Jen Wang

Dr. Ramirez

CS 0445 S21

Assignment 4 Report

After the five runs of random data and four runs of sorted data, the empirical-runtime comparisons came out matching my expectations from the asymptotic runtimes.

When data is normal/random, QuickSort yields better performance than MergeSort for all runs. The best runtime of MergeSort stays steadily around 197% of the best runtime of the best performing type of QuickSort when the data is random. Because MergeSort, unlike QuickSort, does not sort in place and requires copying of data into an extra array, the overhead yields a longer runtime.

Within the QuickSort algorithms for random data, the best performing sort sways back and forth between Simple Pivot QuickSort and Median of Three QuickSort. For all the run-sizes except for 2,560,000, Simple Pivot yields slightly better runtime than Median of Three, with a minimum recursion of 15 as best result for Simple Pivot. For the runs of size 2,560,000, Median of Three yields slightly better results. Because the average asymptotic runtime for all three QuickSort variations is the same O(Nlog2N), their performances are expected to be similar. Median of Three QuickSort was designed to make the worst case of Simple Pivot QuickSort less likely. As data size increases, the number of chunks of somewhat sorted data being assigned together increases. When more of the data is sorted, Simple Pivot's performance drops, and Median of Three would start to outperform Simple Pivot. However, since the data was originally assigned at random, it is unlikely that the data would become so sorted that Median of Three would outperform Simple Pivot by the amount that QuickSort outperformed MergeSort. The result of my runs show that for a smaller sample size, Simple Pivot’s runtime is faster than Median of Three’s runtime by an average of 2%, and for a larger sample size, Median of Three’s runtime is faster than Simple Pivot’s by an average of 6%. However, for the smallest data size, the difference between the average runtimes of the two algorithms is less than 1%. Random Pivot QuickSort falls behind the other two variations only slightly, which could be explained by the overhead in choosing the random index.

When data is sorted, one can see a clear difference between the runtime of QuickSort and MergeSort, especially as data size increases. MergeSort greatly outperforms QuickSort when data is sorted. This is expected because Simple Pivot QuickSort yields worst runtime of O(N2) when the data is sorted, while MergeSort yields a runtime of O(Nlog2N). Median of Three and Random Pivot QuickSort perform much better than Simple Pivot because they are avoiding the worst-case runtime by not decrementing pivot by one each time it runs, but their runtimes are still much longer compared to MergeSort. Random Pivot and Median of Three runtimes are on average 350% slower than MergeSort’s runtime, while Simple Pivot’s runtime is on average 4000% slower than MergeSort’s, which is mind-blowing.

Regarding the minimum recursion value, for random data, QuickSort algorithms almost always yield best results when the value is 15 and yield worst results when the values are 5. When the value is 15, the work of InsertionSort is less than the work done if we keep partitioning; thereby yielding a better result. Although MergeSort performs worse than QuickSort when the data is random, the number of recursions to stop at which yield a worst result is also 5. This outcome was expected because MergeSort requires much overhead from copying data, so as recursion sizes become small, the work of copying data overtakes the work done if we were to use InsertionSort, yielding a longer runtime if we stopped at a small size of 5.

Because our data was homogeneous integers, for normal cases, we would want to use QuickSort with a minimum recursion size of around 15. Median of Three would be a good choice because it makes the worst case of Simple Pivot Quicksort less obvious. From my results, which yielded Simple Pivot QuickSort as the faster algorithm, one could argue that Simple Pivot QuickSort is good enough as well. I do not recommend Random Pivot QuickSort simply because the overhead in randomizing data is not worth the extra time. For cases where we know the data is sorted or mostly sorted, MergeSort with a minimum recursion size around 35 would work well, despite the overhead in not sorting in place. However, if we are sorting non-homogeneous data, where there are strings, numbers, etc together, MergeSort would work better because it is stable.