Assignment 6: Sorting

Cindy Ramirez

December 2019

1 Introduction

I was tasked with implementing three sorting algorithms - Quick Sort, Insertion, Bubble Sort - and another sorting algorithm of my choosing - Selection Sort and Merge Sort in C++. Therefore, we discuss which algorithms are "asymptotically better" than others using a text file to sort double values .

2 Sorting Analysis

The time differences were definitely dramatic, but these differences are sensible considering the differing algorithmic time complexities of the sorting algorithms involved. The trade-offs between the algorithms obviously involve these differing run-times. For larger cases, certain algorithms are completely not feasible, such as bubble sort with its $O(n^2)$ run-time.

Faster algorithms such as Quick sort and Merge sort also raise concerns due to their recursive nature. Recursive functions actually continue to add function calls to the stack without bound, which can unexpectedly cause crashes when the data set is too large, even if the data set could be processed in a reasonable amount of time given enough stack space.

Our choice of C++ as the programming language probably favored the Insertion, Bubble, and Selection sort algorithms because compiled languages like C++ do lots of loop optimization during the compilation process. Interpreted languages like Python do not perform such optimization and would perform even worse for these sorting algorithms, especially those with nested loops. The amount of comparisons and swaps along with the environment are key determinants of performance. In Python, it is recommended to stick with the build in Python sort function for their flexibility on the input and speed. The shortcomings of this empirical analysis are that some of the algorithms tested behave differently when the algorithms are given different data. These algorithms include Quick sort, Insertion sort, and Selection sort, each of which have differing time complexities and worst/best cases depending on the data given. This is important because certain real-world conditions often lead to the worst-case time complexity for some sorting algorithms such as Quick sort which has a time complexity of $O(n^2)$ when the sorting algorithm is used on a sorted list.