Thomas Jordan
Dr. Ali
CPSC-350

Analyzing the Performance of Three Sorting Algorithms

In this assignment, Dr. Ali challenged us to create a Merge Sort, a Selection Sort, and our choice of a sorting algorithm. I chose to use the Cocktail Shaker Sort. Initially, I thought that Merge Sort would be better than Selection Sort and Cocktail Shaker Sort, and that Selection and Cocktail Shaker Sort would be similarly efficient. The results were much more drastic than I thought they would be. When I tested the sorting algorithms on 100,000 values, the Cocktail Shaker sort took more than 80% longer than Selection Sort, and Merge Sort was 700% faster than Selection Sort. That difference is huge, and it only took the Merge Sort 21ms to sort 100,000 values. There are several tradeoffs when choosing which algorithms to use. Namely, Space and Time complexity. Even though a Merge Sort has O(n log n) runtime, the space complexity is O(N). The other sorting algorithms have slower time complexities (N^2), and their space complexity is O(1). I used C++ to run each of these sorting algorithms. C++ has the advantage of being a functional-style programming language, meaning that it is better at mutating the given set being sorted and relies less on copying data. C++ is also Object-Oriented, which means my solution was an Object-Oriented solution. Empirical analysis has several shortcomings. The most glaring issue is that the setting they were run in may not be identical. It isn't a perfectly exact environment where each is run. There's also the issue that Empirical Analysis only focuses on time and duration and ignores space complexity.