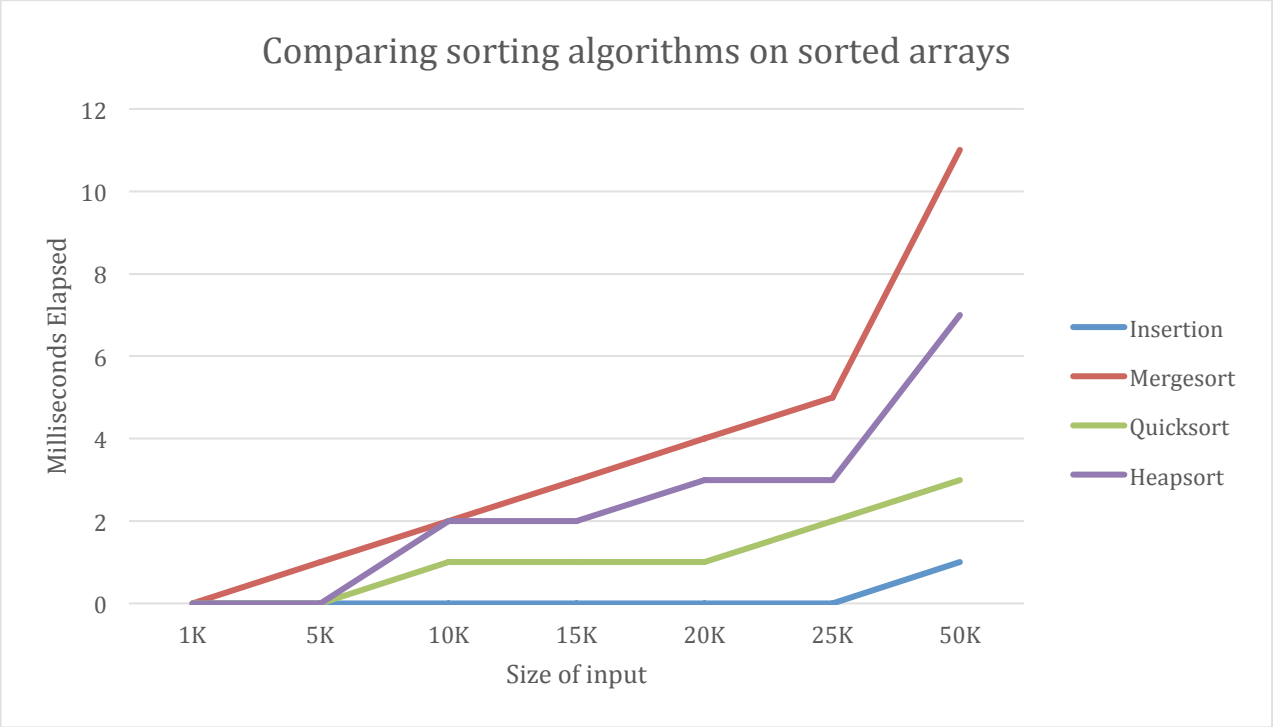
	1K	5K	10K	15K	20K	25K	50K
Sorted	0	0	2	2	3	3	7
Random	1	1	2	2	3	5	9
Reverse	0	0	1	2	3	4	7



The asymptotic running time of Heapsort is nlogn for best, worse, and average case. This graph clearly reflects this theoretical calculation. As we can see the sorted, random, reverse arrays all take roughly the same amount of time.

Sorted Arrays – Algorithms comparison

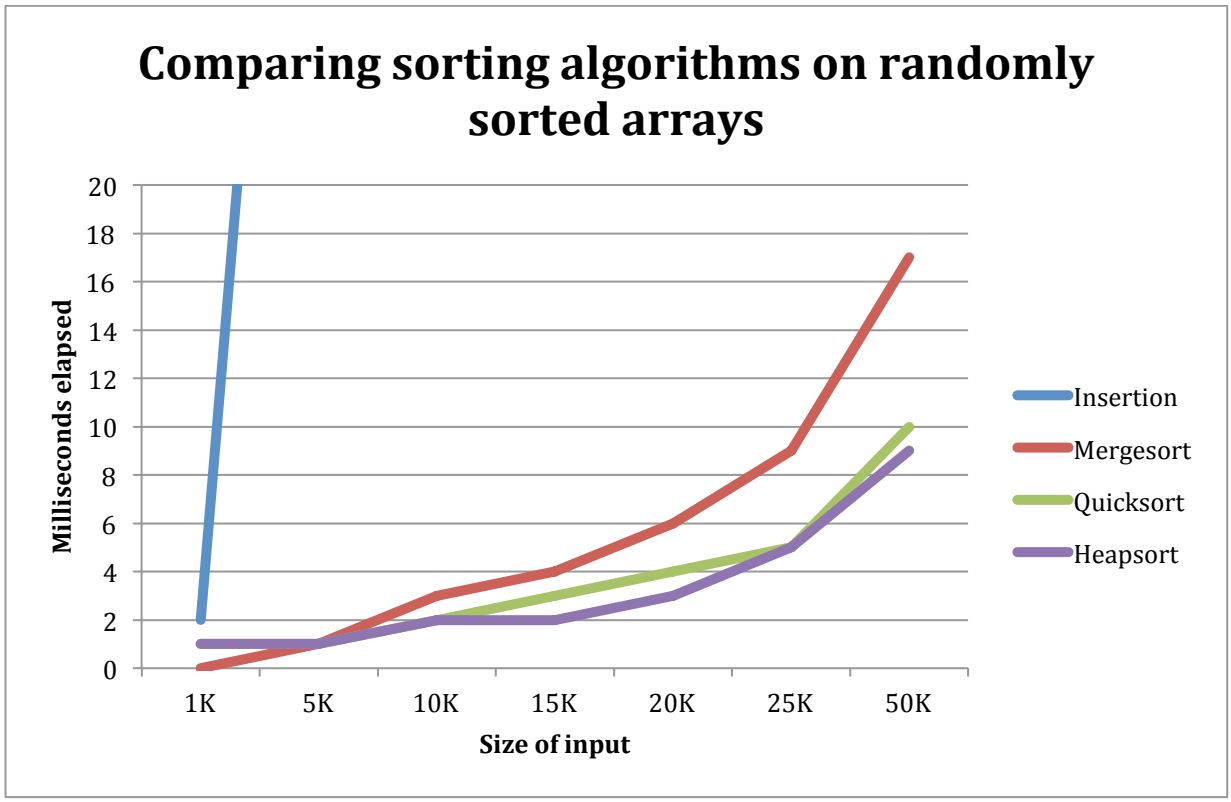
	1K	5K	10K	15K	20K	25K	50K
Insertion	0	0	0	0	0	0	1
Mergesort	0	1	2	3	4	5	11
Quicksort	0	0	1	1	1	2	3
Heapsort	0	0	2	2	3	3	7



