Assignment 4 Writeup

Shirin Rokni
shrokni

October 21, 2022

1 Question 1

Question: What you learned from the different sorting algorithms. Under what conditions do sorts perform well? Under what conditions do sorts perform poorly? What conclusions can you make from your findings?

Answer:

- A notable observation is that Heap Sort, Shell Sort and Quick Sort have similar amounts of moves as the number of elements increases, while Bubble Sort increases significantly. From this, we can infer that different sorting algorithms perform differently depending on the size of the data, specifically when the elements of an array increases.
- From the various sorting algorithms that I have implemented, I have learned that different algorithms have different run times. For example, if we want to sort a small data set, we would want to run shell sort because it can short the array faster. However, if we want to sort a larger data set, we would want to run quick sort because it is optimal for sorting larger arrays.

2 Question 2

Question: Graphs explaining the performance of the sorts on a variety of inputs, such as arrays in reverse order, arrays with a small number of elements, and arrays with a larger number of elements.

Answer:

- As we can deduce from figure 4, Bubble Sort does fewer moves when it comes to smaller data sizes, even fewer from Quick Sort. This means that Bubble Sort has a better performance than the other sorts and is thus optimal for sorting smaller arrays.
- Thus, while Quick Sort is efficient at sorting large array sizes, it is not as effective in sorting smaller sizes which is where Bubble Sort is the optimal sorting algorithm is the optimal choice. So, while an algorithm may make significantly more moves and be less effective on larger data sizes, it can be the optimal choice for data sets with smaller elements. Thus, algorithms that excel at sorting small data sets like Bubble Sort should be used in tandem with algorithms that excel at sorting large data sets like Quick Sort so that their poor performance at sorting small data sets is addressed.

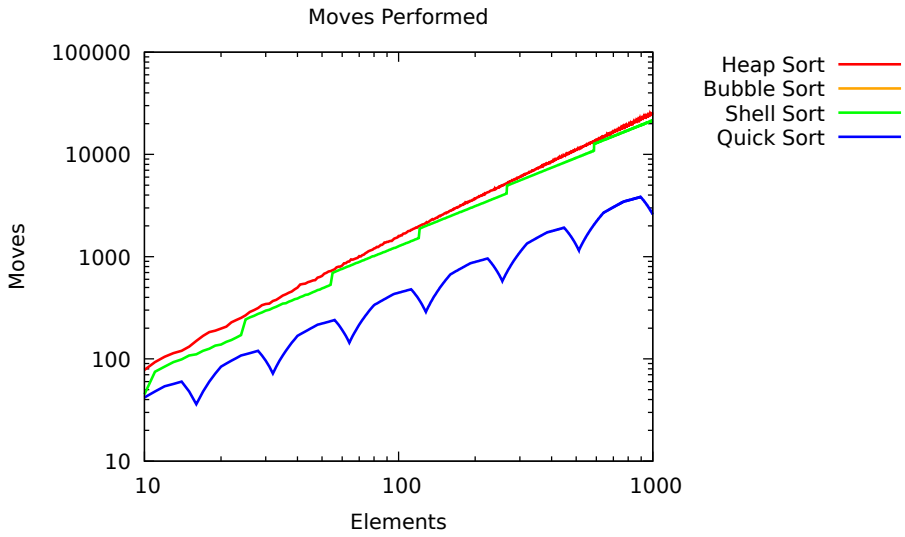

Figure 1: the number of moves each sort performed with an array of [1...1000]. This graph represents the performance of the sorts on an array in order. A notable observation is that Bubble Sort has 0 moves.

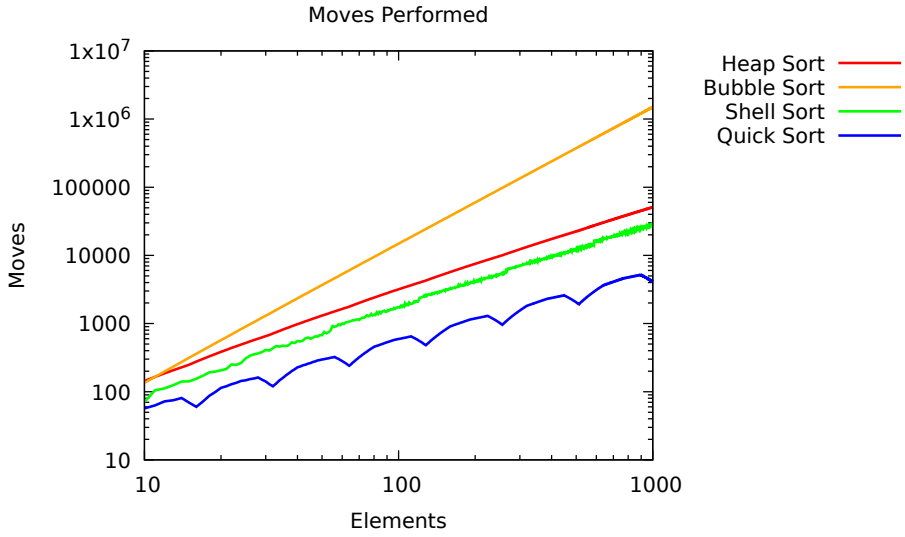

Figure 2: the number of moves each sort performed with an array of [1000...1]. This graph represents the performance of the sorts on an inverse array. A notable observation is that Bubble Sort's moves significantly increased compare to an in-order array while other sorts stay the same.

