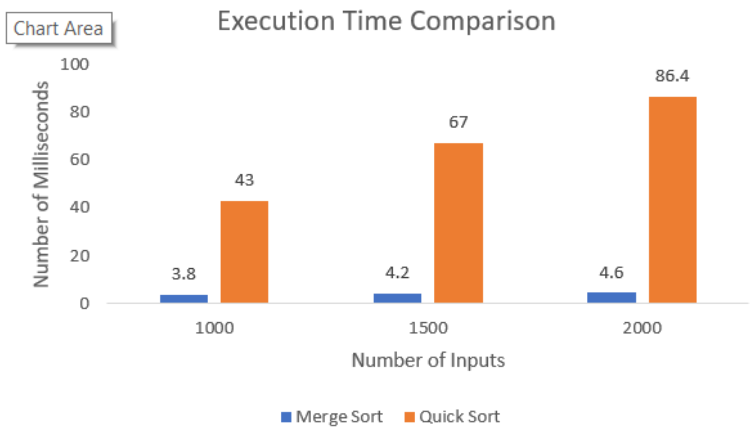
Kailash Bhanushali / 1001398090

Report on experimental results and compare, contrast between Merge and Quick Sort with its Complexity differences



After performing sorting on number of inputs which is more than 1000, we are able to describe few of the things which in short helps us to compare between both of the algorithms.

1. As we took inputs 1000, 1500, 2000 we can see that merge sort execution time is gradually increasing as it is based on (nlogn) which performs similar action without any looking out for number of inputs (independent of number of inputs). Such as while executing 1000 inputs it took 3.8 sec then later while executing 1500 inputs it took 4.2 sec and for 2000 it took 4.6 sec respectively. So, by this we can analyze that, as input has been increased by 500, merge sort taking more of 0.4 sec per 500 extra number of inputs.
2. Talking about Quick sort, as here we have selected the final value as pivot point which is considered to be the worst case (n^2), hence, the number of input as getting increased, more the time quick sort takes to execute the inputs. But, there's one more thing about quick sort is, as we see, number of input is getting decreased, there's a quick reduction in execution time. So, by this we can analyze that for few inputs Quick sort will be faster than merge sort and other algorithm such as bubble and insertion sort, even after best case for quick sort and merge sort is same (nlogn).

   Apart from the visualization we are looking out in the graph which is generated with the help of output, there are few things which helps the real world to choose Quick sort over merge sort and is generally considered to be as an **anomaly** as because in real world, there's an enormous amount of data which by looking at this graph, quick sort will perform worst as compared to merge, but actually its not the case, quick sort has quite a good number of advantages which help us to select it as a preferred algorithm for performing operation. Few of them are as:

a. We have the flexibility to select the pivot point on our own. So if we do a bit of analyze before selecting the pivot value, then quick sort can perform even 2 times faster than merge as well as heap sort, regardless of (n^2) execution time and for smaller inputs its already considered over merge sort. Hence, we can make use of quick sort to get the results quicker even in huge chunks of data.
b. Also, quick sort doesn't require additional memory to perform the operation, whereas merge sort requires additional storage and this is the another reason for choosing quick sort, as in real world transmitting data and storage of operations can be expensive which is not the case with quick sort as it performs swapping in memory.

Comparing the results just by looking above examples would produce incorrect report as explained in the previous paragraph. The other differences between this algorithm are: quick sort splits using pivot which can be selected anywhere in the array whereas merge sort has to be divided in halves. As we are moving towards smaller number of inputs quick sort, execution time of quick sort is getting reduced drastically, while in merge its all same for any number of inputs. Quick sort is considered inefficient for large number of inputs as report states, until and unless pivot selection is not done properly. Merge is considered to be the fine performing algorithm as it perform very less manipulations compared to quick sort.

There shouldn't be any huge difference between **theoretically and experimental difference**, as algorithm is built the same way as its written down, but if there's any difference then mostly it can be because of its environmental factors such as Laptop speed, load and stress and sometimes selecting the pivot point in real world after doing analyze over that array.

After, looking at the above points we can conclude that, no algorithm is considered better, it's the condition and way of implementation which make it difference and better / bad than other options.