Alejandra Sicairos
CSE13s
10/19/21

Assignment 3 - Sorting: Putting your Affairs in Order

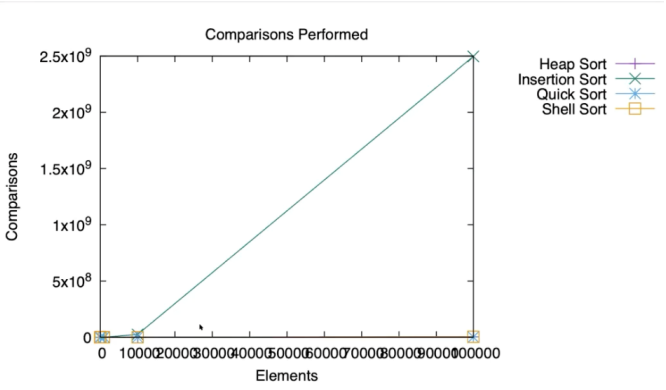
A collection of comparison-based sorting algorithms.

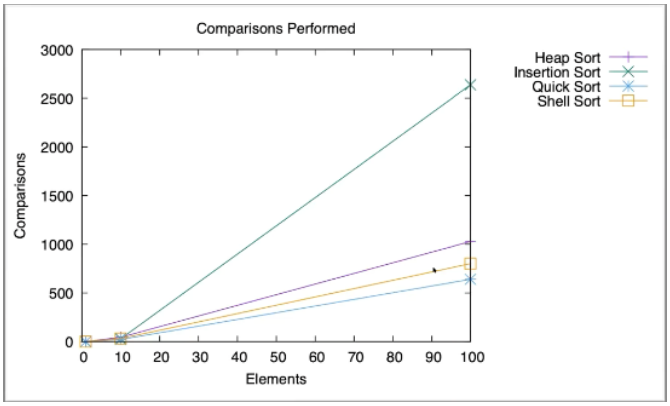
Introduction:

In this assignment, we were asked to implement different sorting algorithms to see the difference between how fast certain algorithms when implemented could be. It is shown by adding different amounts of elements. When doing so we were able to see which sorting algorithm really stood up to its competitors. The four sorting algorithms implemented in this program were; Insertion Sort, Shell Sort, Quick Sort, and Heap Sort. I had difficultly implementing all these algorithms except insertion sort. My main difficulty was actually getting my thoughts and changing the pseudocode to work with the C compiler. This held me back the most. I understood the concept of all the algorithms and how exactly the sort. After some help from tutors, I was able to get the code I am turning in today. Unfortunately my Shell Sorts' algorithm isnt quite correct. I tends to irritate more than it is needed. I wont pass the pipeline at this very moment. I believe it has to do with my inner loop. I tried to do my own debugging at it looks like im not getting the correct gap at the right time, the more I messed with it the more it wouldnt work. So i decided to leave it as it is because it does sort. Revision: I got it to match the output on the resources but still fails on the pipeline.

Materials and Methods:

I wrote this program on a MacBook Pro with an Intel processor. I also used the pseudocode provided in the assignment document. I also push all my code and make sure that is it compatible with Ubuntu using a Virtual Machine. As well as Google Spreadsheets for the graphs, and some graphs from the lecture. I, unfortunately, did not use the GNU plot once again because I still was not able to figure it out on such short notice, I was running late because of the issues in my code that I needed to debug and fix.





Results:

Discussion:

In this graph provided in the lecture, we are able to see the four different sorts compared to one another. The factors that they are being compared by are the number of elements with the number of times that they compare. With this graph specifically, we can see just how fast insertion sort starts to fail. This is because of the algorithm it uses, it goes through every element individually and compares with the previous, and this method is $O(n^2)$ which is a slow way of sorting. Taking into consideration that Quick Sort on the graph is very much the best type of algorithm to use with comparing bigger elements. This is because it is $O(log(n))$

which is very much fast and efficient, but as you can see with it is a smaller number of elements it does not matter what sort is being used.

Conclusion:

By programming this program I was able to really see the O notation in the sense of algorithms. It opened my eyes on how having a proper algorithm is key. First analysing what is to be used for that algorithm, because as shown in the graphs there is no point inreally running Quick Sort or any complex algorithms when doing such small numbers. The importance of the algorithm really is dependent on the number of things you are doing exactly.