Report on complexity of Mergesort And Quicksort

Khondker Salman Sayeed 1805050

1 Complexity Analysis

1.1 Mergesort

Mergesort is a sorting algorithm based on the divide and conquer approach. It succeeds on the basis of a linear time merging method that can produce a sorted list from two individually sorted components. Applying this method from the base case for two elements, the entire list can be sorted by doubling the sample each iteration. So if the time required to sort a list of n elements is T(n), we have the recursion,

$$T(n) <= n + 2 * T(n / 2)$$

The base case T(1) = 1.

Solving the recursion,

$$T(n) <= n + 2 * (n / 2) + 2^2 * (n / 2^2) + 2^3 * (n / 2^3) + \ldots + 2^{lg(n)} * (n / 2^{lg(n)}) = n * lg(n)$$

Therefore the order of complexity for mergesort is O(nlg(n)).

This complexity of O(n lgn) holds for ascending, descending and random order for mergesort because of the guaranteed halving of the list at each iteration.

1.2 Quicksort

Quicksort is also a sorting algorithm based on the divide and conquer approach. The key method for quicksort is partition, which is also a linear time method for dividing a list into two sublists that have the elements that belong in those sublists. However the divide is not guaranteed to be uniform. ‘Slicing’ can occur that can cause the division to be of 1 element per iteration. Therefore for this slicing effect we have time T(n) for a list of n elements,
$T(n) <= n + T(n - 1)$
The base case T(1) = 1.

Solving the recursion,
$T(n) <= n + n - 1 + n - 2 + \ldots + 1 = n * (n - 1) / 2$

Therefore the order of complexity for the worst element distribution for quicksort is $O(n^2)$.

However on most of the practical examples on sorting this ‘slicing’ effect does not occur. In average cases the partition is close to the middle. So for the average case we have,
$T(n) = n + T(n / 2) => T(n) = n * lg(n)$ (Approximation)

Therefore the average case order of growth for quicksort is O(nlg(n)).

For ascending and descending input order, quicksort falls into the ‘slicing’ pitfall and has a complexity of about $O(n^2)$.

For the random input case, quicksort out-performs the linearithmic mergesort by a constant factor and ensures O(n lgn).

2 Machine Configuration

```
Host Name:                 LAPTOP-
OS Name:                   Microsoft Windows 10 Home Single Language
OS Version:                10.0.19042 N/A Build 19042
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:
Registered Organization:   N/A
Product ID:
Original Install Date:     19-Mar-21, 2:37:05 PM
System Boot Time:          04-Jun-21, 2:47:36 PM
System Manufacturer:       ASUSTeK COMPUTER INC.
System Model:              VivoBook_ASUSLaptop X512FL_X512FL
System Type:               x64-based PC
Processor(s):              1 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 142 Stepping 11 GenuineIntel ~1792 Mhz
BIOS Version:              American Megatrends Inc. X512FL.204, 08-Apr-19
Windows Directory:         C:\WINDOWS
System Directory:          C:\WINDOWS\system32
Boot Device:               \Device\HarddiskVolume2
System Locale:             en-us;English (United States)
Input Locale:              en-us;English (United States)
Time Zone:                 (UTC+06:00) Dhaka
Total Physical Memory:     8,043 MB
Available Physical Memory: 2,745 MB
Virtual Memory: Max Size:  17,771 MB
Virtual Memory: Available: 7,878 MB
Virtual Memory: In Use:    9,893 MB
```

