CPSC 350 Data Structures, Assignment 6 / Sorting Algorithms

Hank Moss

Abstract— **We have learned several sorting algorithms, as well as which algorithms are asymptotically better than others. But though it is important to know that, on paper, a O(n^2) algorithm runs slower than a O(n lg n) algorithm, it is also important to realize just how drastically these algorithmic choices can effect your programs performance at run time. To drive this point home, the (simple) assignment for this week is to implement 3 sorting algorithms and time them on large input.**

I. REPORT ON EXPERIENCES

This assignment made it clear to me how important sorting algorithms, and algorithms of any type, are in the coding world. In class projects it is easy to forget the importance of efficiency at all times, as our homework assignments do not normally involve such large amounts of data. The differences of speed between all the sorting algorithms was much more drastic than I would have expected. I was most impressed by QuickSort and how efficient it still ran with a large amount of data.

Some of the tradeoffs of the different algorithms is that those which are more efficient, like QuickSort, are normally more complex and harder to code. So, if a computer scientest was alone on a desert island with his computer but no wifi, bubble sort still gets the job done (eventually) and is very simple to code. I do not exactly understand how the choice of the programming language would effect these results, but perhaps in some languages, one can be limited to a simpler algorithm or cannot write these algorithms as efficiently. The shortcomings of this empirical analysis was that everyime I wanted to test the algorithms, I had to wait to see what the results were after running the program. If I were to do this on an old computer, I could have spent a lot of valuable time waiting for the output. And because I have to wait for it, I am less likely to choose an extremely large data sample to use.

II. CONCLUSIONS

I am thoroughly surprised at how quick QuickSort stays, even with a large data set. I also knew ahead of time how inefficient BubbleSort was, but now I can clearly see how fast it runs from examples. It was also interesting to learn of new sorts when looking up more such as shellSort, heapSort, and cocktailSort (what I implemented). I was curious to implement cocktailSort, which some call a more efficient bubbleSort, and see how it ran too.

TABLE I
TIME TO SORT (SECONDS)

Elements Sorted	Bubble	Insert	QuickSort	Cocktail
1000	0.005	0.002	0.001	0.002
100000	39.255	12.417	0.020	30.023
500000	891.734	333.333	0.313	793.350

REFERENCES

[1] $https://www.youtube.com/watch?v = COk73cpQbFQ$
[2] $https://www.youtube.com/watch?v = Xmx_6YRBaq8$
[3] $https://www.youtube.com/watch?v = kPRA0W1kECg$