CSCI – B505 Applied Algorithms

Assignment – 2

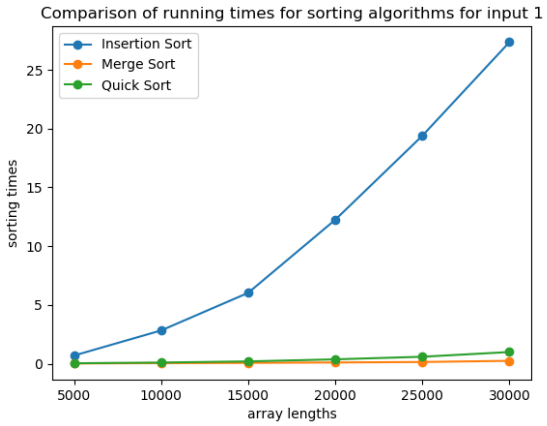
Submitted by – Vishal Singh (singhvis)

Input 1

Runtimes-

Array Size	Insertion Sort runtimes (in sec)	Merge Sort runtimes (in sec)	Quick Sort runtimes (in sec)
5000	0.65799	0.02302	0.04525
10000	3.53089	0.10665	0.24091
15000	11.59871	0.07794	0.19233
20000	17.48384	0.13543	0.47599
25000	30.63683	0.17747	0.77543
30000	43.41313	0.21656	0.9484

Plot-



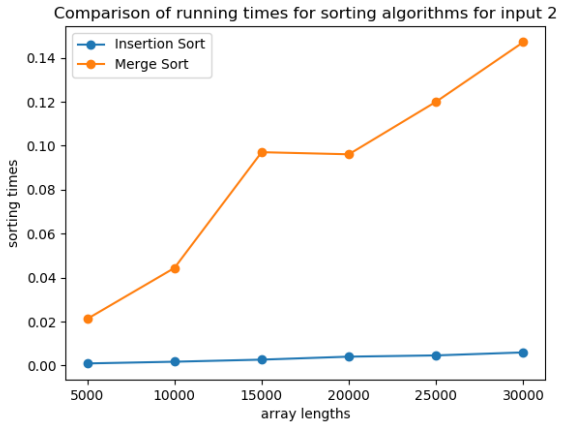
Input 2

Runtimes-

Array Size	Insertion Sort runtimes (in sec)	Merge Sort runtimes (in sec)	Quick Sort runtimes (in sec)
5000	0.00094	0.0212	Kernel Died error
10000	0.00172	0.04438	Kernel Died error
15000	0.00267	0.09707	Kernel Died error
20000	0.00402	0.09612	Kernel Died error
25000	0.0046	0.12003	Kernel Died error
30000	0.00596	0.14715	Kernel Died error

Note: Kernel Died error was occurred even when recursion limit was set to 10000

Plot-

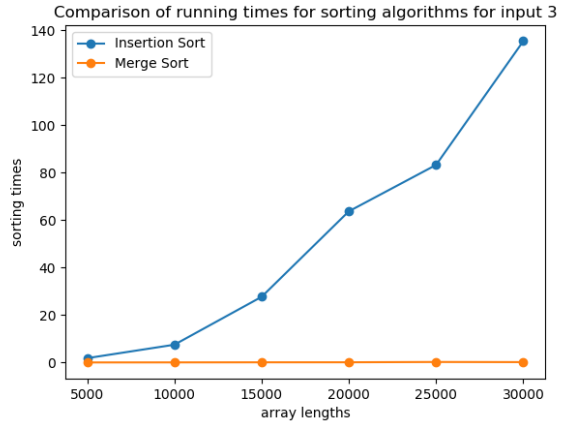


Input 3

Runtime-

Array Size	Insertion Sort runtimes (in sec)	Merge Sort runtimes (in sec)	Quick Sort runtimes (in sec)
5000	2.23647	0.02183	Kernel Died error
10000	8.02623	0.07625	Kernel Died error
15000	26.34587	0.06786	Kernel Died error
20000	46.19113	0.10374	Kernel Died error
25000	79.41371	0.12445	Kernel Died error
30000	111.27105	0.46046	Kernel Died error

Plot-

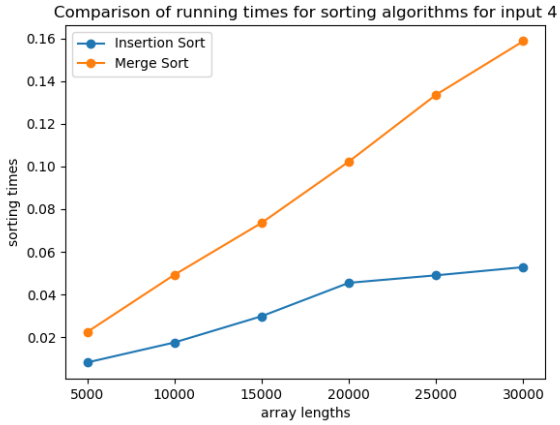


Input 4

Runtimes-

Array Size	Insertion Sort runtimes (in sec)	Merge Sort runtimes (in sec)	Quick Sort runtimes (in sec)
5000	0.00821	0.02246	79.4808487
10000	0.01757	0.04928	334.8756467
15000	0.02987	0.07359	902.3560956
20000	0.04551	0.10229	Takes too long
25000	0.04902	0.13359	Takes too long
30000	0.05285	0.1587	Takes too long

Plot-



Input 5

Total insertion sort running time for input 5: 29.02201

Total merge sort running time for input 5: 26.93882

Total quick sort running time for input 5: 39.25593

Discussion

I choose Python for this programming assignment as being a data science student, it is my most preferred language. I'm also most familiar with visualization (plotting) techniques used in Python. I stored the data required to make plots as a list. I construction separate lists for the three algorithms. I chose Pycharm environment for writing code, the running times were slower than Spyder runtimes which were obtained in Programming assignment 1

The first plot (for Input 1) shows an asymptotic quadratic plot for insertion sort algorithm which quick sort and merge sort both take significantly less time since their running time is O(nlogn). The time taken for

insertion sort algorithm is most than other sorting algorithms (of O(nlogn)) for all values of n and the difference increases as n increases. For the other two algorithms, merge sort is slightly faster than quick sort even though both take < 1 sec

In the second plot, the time taken for insertion sort is quite less (< 0.01sec) as the array is already sorted. This is the best-case scenario for insertion sort algorithm. The algorithm executes in linear time. Merge takes almost same time for this input as for input 1 which is a property of merge sort that it takes almost same time for any input. Quick sort initially gave an error of 'recursion limit reached' even for input of size 5000, on increasing the recursion limit to 10000, it threw a 'kernel died' error

In the third plot, the time taken for insertion sort is higher than merge sort. It is the worst possible case for the insertion sort. The time taken by insertion sort is relatively high (112 sec for array of size 30k). Merge sort takes similar time as in input 1, 2 (<1 sec)

In the fourth plot few of the number are swapped from a sorted array this accounts for a very small share so the times are comparable to plot 2, we can see times for insertion sort are slightly higher than the ones in plot 2 due to the few numbers being swapped. Insertion sort is faster than merge sort in this case which takes almost same time as in previous input. Quick sort takes a large amount of time in this case. I've run it separately and hence it is not plotted.

In the output for input 5, we can see that the running times are not much different for the three algorithms for small input of 50. We can see that merge sort is the fastest whereas insertion sort is faster than quick sort in this case

While there is no preferred algorithm for small size, I would prefer using either quick sort or merge sort for arrays of large size(provided the array is not already sorted).