3 Table

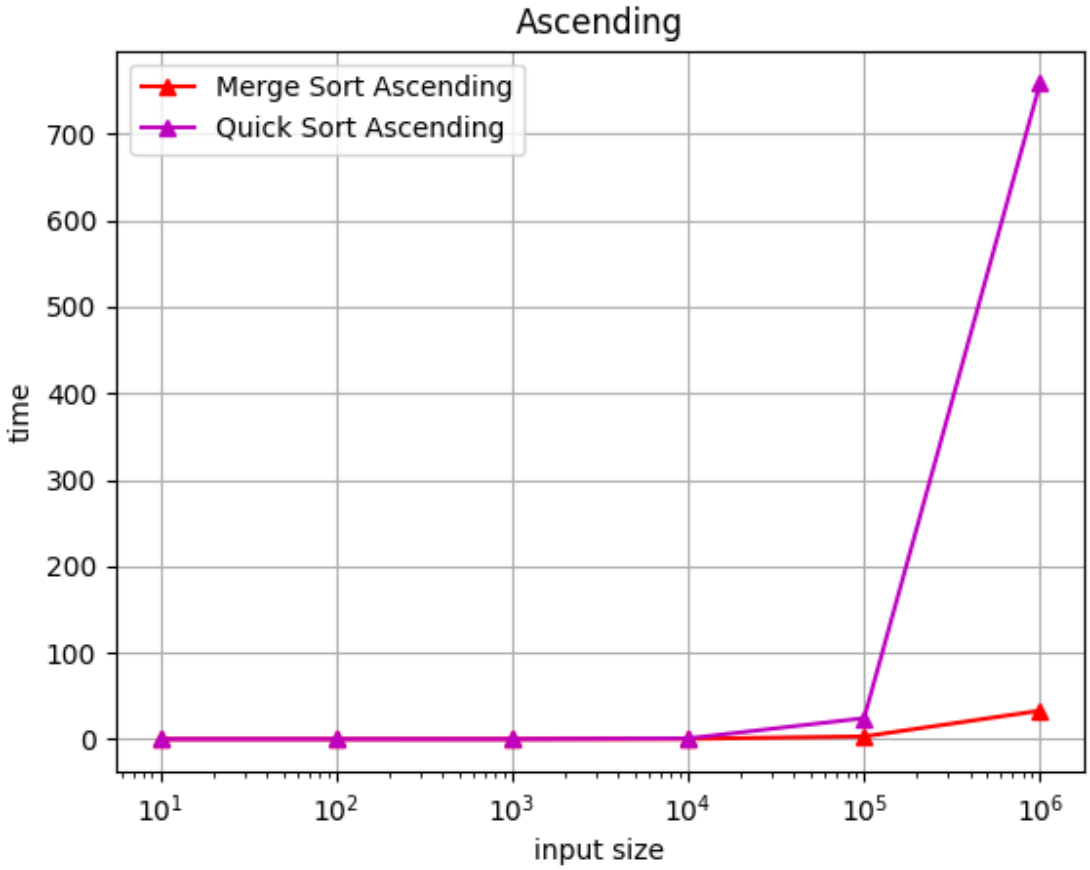
Input Order	N = / Sorting Algorithm	10 us	100 us	1000 us	10000 ms	100000 ms	1000000 ms
Ascending	**Merge**	**0.38454**	**3.6286**	**35.243**	**0.36759**	**3.1243**	**32.8754**
	Quick	**0.08683**	**2.0881**	**45.749**	**0.84635**	**24.228**	**758.546**
Descending	**Merge**	**0.35568**	**3.7317**	**32.201**	**0.8086**	**4.7529**	**50.032**
	Quick	**0.0844**	**5.0601**	**549.866**	**52.9605**	**5278.78**	**567573**
Random	**Merge**	**0.30698**	**3.0576**	**66.135**	**0.8976**	**8.6263**	**100.363**
	Quick	**0.16752**	**1.5977**	**39.389**	**0.5984**	**5.2858**	**50.2976**

