CSCI 2210 - Project 2 - Sorting Algorithms

Reid Bandy
East Tennessee State University
Johnson City, United States
bandyr@etsu.edu

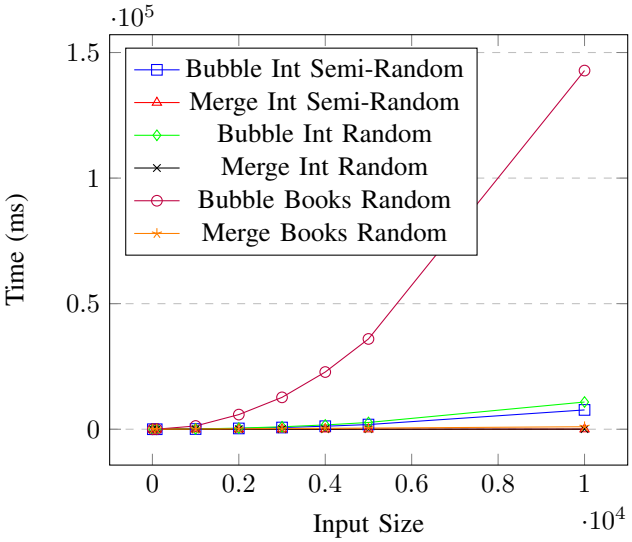

Fig. 1. Comparison of Bubble Sort and Merge Sort

***Abstract*—Testing methodology is not sufficient to make any concrete assertions, but simple implementations of bubble sort and merge sort show that merge sort is immensely superior.**

I. Introduction

This paper compares between Merge Sort, a recursive sorting algorithm, and Bubble Sort, an iterative algorithm. The implementations used in this paper are custom implementations and do not represent industry-standard implementations or libraries.

A. Methodology

Two data types are used: integers and a more complex data type. The complex data type is a "Book" comprised of strings to identify the book, which uses alphabetical ordering.

Both random and semi-random numbers for integers are used when benching these algorithms. Only random is used for books. There is a 10% chance for each number in the unsorted case to be unsorted. This is not to say that they are necessarily not in order, but that they have just not been sorted.

II. Results

As seen in Figure 1, bubble sort is a very slow algorithm compared to merge sort. This is not a surprise as bubble sort is generally an inefficient algorithm and doesn't represent the best iterative methods.

However, this does show the value in recursive algorithms when compared to iterative ones in simple implementations.

Sorting the complex data types is significantly more expensive irrespective of the sorting algorithm. This is not surprising when considering that this is most likely due to increased wait when pulling from memory and more expensive comparisons, but we presume that the cost comes from memory wait.