CSE- 204

Data Structures and Algorithms (Sessional)

Offline 7: Sorting Algorithms

Date of Submission: 11.06.2021

Submitted by:

Sanju Basak

Student ID: 1805064

Dept. CSE, BUET

Section: B1 Batch: 18

Table:

Table: Average time(in milliseconds) for sorting n integers in different input orders

Input Order	n =	10	100	1000	10000	100000	1000000
	Sorting Algorithm						
Ascending	Merge	0	0	0	1.6	13.6	116
	Quick	0	0	1.5	162.3	16188	1640009
Descending	Merge	0	0	0	1.2	13.3	107.667
	Quick	0	0	1.5	134.4	13749	1393270
Random	Merge	0	0	0	1.6	15.7	184
	Quick	0	0	0	1.5	9.4	168

Complexity Analysis:

For Merge Sort:

Theoretically, for merge sort, time complexity is O(nlogn). That means, ideally, an increase of ‘n’ by 10 times, would increase the runtime by a little more than 10 times, for all orders.

Now, we will analyze it with our data table.

1. Comparing runtime with change in input size, (n= $10^6 : 10^5 : 10^4$) :
    a. ***Ascending order-*** 116 : 13.6 : 1.6 ≈ 100:10:1 (A bit more than 10x in each step, to be precise)
    b. ***Descending order-*** 107.667 : 13.3 : 1.2 ≈ 100:10:1 (A bit more than 10x in each step, to be precise)
    c. ***Random order-*** 184 : 15.7 : 1.6 ≈ 100:10:1 (A bit more than 10x in each step, to be precise)

    This supports the theoretical O(nlogn) time complexity.

2. Comparing the runtimes with change in **input size,** (Ascending : Descending : Random):
    - ***n=10^6-*** 116 : 107.667 : 184 ≈ 8:7:12
    - ***n=10^5-*** 13.6 : 13.3 : 15.7 ≈ 6:6:7
    - ***n=10^4-*** 1.6 : 1.2 : 1.6 ≈ 3:4:3

    We see that, on average, Ascending takes less than Descending. And Random order takes more times than the rest.

For Quick Sort:

Theoretically, for average case, time complexity is O(nlogn), for random ordering. That means, ideally, an increase of ‘n’ by 10 times, would increase the runtime by a little more than 10 times.
For worst case scenario, which happens for ascending and descending ordering inputs, time complexity is O(n^2), which means the runtime should increase by a factor of 100, for 10 times increase in n.
Now, we will analyze it with our data table.

1. Comparing the runtimes with change in ***input size,*** (n = 10^6 : 10^5: 10^4):
    - ***Ascending order-*** 1640009 : 16188 : 162.3 ≈ 10000:100:1
    - ***Descending order-*** 1393270 : 13749 : 134.4 ≈ 10000:100:1

    These support the theoretical O(n^2) Worst Case Time Complexity
    - ***Random order-*** 168 : 9.4 : 1.5 ≈ 100:10:1 (A bit more than 10x in each step, to be precise)

    This supports the theoretical O(nlogn) Average Case Time Complexity
2. Comparing the runtimes with change in ***input size,*** (Ascending : Descending),
    - ***n=10^6-*** 1640009 : 1393270 ≈ 5:4
    - ***n=10^5-*** 16188 : 13749 ≈ 5:4
    - ***n=10^4-*** 162.3 : 134.4 ≈ 5:4

    We see that, on average, Ascending takes more time than Descending. This can be explained by the fact that when sorting descending arrays, there is no swapping in the “Partition” function other than the single swap of the pivot with the first element at the end. But in sorting an ascending array, a swap is done at every step of the “Partition” function.

Merge Sort vs. Quick Sort:

- Theoretically, although both algorithms have an average case time complexity of O(nlogn), the constant associated with Merge Sort > that of Quick sort.

