CS 0445: Assignment 4

Due on April 17, 2021 at 6:00pm

Professor Ramirez TuTh 1:15PM

Made using LaTeX/Inkscape. Source files available upon request.

Shane Riley

Introduction and Methodology

This document serves as an explanation of the results turned in for Assignment 4, namely a discussion of the observed relative performance of the algorithms. A brief recap of the assignment: To examine the relative performance of each algorithm under different array sizes for sorted or random conditions, each sort method is run and timed. For each algorithm, the insertion sort threshold (Min Recurse) is iterated to find the fastest and slowest configuration. 10 runs per configuration are made per configuration and the time is averaged to produce more trustworthy results.

The situation descriptors (array size, number of runs to average, and whether sorted) are listed as run arguments to the Assig4 main method. To store the test informtation for output, an additional class "TestAlg" is presented. It records in its state the best and worst cases for the algorithm in question. The sorting methods are listed:

1. QuickSort with Simple Pivot
2. QuickSort with Median-of-Three Pivot
3. QuickSort with Random Pivot
4. MergeSort

The min recurse value is tried from 5 to 75 in intervals of 10. The configurations tried are 10000, 20000, 40000 and 80000 for sorted data, and 320000, 640000, 1280000, 2560000 and 5120000 for random data. The stack size is increased to avoid overflow.

Results

In all cases for random data, the best sorting method is always one of the QuickSort's. Which pivot is best tends to be random, which makes sense. Since the data is already random, there should be virtually no effective difference between these pivot methods in avoiding worst-case partitioning. Additionally, in every case but one, the worst MergeSort ends up timing the longest, though this could just be because the 75-min MergeSort is exceptionally slower than the other min-threshold MergeSort's.

For the sorted data though, the MergeSort tends to outperform the QuickSort's. This makes sense, since for the simple pivot the QuickSort is at it's worst case (all of the worst algs for sorted data were simple Pivot QuickSort methods). The median-of-three and random pivot partitioning methods significantly outperformed simple pivot for the sorted data (which is expected–much less likely to be worst-case per partition).

Looking at min recursion, the worst setups across the board for random data typically feature high Insertion sort thresholds (for 65 or 75 random elements, it is better to recurse). However, the sorted data shows a strong preference for these high thresholds, since pre-sorted data is best-case for Insertion sort.

To conclude, the optimum sorting method for sorted data was typically MergeSort (though the advanced pivoting methods helped), and the optimum method for random data was typically QuickSort. Higher Insertion sort thresholds were far better for sorted data than for random data as well. By employing a MergeSort of a random pivot QuickSort with a high Insertion Sort threshold, it is possible to sort both types or data somewhat efficiently.