Machine Configuration:

Processor: Intel(R) Core(TM) i5-8250U CPU
1.60GHz up to 1.80 GHz

RAM: 4.00 GB

Operating System: Windows 10 64-bit

Complexity analysis:

1. Merge Sort:

Let us consider, the running time of Merge-Sort as ***T(n)***.

Hence,

T(n)= O(1) when n<=1

Otherwise,
T(n)= 2*T(n/2) + bn where b is a constant
Therefore, using this recurrence relation,

Base case, T(n) =b, case occurs when $2^i = n$

That is, i = log n

T(n) = bn +bnlogn

Thus, T(n) = O(nlogn)

For merge sort the ascending, descending and random order cases are the same. The time complexity doesn't change according to the order of the array.

Space Complexity: O(n)

2. **Quick Sort:**

The worst case complexity of Quick-Sort algorithm is **$O(n^2)$**. However using this technique, in average cases generally we get the output in **O(n log n)** time.

The real liability of quicksort is that it runs in $O(n^2)$ on already-sorted input.

Space complexity: O(n)

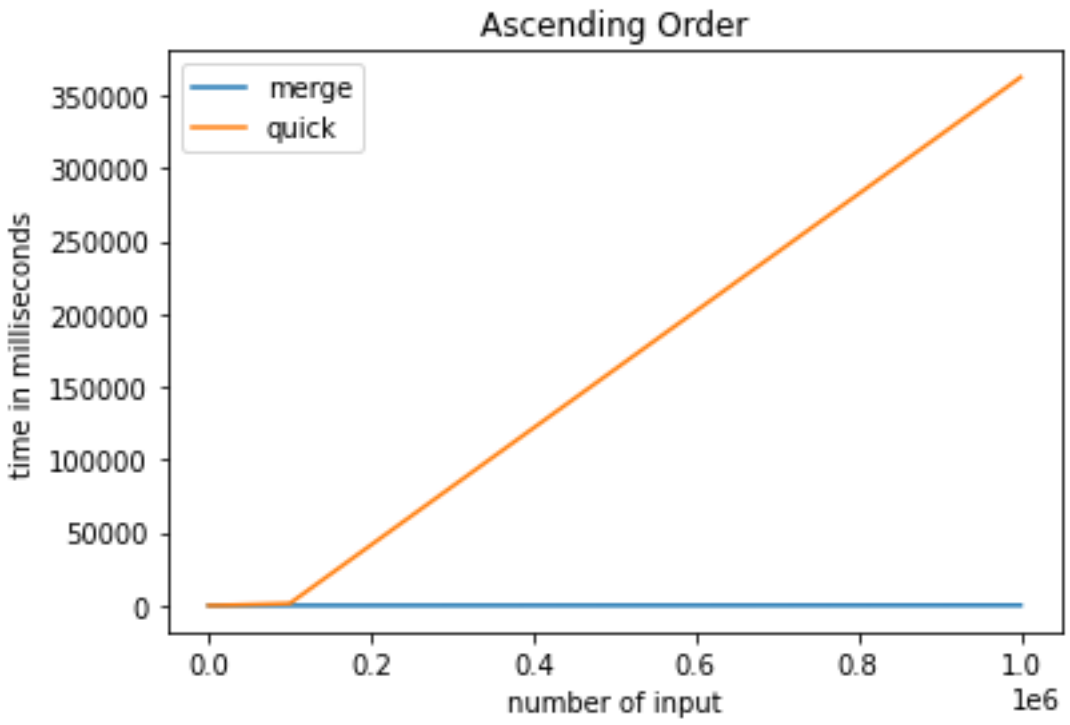
Data Table:

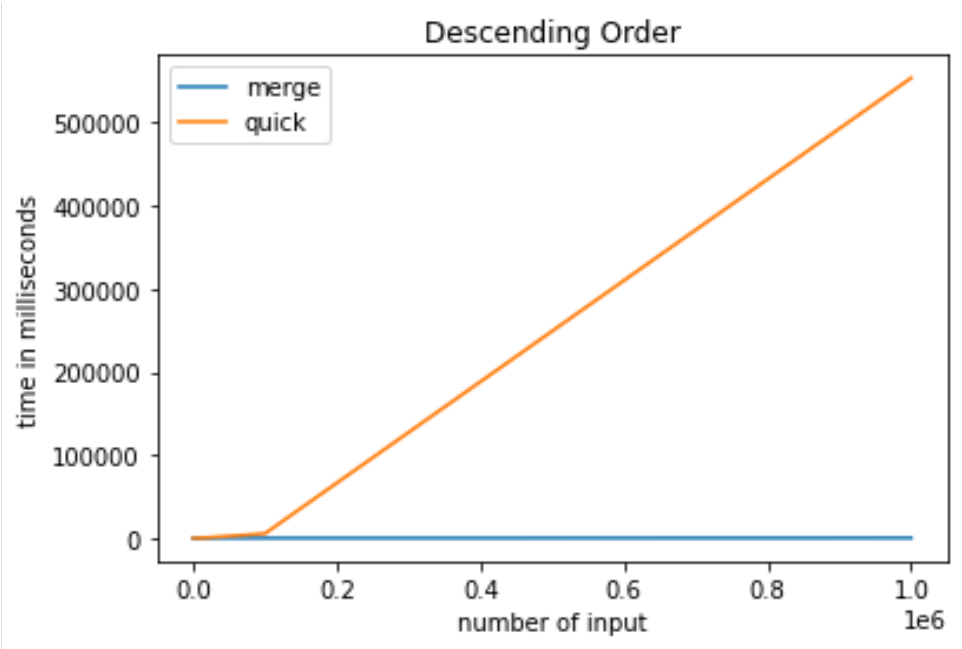
Time taken in Milliseconds

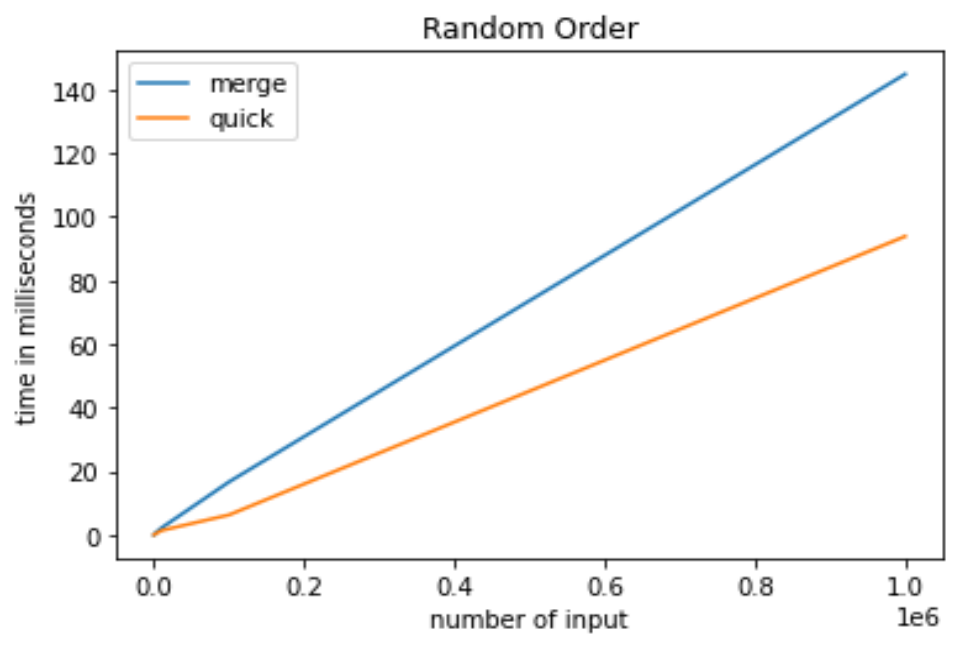
Input Order	N= Sorting Algorithm	10	100	1000	10000	100000	1000000
Ascending	Merge	0	0.1	0.3	1.099	7.6	137.5
	Quick	0	0.2	3	35.1	1382.8	362380.2
Descending	Merge	0	0.1	0.3	0.7	7.5	167.3
	Quick	0	0.3	2.7	50.01	5365.2	553921.2
Random	Merge	0	0.2	0 .4	2.2	16.7	144..9
	Quick	0	0	0.2	1.4	6.3	93.9

Graph Plots:

The graphs for comparison between Merge sort and Quick sort is attached herewith.


Ascending Order
merge
quick
350000
300000
250000
200000
150000
100000
50000
0
time in milliseconds
0.0
0.2
0.4
0.6
0.8
1.0
1e6
number of input


Descending Order
merge
quick
500000
400000
300000
200000
100000
0
time in milliseconds
0.0
0.2
0.4
0.6
0.8
1.0
1e6
number of input



Submitted by: Ayesha Binte Mostofa

Student ID: 1805062

Section: B-1