Insertion sort seems to perform the best when the input is already sorted. Mergesort takes the most time out of the four algorithms to sort any number of inputs.

Randomly sorted arrays – Algorithms comparison

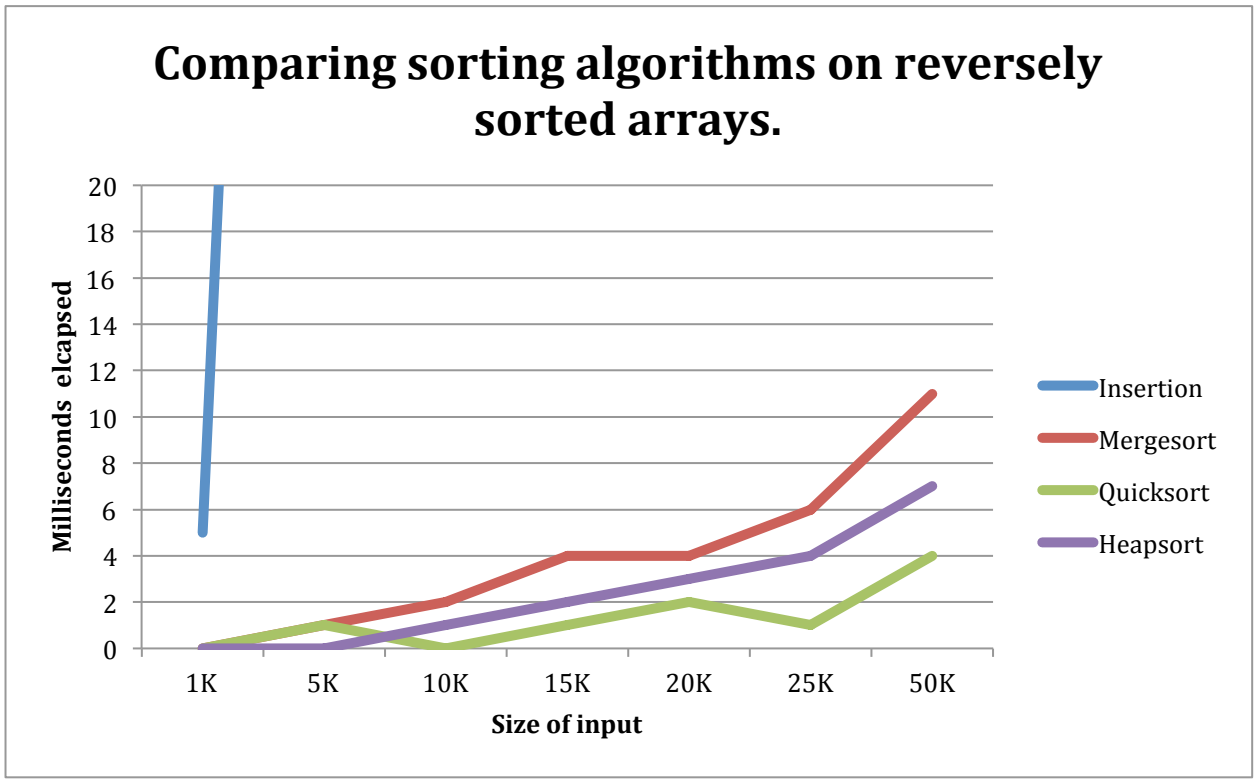
	1K	5K	10K	15K	20K	25K	50K
Insertion	2	60	230	529	949	1539	5853
Mergesort	0	1	3	4	6	9	17
Quicksort	1	1	2	3	4	5	10
Heapsort	1	1	2	2	3	5	9



From this graph, we can easily deduce that insertion takes the longest amount of time. Furthermore, heapsort and quicksort seem to be performing at relatively the same pace. Note I had to change the scale on the y-axis because of the dramatic difference between insertion sort and the other algorithms and to better view the graph.

Reversely sorted arrays – Algorithms comparison

	1K	5K	10K	15K	20K	25K	50K
Insertion	5	118	467	1076	1895	2924	11715
Mergesort	0	1	2	4	4	6	11
Quicksort	0	1	0	1	2	1	4
Heapsort	0	0	1	2	3	4	7



This graph is fairly similar to the graph for randomly sorted; most of those attributes apply to this graph analysis also. However, here we can notice that quicksort out performs all the other algorithms in comparison.