Thomas Weis
ID: # 5725560

Course Title: Sequential and parallel Algorithm
Course Number & Section: COMP 273-01
Homework #2
(Comparative Analysis of Sort Algorithms)

Spring 2024

March 18 -31, 2024

Ex. 1
Using the Java code of the Experiment class and the various sort algorithms classes we have discussed during lectures, complete the table given below. **[30 pts]**

sn	Input size, n	Time(T) taken by sort algorithm for each input size(n)					
		MergeSort TD	MergeSort BU	QuickSort	InsertionSort	SelectionSort	Radix Sort (LSD or MSD)
1.	100	.003 s	.003 s	.001 s	0.0 s	0.0 s	.004 s
2.	1000	.002 s	.001 s	0.0 s	.013 s	.02 s	.067 s
3.	10000	.000 s	.013 s	0.0 s	.216 s	.141 s	5.006 s
4.	50000	.012 s	.03 s	.032 s	8.639 s	7.716s	107.044 s
5.	100000	.063 s	.072 s	.072 s	58.326 s	33.728 s	479.748 s
6.	200000	.102 s	.112 s	.103 s	267.483 s	146.682 s	2052.055 s
8.	300000	.314 s	.25 s	.188 s	494.339 s	140.319 s	4708.15 s
9.	500000	.252 s	.283 s	.253 s	4289.36 s	1474.283 s	13356.978 s
10	1000000	.515 s	.64 s	.568 s	22152.817 s (or 6.1 hours)	7462.338 s (or 2.07 hours)	53881.296 s (or 14.97 hours)

Ex. 2
Write your observations based on your experimental results you obtain in the above table. **[10 pts]**

The first three items (both merge sorts and the quick sort) appeared to be the fastest by far. None of these times exceeded one second. MergeSortTD appears to have run the fastest, but the times were not very different at all.
The InsertionSort sort appears to be the least efficient with SelectionSort being second least efficient. These both have a high parabolic growth over its change over time where the previous three could almost be mistaken for a more linear-looking function.

The implemented LSD function appears to be the least efficient by far. This might be my implementation of the sorting solution, as the function in the text book worked for strings and I tried to make it work for Doubles (which it does) but this might have impacted the efficiency.

Note:

1. *I have included the Experiment (modified version) and Stopwatch classes (Java codes) as the driver classes.*
2. *Implement the various sorting algorithms using the techniques discussed in the lecture slides (or recommended textbook). Do not copy codes from the internet or other sources.*
3. *Upload your solution codes.* ***[60 pts]***
4. *Complete the attached report document and submit it (in person during class) or upload it via Moodle (Online).* ***[40 pts]***
5. *[Optional] 10 pts bonus credit if you can implement and use the radix sort algorithm described on pages 707 and 712 of course the recommended textbook.*