1 Theoretical Analysis

The theoretical analysis of all the sorting algorithms were covered in class. For reference, we have the table of the relevant sorting algorithms and their efficiency for the best, worst, and average cases.

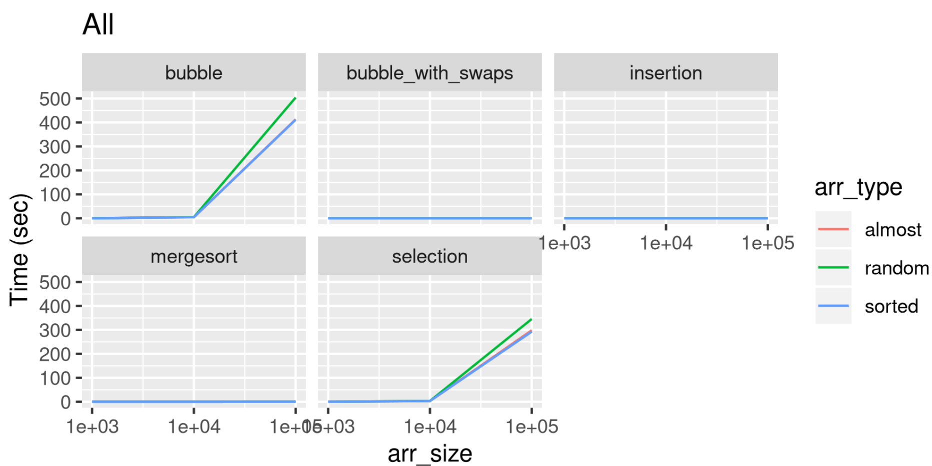
Algorithm	Best	Worst	Average
Selection	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$
Insertion	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$
Bubble	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$
Bubble (with swaps)	$\Theta(1)$	$\Theta(n^2)$	$\Theta(n^2)$
Quick	$\Theta(n \log n)$	$\Theta(n^2)$	$\Theta(n \log n)$
Merge	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$

2 Empirical Analysis

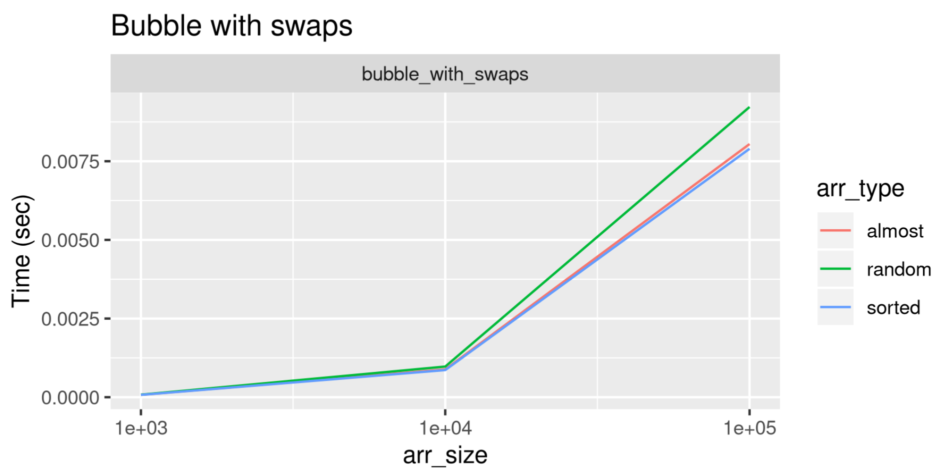
The following section covers the empirical analysis of the above sorting algorithms. Due to memory issues (for more details see), quicksort gets its own section. All the algorithms were implemented in Python, and the source code can be found on GitHub. The tables with the raw data for the analysis can be found in section BLANK.

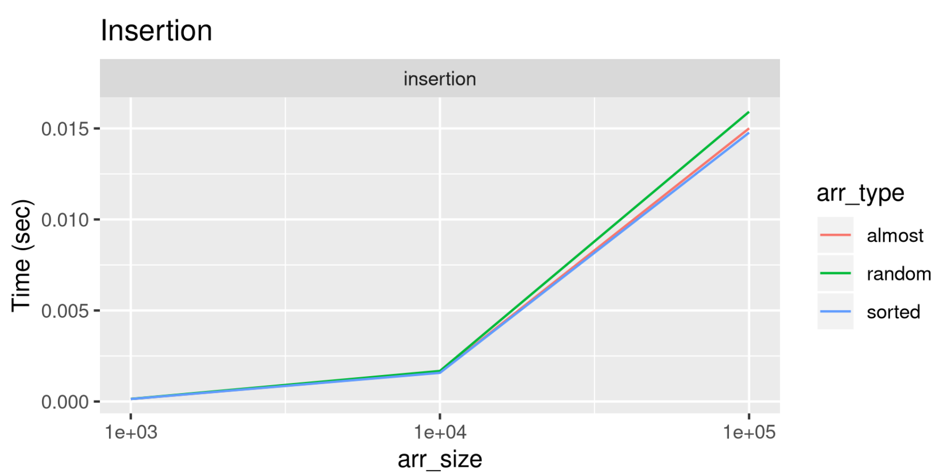
2.1 Graphs and Discussion

We have the graphs from all the algoritms below.

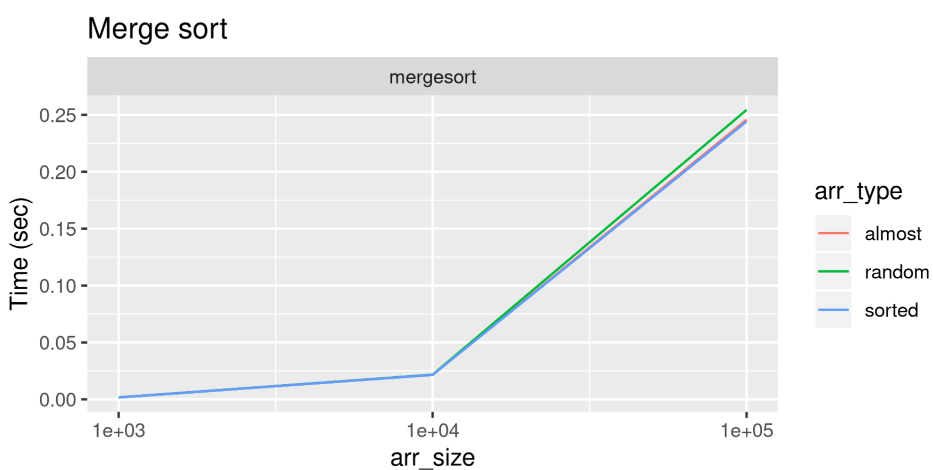


From the graphs we can see that bubble and selection sort grow much quicker than the other sorting algoritms. So much so, that the scale of the vertical axis makes it to where the other algorithms are basically a flat line. To see the growth more clearly of the other sorting algoritms, we have separate graphs.





It is hard to tell with only three data points, but we can see that all the sorting algorithms have similar shapes, albeit with bubble and selection having much steeper curves.


Merge sort
mergesort
Time (sec)
0.25
0.20
0.15
0.10
0.05
0.00
1e+03
1e+04
1e+05
arr_size
arr_type
almost
random
sorted