4 Plot

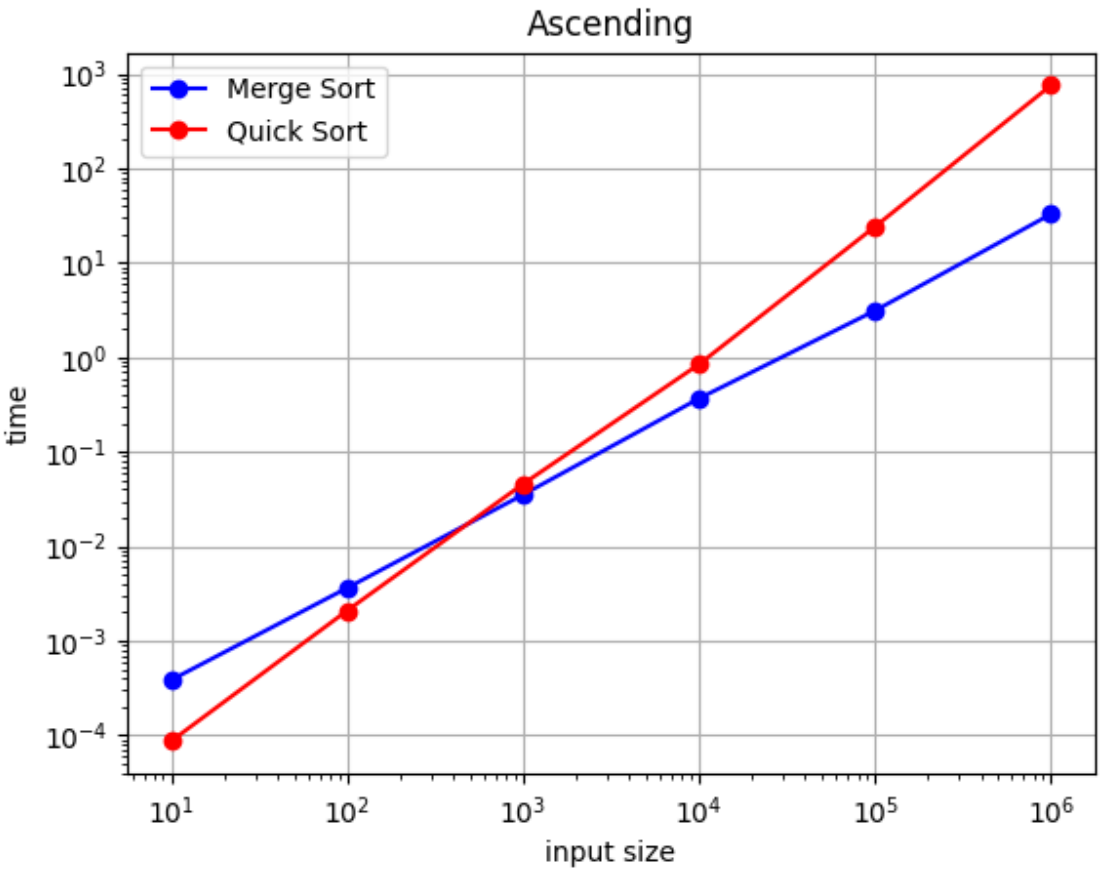
The figures are plotted in the log axis for better visualization of the linearithmic nature of the algorithms on average.

We can use the plot for **mergesort as a basis for linearithmic performance** and compare the deviation for quicksort.

