Lab 2 Report

*Benchmarks*

30 benchmarks were ran to test the performance of the merge sort method and the (deterministic) quick sort method. The benchmarks start with 10 numbers in an array, which is what I used as my abstract data type and implementation, and increase to 50, 100, 500, 1,000, 5,000, 10,000, 50,000, 100,000, 500,000.

The first set of 10 benchmarks are sorted in descending order from the max value (e.g., for 10 numbers the value at index 0 is 10) down to 1. It isn’t until around 1,000 values in the array that it becomes clear that merge sort performs better with larger arrays (merge sort takes ~3x less time than quick sort for 1,000 values). At 50,000 values and above quick sort hits a stack overflow whereas merge sort is still able to complete the sorting.

The second set of 10 benchmarks are randomly generated unique values between 1 and the max number of values in the array (e.g., for an array of 10, the numbers are in random order from 1 to 10). On average quick sort out performed merge sort (around half the time as merge sort) and both sorting methods were able to complete sorting up to 500,000 values.

The third set of 10 benchmarks are sorted in ascending order from the min value (e.g., for 10 numbers the value at index 0 is 1) to the max value, with a 0 in the middle of the array. It isn’t until around 10,000 values and above in the array that quick sort is unable to handle that many values and causes a stack overflow, merge sort however is able to finish sorting all 500,000 values. On average, merge sort out performs quick sort 2 to 3 times faster.