At what array size is there a change in the "best" clock time for the algorithms?
ANSWER:

Expected output:

As we are iterating the time taken to sort using each alogorithms 100 times and finding the average, we can expect to see the theoretical average time complexity as output.

The average time complexity -
Insertion sort -n^2
Merge sort - nlog(n)
Quick sort - nlog(n)
Insertion sort can be faster for small number than quicksort because Quick Sort has extra overhead from the recursive function calls. For small arrays , merge sort also can be slower than insertion sort as merge sort needs to divide arrays and create extra arrays. Moreover, for already sorted arrays, insertion sort has a very fast sorting time with a time complexity of n. As the size of our arrays are fairly small, we can expect that insertion sort of quicksort will be the fastest.

Observed output:
After running the code, it's showing that the insertion sort is sorting the array faster for any size, followed by quick sort and lastly the merge sort. I have used the same array with random numbers for each three sortings which I kept an original copy of as (arr _1b, arr _1c) before any sorting took place. Then I iterated the sortings 100 times for each size of 16, 32, 64, 128, 256 and found their clock time individually which then I stored in counter variables like accumms1, accumis1, accumqs2 and etc …….. Later I took the average of those and stored them in variables like ms16, is16, qs16 and etc …….

After running the algorithm I have found the insertion to work faster on all sizes of arrays in average while quick works the second best and merge sort being the 3rd.

The output of my code after running in the terminal is added as a txt file. Below. i have added the average output to discuss further on the change in the best clock time for each algorithm:

For array size of 16:
Average time taken for Merge sort: 5578
Average time taken for Insertion sort: 2669
Average time taken for Quick sort: 3604

For array size of 32:
Average time taken for Merge sort: 4235
Average time taken for Insertion sort: 1843
Average time taken for Quick sort: 2000

For array size of 64:
Average time taken for Merge sort: 6576
Average time taken for Insertion sort: 4255
Average time taken for Quick sort: 4978

For array size of 128:
Average time taken for Merge sort: 13080
Average time taken for Insertion sort: 2606
Average time taken for Quick sort: 5123

For array size of 256:
Average time taken for Merge sort: 44588
Average time taken for Insertion sort: 7277
Average time taken for Quick sort: 11755

Merge sort:
Here, if we look carefully we see that the merge sort was sorting the algorithm slower than the other two. However , when the array size grew to 128 there was a huge change in the clock time from before when the array size was just 16, 32 or 64. However, it was again taking less time for 256.

Insertion sort:
Insertion sort seems to be fluctuating from size to size. However No huge change was observed here.

Quick sort:
Here, we can observe the pattern that the quick sort was taking more time as the array grew bigger. So we can conclude that here in the sorting quick sort was sorting faster when the array size was smaller.

Finally as mentioned earlier altho insertion and quick sort , both can be faster on small size of arrays , insertion sort however performs faster than the quick sort due to having the extra overhead as mentioned at the beginning. Here in our output, , we do see that the insertion is performing faster than quick sort with quicksort being the second better which is almost as close to the insertion sort.