Christopher Mills-Bowling
chrismillsbowling@csu.fullerton.edu
888894185
Project 2 - Algorithms

Data:

Data: Sort runs by seconds:

Mergesort @ words = 10000: 2.55, 2.47, 2.51, 2.41, 2.44, 2.48, 2.52(s)

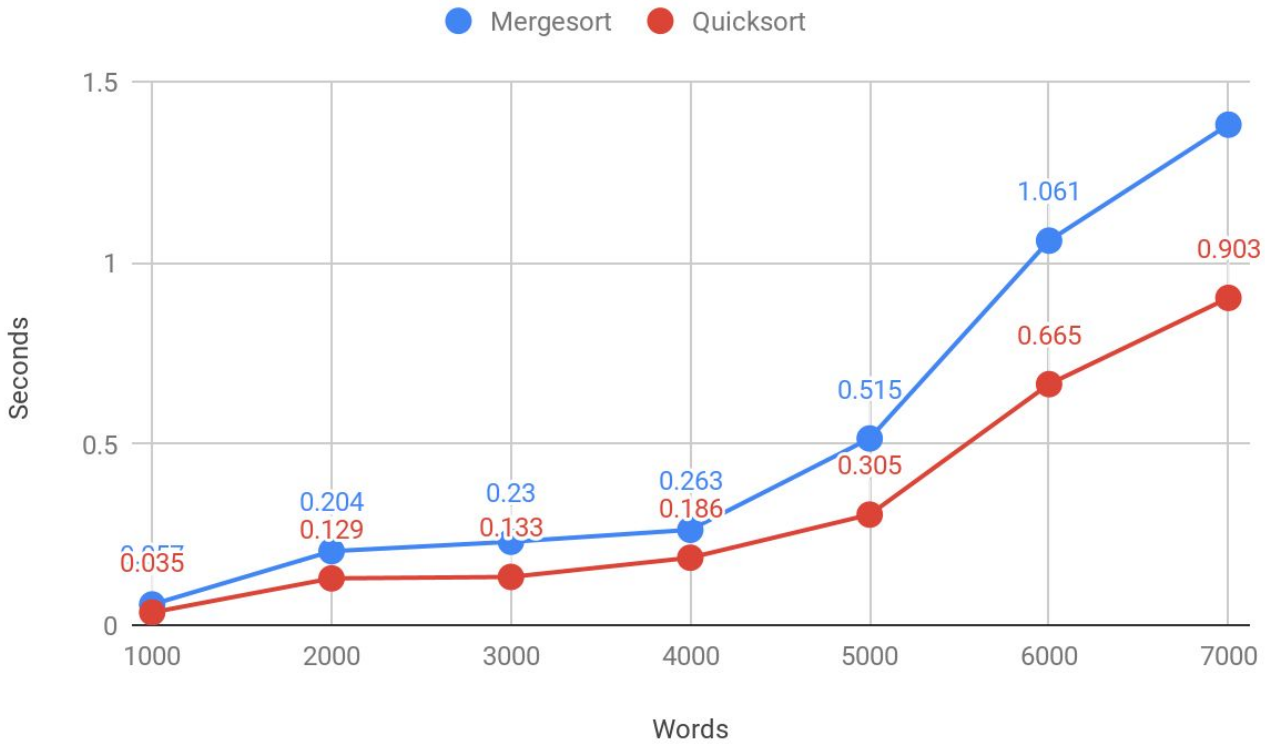
Quicksort @ words = 10000: 1.73, 1.84, 1.76, 1.75, 1.77, 1.76, 1.80(s)

Mergesort @ words = 10000:
Mean = 2.48s
STD Deviation = 0.0482s

Quicksort @ words = 10000:
Mean = 1.77s
STD Deviation = 0.0364s

Data Analysis:

Average performance:
The average performance of Mergesort was generally slower than that of Quicksort. In the two 7 run tests conducted for the mean calculations at 10,000 words the results between the two algorithms sported a 31%~ difference. The differences seen can be attributed to favorable data sorting for each algorithm being different. Quicksort performs especially well in cases that are in nearly sorted organization. Quicksort performs terribly in cases of reverse ordering where it often approaches its worst case. Mergesort is often more average due to it not having a frequently occurring favorable case or worst case. The average performance is indicative of a dataset that tended to randomise more sorted least to greatest than not.



Range
Quicksort:
Worst complexity: n^2
Average complexity: n*log(n)
Best complexity: n*log(n)
Mergesort:
Worst complexity: n*log(n)
Average complexity: n*log(n)
Best complexity: n*log(n)

As mentioned before the average performance of both algorithms are very similar, but the rate at which their upper and lower bounds are achievable are what make the difference. To utilize quicksort the best, when implementing it is important to be aware of its worst case of a reverse ordered list which causes the n^2 result seen above. Given the potential of reverse ordered lists, mergesort's range is much more beneficial for being consistent.
Standard Deviation
The standard deviation of Mergesort at 10,000 words was 0.0482 seconds and 0.0364 seconds for Quicksort. This is indicative of a dataset that played more towards Quicksort lower range order due to the consistent efficiency of it and its overall out performance of quicksort. Mergesort's Standard deviation being slightly higher than Quicksort is due to, again, mergesort's optimal case being a much more rare randomly occurring event than a slightly ordered list.

The Hypothesis

This experiment tested the following hypotheses:

1. Randomization can be used to generate data for testing an algorithm and determining performance.
2. Two algorithms of the same efficiency class can have different average running times and different ranges of performance.

During this project I concluded that both hypothesis can be considered likely true. For data randomization, this can be seen in the observation of the Standard Deviation and Range observations of the two sorting algorithms. Second, the two algorithms tested in this project showed clearly different running times despite having similar hypothetical order of growths.