4.1 Ascending Order

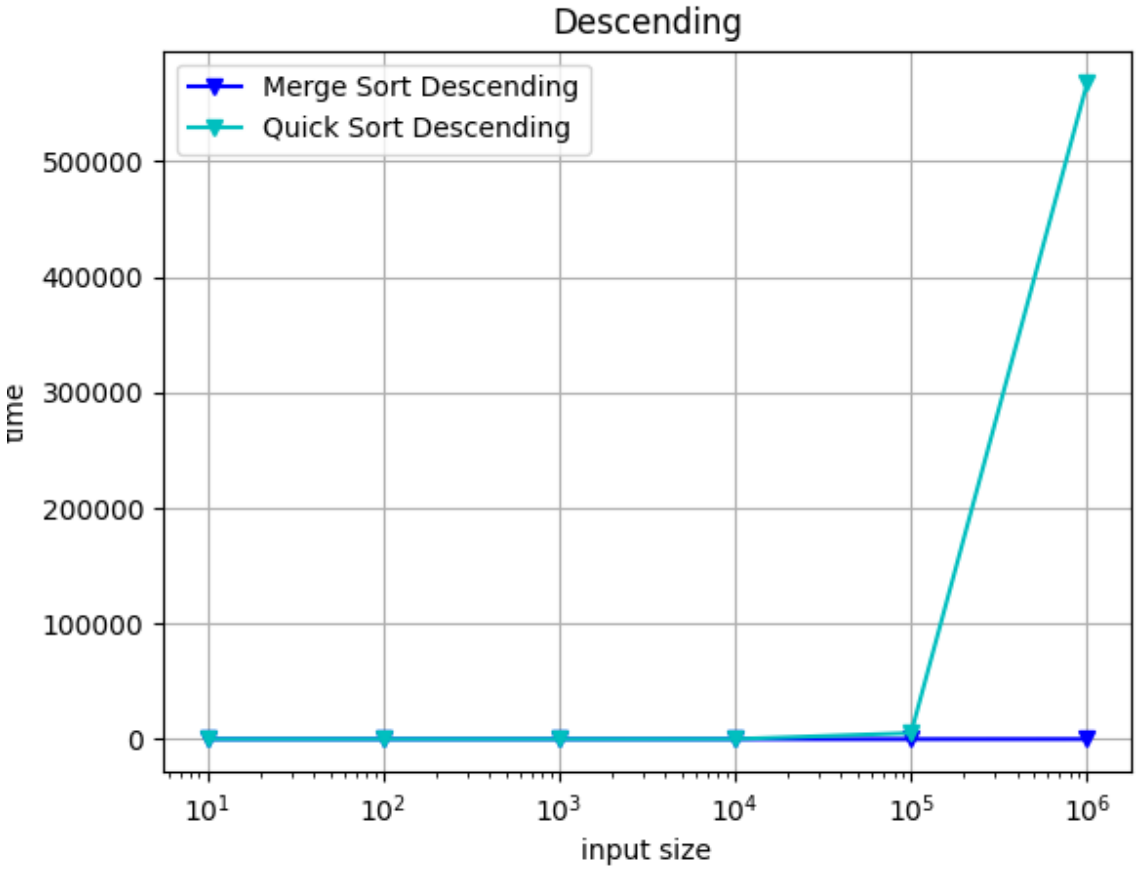


Plot on **linear scale**

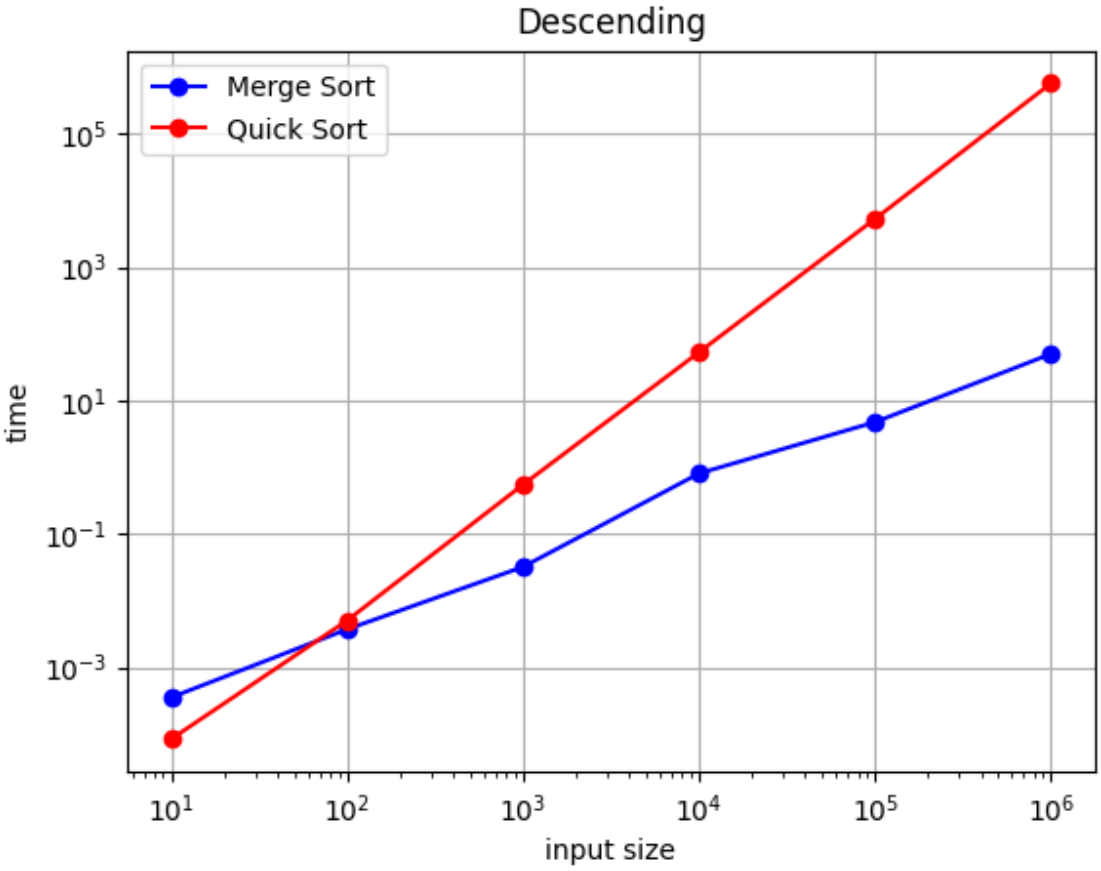


Plot in **log scale**

