Jacob Salmon

Prof. Ramirez

Data Structures

17 November 2020

Assignment 04 Results

1. **Which algorithm was best for random data?**

Perhaps it is easier to answer this question by beginning with what we know is *not* the most efficient for random data. With this approach, we can start by eliminating Simple Quicksort, which did not perform exceedingly well in any of the random/min size combinations. For base cases of 50 and 150, Simple Quicksort is actually visibly slower than the other three algorithms even before it is eliminated. The other three are in much closer competition, but I would say that overall MergeSort is the best choice for randomly organized data. The difference between MergeSort and other algorithms for random data with MIN\_SIZE = 5 is quite significant, especially with the larger array sizes. Even for other random/MIN\_SIZE combinations, MergeSort still observably outperforms the other algorithms in the large array sizes. Overall, I’d say for large array sizes (200K+) MergeSort is without a doubt your best choice, but for smaller array sizes you could go with anything but Simple Quicksort and get similar results.

1. **Which algorithm was best for sorted data?**

My results for sorted data were surprisingly similar to those I got for random data. Mergesort significantly outperformed the other algorithms, particularly in large array sizes of above 200,000. The only difference between my answer here and my answer for random data is that I said for random data with smaller array sizes you could practically use any algorithm you wanted and get similar results; here, for sorted data, MergeSort is marginally better even with small array sizes. Overall, for sorted data, I’d say that MergeSort is your best bet regardless of array size or MIN\_SIZE.

1. **Which algorithm was best for reverse sorted data?**

For data sorted in reverse, the results were actually different than the two previous questions. Here, for all three reverse sorted and MIN\_SIZE combinations, MergeSort actually performed marginally *worse* than Random Pivot and Median of 3 Quicksort. The results for the best algorithm were close between Random Pivot and Median of 3, especially at the lower array sizes, but when experimenting with array sizes over 1 million it became clear that Median of 3 Quicksort would edge out Random Pivot ever so slightly. Overall, for reverse sorted data, Median of 3 Quicksort is your best choice.

1. **Which MIN\_SIZE value gave you the best performance, and which gave you the worst?**

This question is tough to answer 100% definitively, as the answers vary heavily for array size, algorithm, and initial data setup. Doing some math to average all my results from either MIN\_SIZE = 5, 50, or 150, however, it seems that using MIN\_SIZE = 5 provides the slowest runtime overall. For whatever reason, MIN\_SIZE = 5 with Random Pivot Quicksort provided particularly slow runtime results. MIN\_SIZE = 50 and 150 provided similar averaged runtimes, but MIN\_SIZE = 50 edged it out by a margin of just .025 seconds (on average). Interestingly enough, my results for sorted with MIN\_SIZE = 50 and 150 were both some of the quickest results I had in any table out of all my observations. Overall, however, I’d say that MIN\_SIZE = 50 is the best choice overall, going off of the averages that I calculated.

* MIN\_SIZE = 5 AVG = 0.30821
* MIN\_SIZE = 50 AVG = 0.2875
* MIN\_SIZE = 150 AVG = 0.2931

1. **Based on all of your results, if you had to pick one algorithm, which algorithm would you choose?**

In this question, just like the last one, it’s hard to give a definitive *best* algorithm, without knowing the array setup, MIN\_SIZE, etc. However, if we think about which data setup we are most likely to run into, the answer is without a doubt randomly sorted (simply because in the vast majority of cases where we sort, the data is not already sorted/reverse sorted). Because of this, if I had to select an algorithm without information on the data setup, etc. I would pick MergeSort because of its consistently good performance in the random data setup category, the setup we are most likely going to run into.