Algorithms Are Everything

dlee181

February 2023

1 Introduction

Hello, this paper will go over what it was like implementing the sorting algorithms of Shell sort, Heap sort, Batcher sort, and Quick sort. Each algorithm takes $\mathcal{O}(n \log n)$ except Shell sort which takes $\mathcal{O}(n^{5/3})$.

2 The idea behind every Sort

Shell Sort
Shell sort is relatively simple, it is essentially an insertion sort that instead of taking the values adjacent to each other in an array, will do it in gaps(which are based on the Pratt sequence in our case) which are more efficient because they allow us to move things on the further ends closer to their proper position instead on having to travel long distances across the array to move very far apart objects.
Heap Sort
Heap sort implements the data into trees which follow the heap data structure. This data structure consists of a parent and two children. In a heap, you can choose between a max heap in which the largest value will always be the root. Or you can choose a min heap in which the smallest value will be the root. Ultimately the result is the same, but this time we implemented a max heap. In the heap data structure, I learned that the heap must follow the structure of building a heap and fixing a heap constantly.
Next, after we have established these structures, everything will ultimately end up in a list that we made ourselves, randomly generated. The array will be structured in a way where the parent will be x, the left child will be 2k and the right child will be 2k+1. By doing so it is very easy to arrange the list in a way that it can be sorted into a properly sorted list. Lastly, because we sorted the max heap in such a way, the max value will always be the first value which is something to keep in mind.
Quick Sort
Quick sort is actually the fastest on average compared to the rest of the $\mathcal{O}(n^{5/3})$ functions. Why you might ask? Well, the reason is that although all of these functions (excluding shell sort) have the same time complexity, their difference

is constant. For example, on the graphs page, I have provided two graphs(credit to Darrel Long for his graphs and his explanation which I am using). Although the functions are both the same(time complexity wise, one has a constant that is 10 times the other one and thus takes much longer before it can reach the same efficiency as the other one.

Batcher Sort

I don't really quite understand Batcher sort but here is what I kind of understand. It is essentially a collection of comparison switches used in parallel computations. It is really good for CPU chips and some other near hardware-related things.

3 Explanation of the results

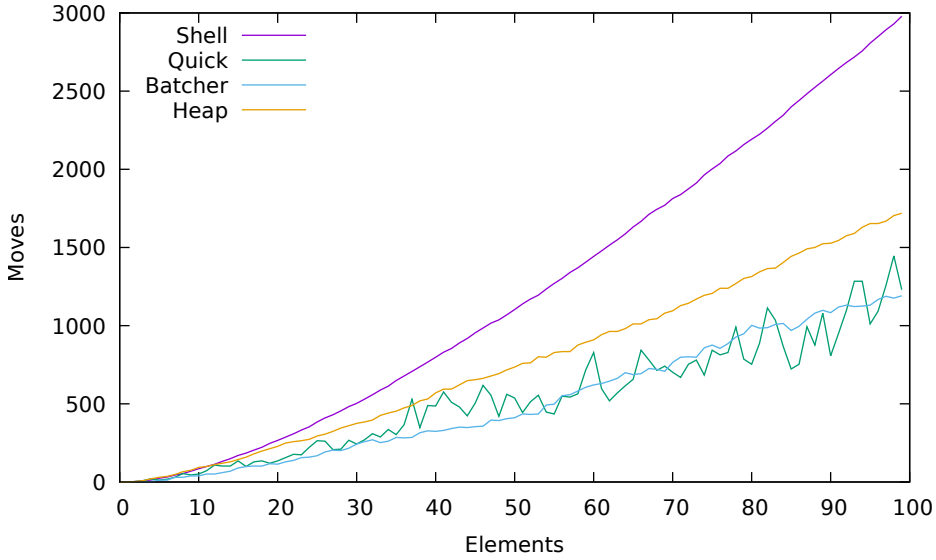
Lastly, I will go over what I learned about each sorting algorithm as a whole, and also I will analyze and compare the performances of each graph with some relationship to the graph below.

I learned a lot in this assignment, starting off with time complexities and efficiencies. As Dr.Long stated, algorithms are everything I truly learned the meaning of that this time. Given a billion things, a computer can compute that in what I believe is either 30 or 38 iterations. That is insane. That's why hardware has been "supposedly improving" a double the efficiency every year. That statement is in fact a lie and the only way to truly speed things up is to optimize the software behind those computers.

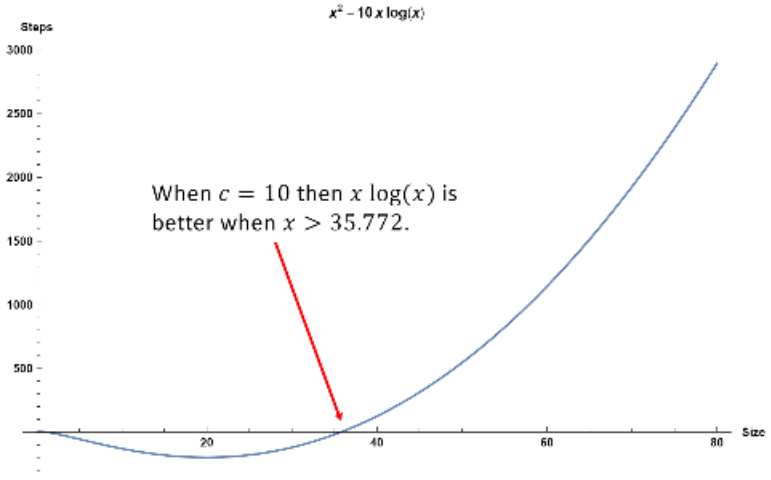
I also learned that quick sorting is essentially the way to go. Quick sort is on average the fastest sort due to its low constant and in the worse case, it is just as fast as insertion sort.

Lastly, when analyzing the graphs we can see that there are four lines representing the moves getting larger(as a whole) as elements get larger. We can see that shell is by far the slowest followed by heap, and although batcher and quick are blurred, the regression line is lower than that of quick by a bit. We also see that the line is very jagged in the quick function and that is due to the median changing when it is partitioned in half every time. I wanted to show the difference in jaggedness because when the values are extremely high, it is almost impossible to see the difference. Also we see that the path of quick, batcher, and heap all have about the same slope, whereas shell goes off and increases at a much faster pace compared to the rest and that is due to their algorithms with the bottom three having the same $\mathcal{O}(n \log n)$ time and shell having a $\mathcal{O}(n^{5/3})$ time.

4 Graphs!



Consider $x^2 - 10\,x\,\log(x)$



Consider $x^2 - 100\ x\ \log(x)$