Yixing Zheng
2293298
zheng129@mail.chapman.edu
CPSC 350
Project 6

Assignment 6 Report

After operating and timing all three algorithms, the time differences among all three sorting algorithms certainly surprised me. The results from slowest to fastest are as followed: insertion sort, 670.72 ms; quick sort, 6.10 ms; and heap sort, 0.002 ms. It is very clear that there is a tremendous gap between each sorting algorithms, and that the differences between quick sort and heap sort are certainly more drastic than I expected. I had to modified the printing method for heap sort so it shows three digits of time spend instead of two, otherwise the time spent is too small and rounded to 0.00 ms. As for now, I think the most significant factor that is involved in picking one algorithm over another is the difficulty to implement it. For example, for a relatively not very large number, insertion sort can work just fine and it is relatively easy to implement, thus it is unnecessary to implement a more complex algorithm, such as heap sort. However, when the amount of data becomes larger, then it is crucial to pick up an efficient algorithm, heap sort, in this case, to sort the data efficiently. Furthermore, now it is difficult for me to conclude that whether my choice of programming language would affect the result due to the fact that I do not have a comparison on the runtime between different languages. I have implemented quick sort and bubble sort years ago in Java, but I did not count the runtime, nor having the date right to compare to the result obtained in C++. However, I would predict that the algorithms would run faster on C++ due to the manually memory allocating feature it has. Finally, I would argue that the most significant shortcoming of empirical analysis is that it certainly requires many conditions, such as time and equipment, in order to obtain a result. Personally, I would utilize this method only when it is for verifying or gaining a very accurate result.