Assignment 6: Sorting

Koby Yoshida
1023362
koyoshida@chapman.edu

December 2019

1 Introduction

I created a program that implements bubbleSort, insertionSort, quickSort and selectionSort. I ran a text file with 100 random integers.

2 Algorithms

BubbleSort I was not surprised with the outcome of bubbleSort. It may not be the most efficient method of sorting, however it was very easy to implement. However i was surprised to see that selectionSort was very close in comparison to the speed of bubbleSort. After this result, I changed the data to have more integers to sort, and bubbleSort returned to its low efficiency as the obviously slowest algorithm. This concluded that the larger the amount of data, the slower bubbleSort will run. Although it has a slower runtime compared to other algorithms, it is more memory efficient than faster algorithms.

QuickSort Going in, i knew QuickSort would be the fastest running sorting algorithm we were going to try. I was still surprised to see that it was more than twice as fast as the runner-up insertionSort. QuickSort is definitely the best method to use for larger amounts of data if you are not concerned with memory. Of all the sorting algorithms in this program, QuickSort was the only one that has a run time of O(nlog(n)). The cost of having this fast of a runtime is that it uses a lot of memory space.

SelectionSort I was surprised to see that selectionSort was a slower sorting algorithm for smaller amounts of data. SelectionSort barely passed bubbleSort in terms of speed and was a middle of the pack algorithm for speed. vs. other constraints. SelectionSort would be a good option for a system with limited memory, and time being a somewhat limiting constraint. SelectionSort is fairly balanced between the two.

InsertionSort With 100 elements of data, insertionSort came out as one of the best options in terms of speed. This algorithm would a great choice for a system with a somewhat limited amount of memory space. This may not be quite as fast as QuickSort, but it also consumes less memory than quickSort. I would likely choose insertionSort if I needed a balanced sorting algorithm.

3 Conclusion

In conclusion, there are many different sorting algorithms for different situations a system might run in to. It all ends up coming down to memory vs. speed. C++ is a faster language for sorting algorithms than other languages because it is a compiled language that works well with Windows, which is the operating system i use. Other languages would likely result in slower run times. Also, the computers we are using to sort these numbers have lots of memory and processing power which also results in faster run times. This is the main problem with empirical analysis. We are only able to test on our machines and not across different platforms.