Shweta Jones
CSE 13S - Winter 2021
Prof Darrell Long

Assignment 3 Writeup - Sorting: Putting Your Affairs in Order

Description:

This assignment focuses on writing up 4 different sorting algorithms: Bubble Sort, Shell Sort, Quick Sort implemented using stacks, and Quick Sort implemented with queues. This allows students to familiarize themselves with different sorting algorithms and what works best when. The main program is run with sorting.c file which is run on terminal.

Bubble Sort Time Complexity:

Bubble sort has the time complexity of $O(n^2)$ which is the worst time complexity compared to the other 3 sorts. This time complexity is consistent for both the best and worst cases. Since bubble sort is the least efficient out of all the sort since it is so repetitive, it's probably the worst to use in reality but the easiest to understand as well.

Shell Sort Time Complexity:

Shell sort generally has the time complexity of O(nlog(n)), which is the same as it's best time complexity. Shell sort's efficiency and time complexity is dependent on the increments, or in our case the gaps within the gaps.h file. The way that shell sort works is that it is ordered based on the intervals that it received. The intervals get smaller as you go through the list.

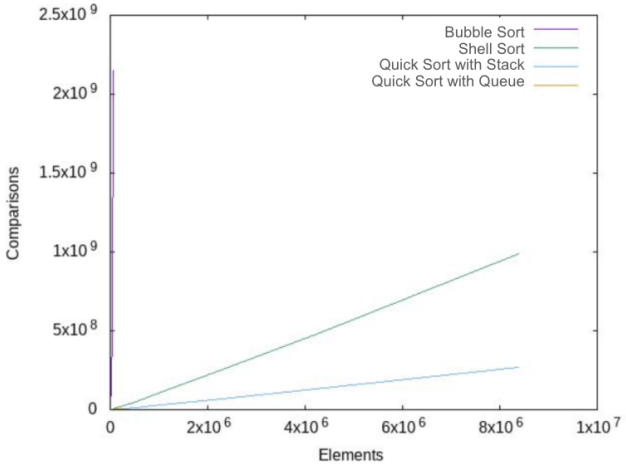
Quick Sort w/ Stacks Time Complexity:

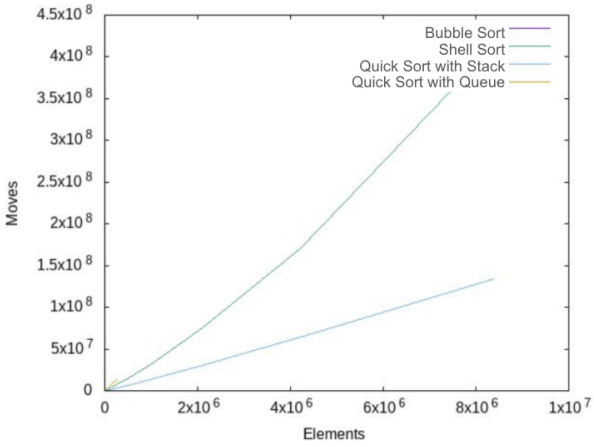
Quick sort is known as one of the most efficient sorting algorithms and has a time complexity of o(nlog(n)). Quick sort breaks up its set and sorts in part before a whole, unlike bubble sort. For quick sort a base case scenario is an array where the partitions are balanced, and so that makes the worst case scenario an array where there is no balance at all where the time complexity is $O(n^2)$.

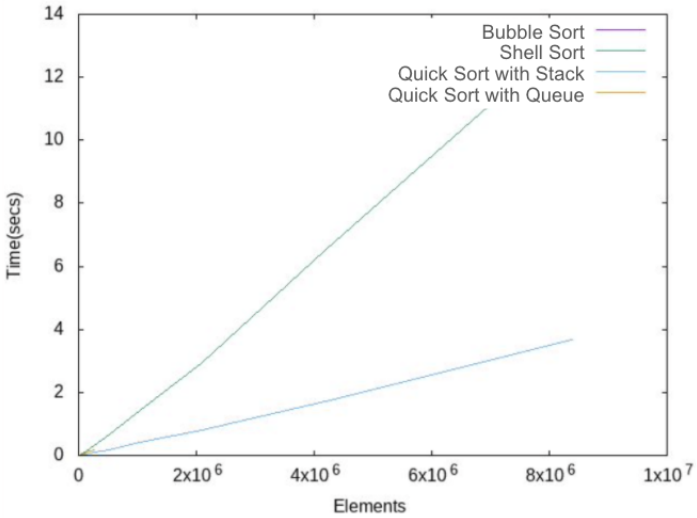
Quick Sort w/ Queues Time Complexity:

The time complexity of quick sort with queue and stack are the same. The number of compares and moves are the same in both implementations of quick sort. Quick sort breaks up its set and sorts in part before a whole, unlike bubble sort. For quick sort a base case scenario is an array where the partitions are balanced, and so that makes the worst case scenario an array where there is no balance at all where the time complexity is $O(n^2)$.

Graphs:







Graph Analysis:

The first graph shows the correlation between the number of elements and number of comparisons for all 4 sorting algorithms. The second plot displays the relationship between number of elements and moves for the different sorting algorithms. And the last plot displays the correlation between the number of elements and the amount of it takes to sort an array in order of smallest to largest. Since the plot for quick sort with stacks and quick sort with queues are the same since the sorting algorithm in theory is the same, just the structure that is used is different, hence why you can only see one of the 2 quick sort plots. Based on looking at these graphs you can see that quicksort is the most efficient when compared to the other sorts: bubble sort and shell sort.

Lessons Learned:

Prior to this assignment my knowledge of sorting algorithms was limited to bubble sort, however, through this lab I was able to learn different types of sorting algorithms and the effectiveness of using these sorting algorithms for more efficient programs. Knowing more about these different sorting algorithms and their corresponding time complexities can allow me to implement them in future assignments, projects and even my job in the future.