4.2 Descending Order

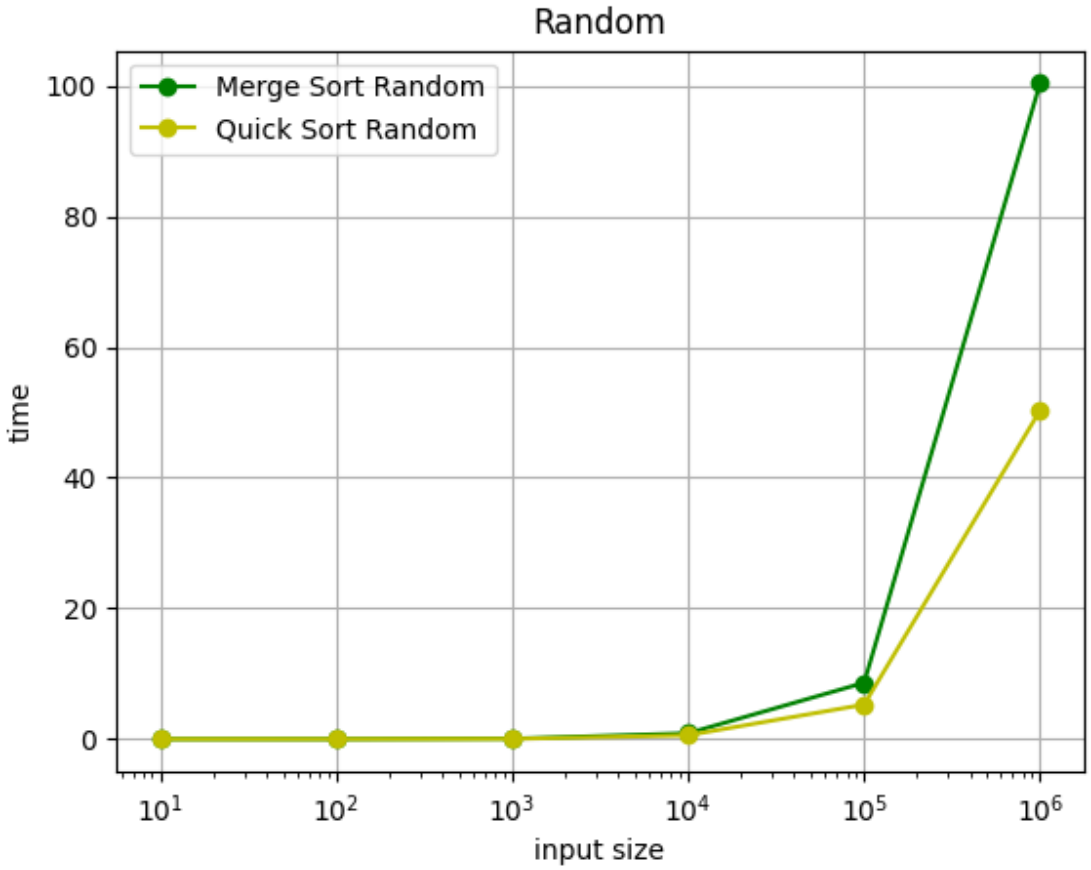


Plot on **linear scale**

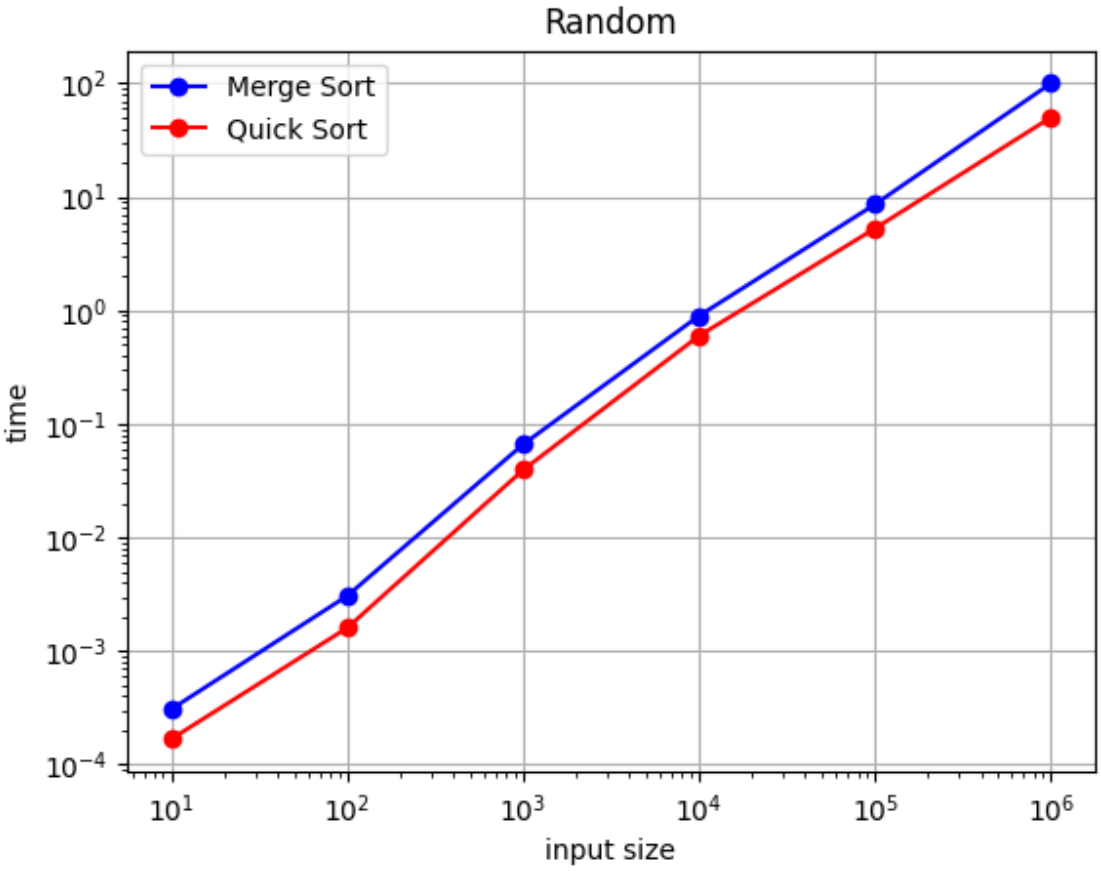


