SYSC 2100 Lab 11

Matthé Bekkers

March 2025

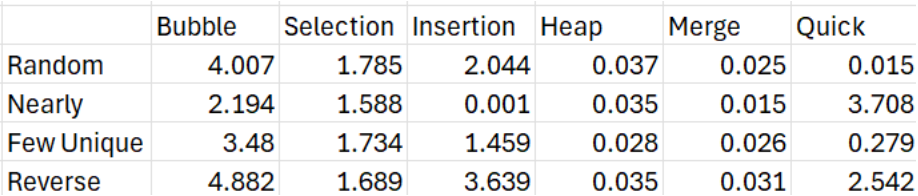
1 Questions

1.1 Q1: For each sorting algorithm, do the experimental results agree with the theory?

Yes, they do. Bubble sort is $O(n^2)$ and swaps elements at a very high frequency, which reflects in its extremely long runtime. Similarly, selection and insertion sorts are also $O(n^2)$, but make less comparisons and swaps, resulting in noticeable better performance.
Heap sort and merge sort are both $O(n\ log(n))$, which is significantly faster than the $O(n^2)$ algorithms. This is shown through their consistently very fast performance accross all test cases.
Quick sort is a bit stranger since it is technically $O(n^2)$, but performs quite well compared to the other n^2 algorithms, with the exception of the nearly sorted case. This is because quick sort partitions the list into 2 partitions, which in the nearly sorted case ends up being an element towards the extremities of the list, leading to worse performance, since a larger partition must be sorted, bringing the complexity closer to $O(n^2)$.

1.2 Q2: Is heapsort faster than bubble sort? Is heapsort faster than merge sort? Is quick sort faster than the other algorithms? If not, can you suggest a reason why your experimental results don't agree with the theory?

Heap sort is *significantly* faster than bubble sort, since, who knew, $O(n\ log(n))$ is way faster than $O(n^2)$! Quick sort is actually not all that faster than the other algorithms except in the fully random case, due to the reasons explained above. Quick sort is really optimized for a fully random case, and it struggles in any other case.



	Bubble	Selection	Insertion	Heap	Merge	Quick
Random	4.007	1.785	2.044	0.037	0.025	0.015
Nearly	2.194	1.588	0.001	0.035	0.015	3.708
Few Unique	3.48	1.734	1.459	0.028	0.026	0.279
Reverse	4.882	1.689	3.639	0.035	0.031	2.542

Figure 1: Table of Running Times

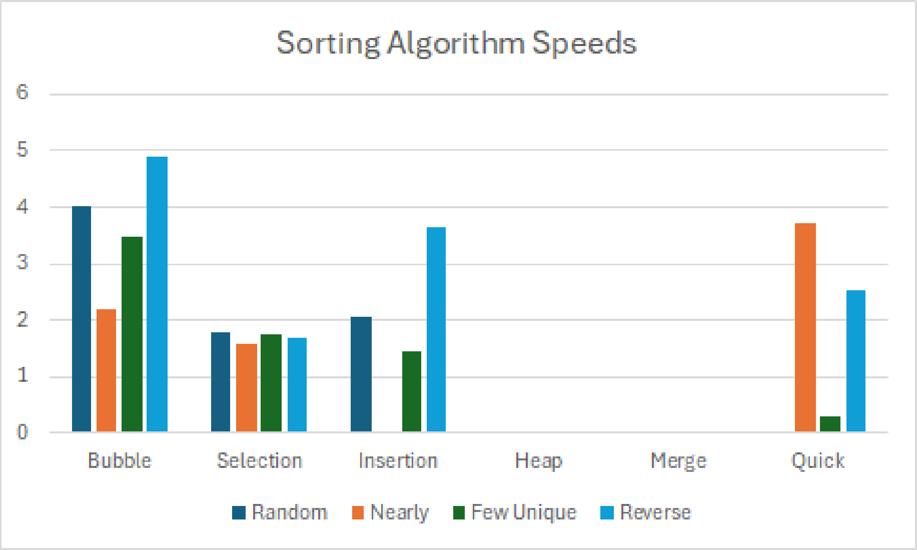


Figure 2: Graph of Running Times