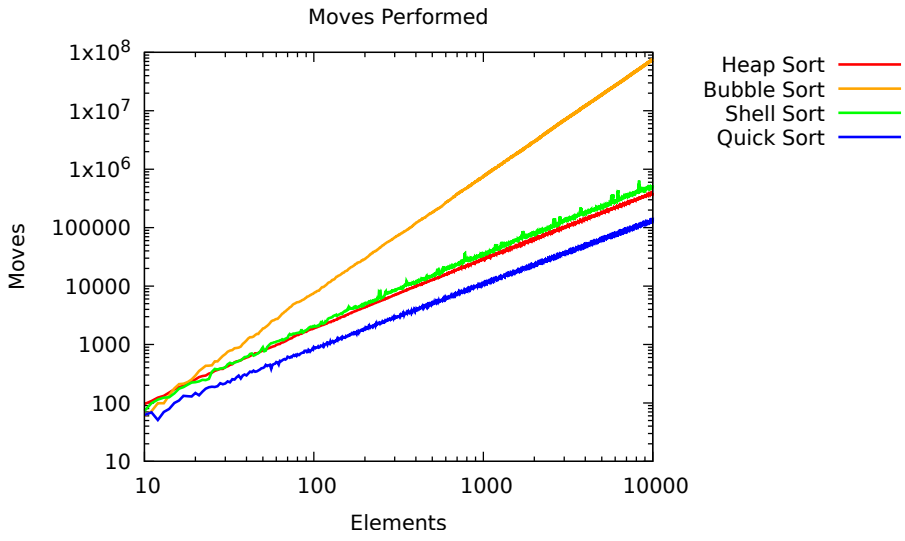

Figure 3: the number of moves each sort performed from 10 to 10000 elements. This graph largely represents the performance of each sort with a large number of elements.

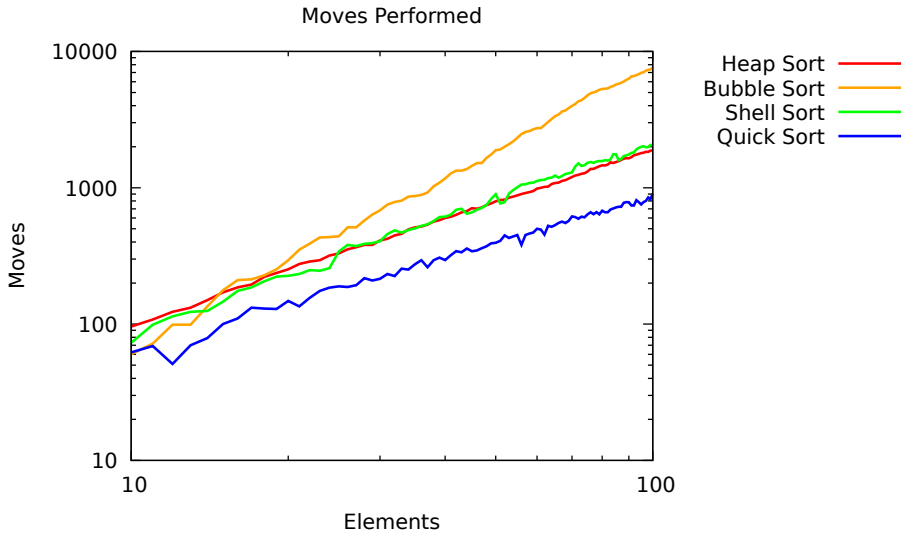

Figure 4: the number of moves each sort performed from 10 to 100 elements. This graph represents the performance of each sort with a small number of elements.

3 Question 3

Question: Analysis of the graphs you produce. You should look carefully at any graphs that you produce. Are all of the lines smooth? What could be causing features that appears in your own graphs?

Answer:

- One feature about all of the produced graphs are that some lines are more straight than others. This is because different sorting algorithms have different levels of stability. For example, Quick Sort is an algorithm that uses the pivot's position to swap the elements, and this results in either having more or fewer moves to sort the array. This means that the algorithm behaves unstably.
- However, Bubble Sort maintains the relative order between elements since elements are only swapped if one is less than the other, and not when they are equal. Thus, we can see with figures 2-4 that Bubble Sort has a moderately straight line and a much better stability level.