Insertion Sort vs Merge Sort

Hypothesis

The running time of both insertion sort and merge sort will literally be the same for arrays having just 1-2 elements because these arrays are either sorted or reverse sorted. Thus, I am almost sure that their running times will intersect in that area. Also the running time of both the algorithms will intersect at some other n. I have no idea about how large this n would be so I would probably guess around 10 because insertion sort is inherently slower than merge sort and I think merge sort should take over for a n in somewhere around that range and so that will be my hypothesis.

Methods

I choose the range of N as 0-50 both inclusive. Based on my hypothesis I thought I would find the intersection point or region in that range.

For every value of N, I generate an array of random integers in the range (0,1000000). I choose a large enough range of values just to have no bias, although I am pretty sure the range of values should not have an impact on the time complexity of both the algorithms.

I measure the time taken to do both insertion sort and merge sort using the perf_counter() function of the time library in python.

One thing I understand is that using the time from just one experiment for one N can be very misleading and can highly depend on what array was sampled in that run. Therefore, to remove such ambiguities, I perform the experiment 1000 times for every N and take the average time of each run

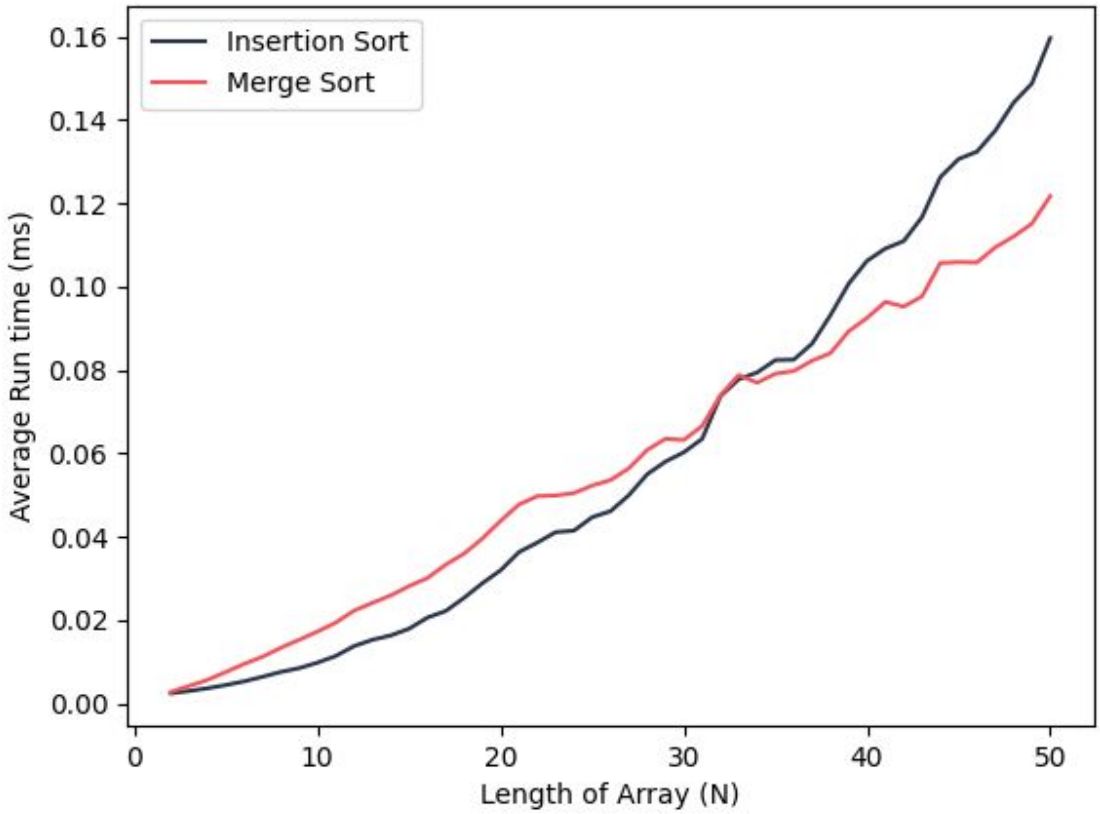
as the time complexity for that N. I believe that the average can give us a much better approximate of what the time complexity at that N is.

Finally I plot the results of run time of both the sorting algorithms using the matplotlib library in python.

The code is available at
https://github.com/divyanshaggarwal/CSE830/blob/main/HW4/Ques1/hw4_ques1.py

Results

The plot of the run-time of Insertion sort and Merge sort over different values of N is shown as below. The x-axis represents the size of the array being sorted (N) and the y-axis represents the average running time of both the sorting algorithms in ms over a 1000 runs.



The run-time of both Insertion Sort and Merge Sort are literally the same in the range of 32-34 i.e when the length of the array being sorted is in the range 32-34.

Discussion

From the plot we observe that the run-time complexity of both insertion sort and merge sort are the same in the range of 32-34. For smaller values of N, insertion sort is indeed a faster sorting algorithm than merge sort but for larger values of N, when the asymptotic time complexity comes into play, the time complexity of merge sort becomes less than insertion sort. Also, we observe that after 34 the difference in the run-time of merge sort and insertion sort keeps on increasing and merge sort increasingly becomes a much efficient algorithm for sorting which is expected as the asymptotic running time of merge sort is O(nlogn) and that of insertion sort is $O(n^2)$.

Conclusion

I hereby conclude that below 32, insertion sort is indeed a faster sorting algorithm than merge sort. In the range 32-34, there is not much difference in the running time complexity of both the sorting algorithms and therefore we can use either one of them. But if the array to be sorted is greater than 34, merge sort provides a much efficient and faster sorting procedure than insertion sort and therefore we should prefer merge sort for larger array sizes.