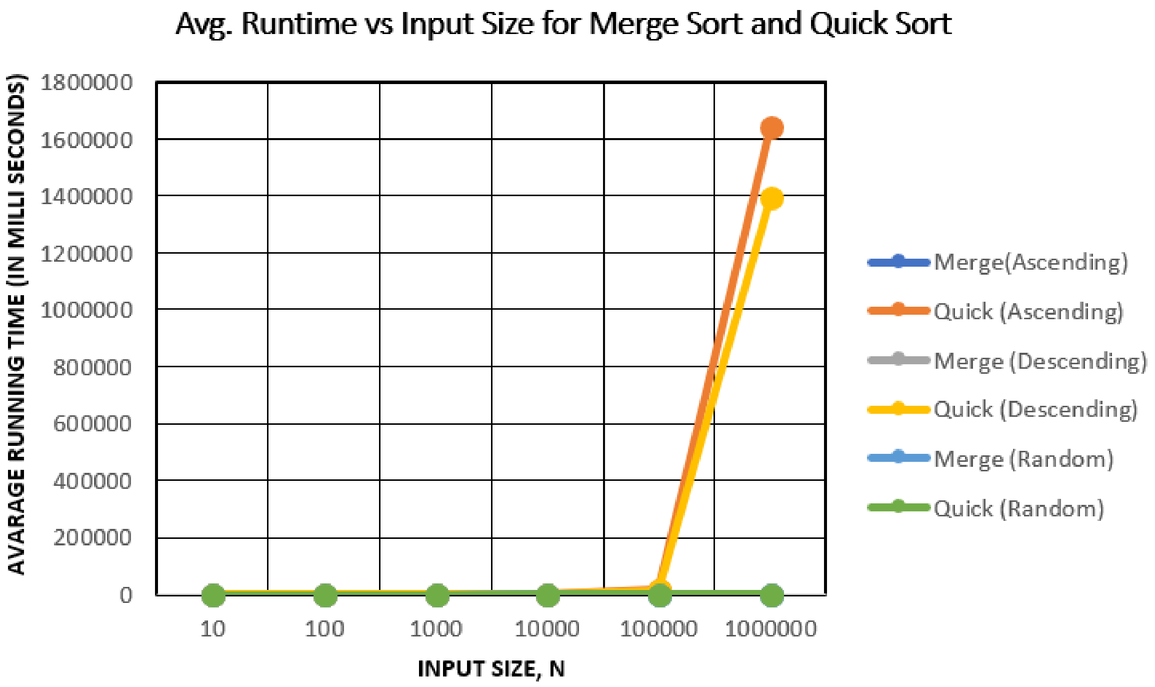
  This can be observed in sorting ***"Random Order"*** arrays. Where, time for Merge : Quick is always Greater than 1:

  184: 168 ≈ 15.7: 9.4 ≈ 1.6 : 1.5 ≈ 2 > 1
- As for sorting in ascending and descending orders, the merge sort is faster by a huge margin, since the complexity is O(nlogn) vs. the O(n^2) worst case complexity of the Quick Sort

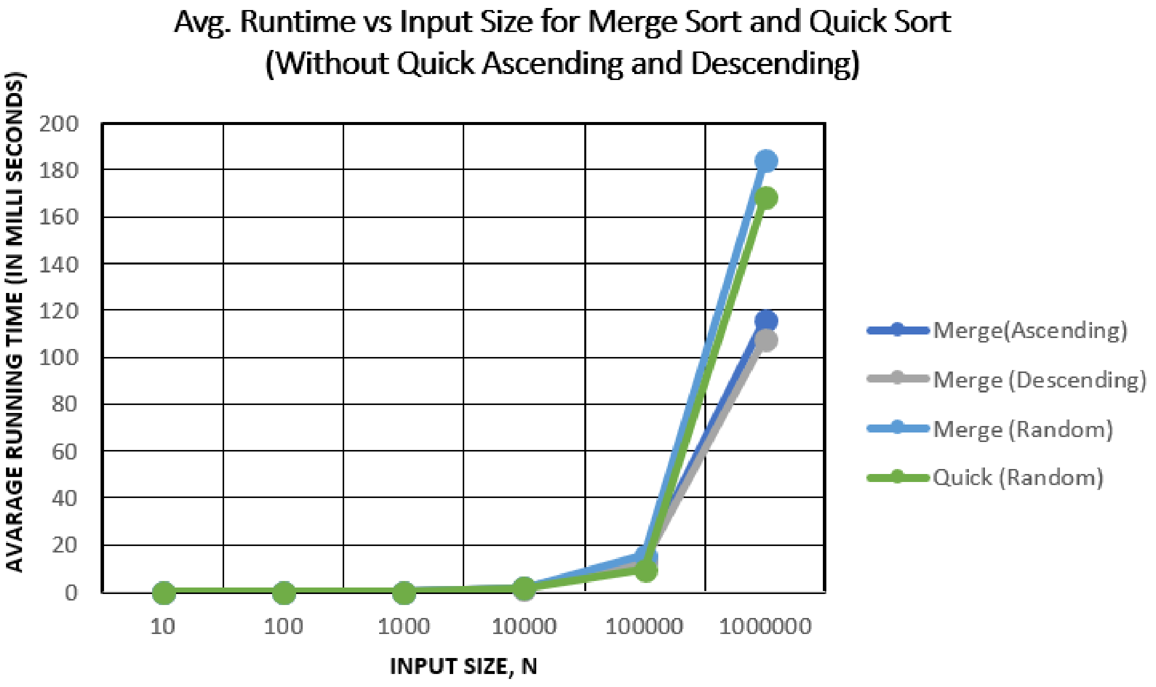
Machine Configuration:

- Processor: Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz, 1800 Mhz, 4 Core(s), 8 Logical Processor(s)
- Ram: 12GB
- System type: 64-bit operating system, x64 based processor

Graph:



As quick ascending and quick descending lines values are much higher than other lines, we can not determine other lines data properly. That's why, there is another graph to show those lines.


Avg. Runtime vs Input Size for Merge Sort and Quick Sort
(Without Quick Ascending and Descending)
AVARAGE RUNNING TIME (IN MILLI SECONDS)
INPUT SIZE, N
Merge(Ascending)
Merge (Descending)
Merge (Random)
Quick (Random)