Plot in **log scale**

4.3 Random Order

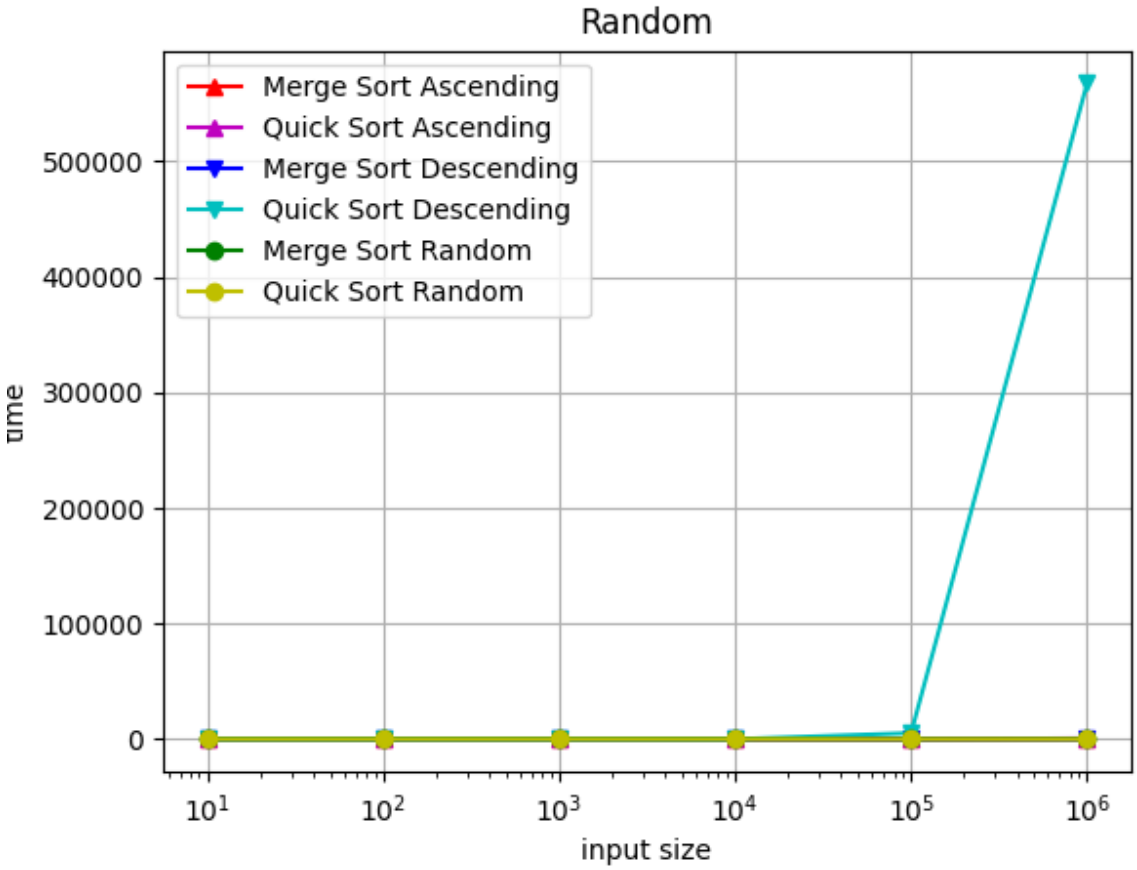


Plot on **linear scale**

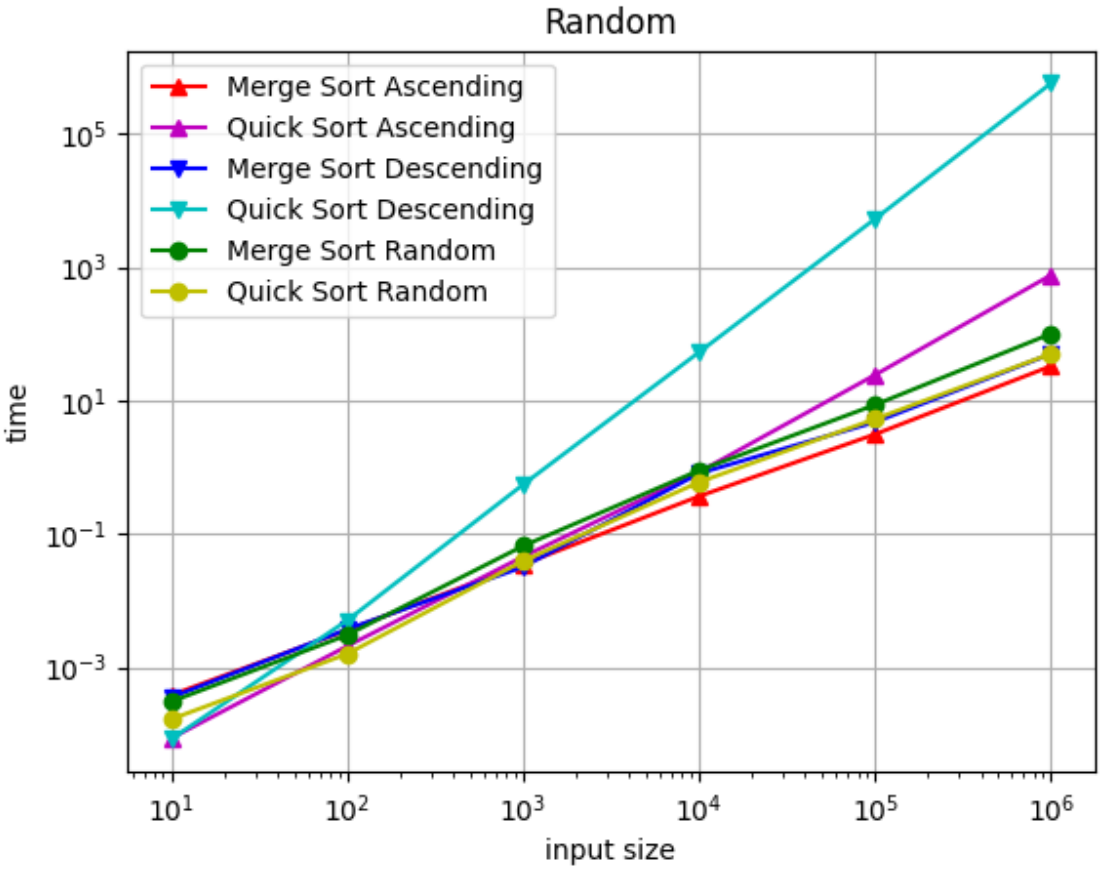


Plot in **log scale**

4.4 All In One



Plot on **linear scale**


Random
Merge Sort Ascending
Quick Sort Ascending
Merge Sort Descending
Quick Sort Descending
Merge Sort Random
Quick Sort Random
time
input size
10^{5}
10^{3}
10^{1}
10^{-1}
10^{-3}
10^{1}
10^{2}
10^{3}
10^{4}
10^{5}
10^{6}