Roleen Ferrer 1731261
Professor Maxwell Dunne
CSE 13S - Computer Systems and C Programming

Assignment 5: Sorting Write-Up

For this lab, I was able to correctly implement the four different types of sorting algorithms: Bubble sort, Shell sort, Quick sort, and Binary Insertion sort. I will be using the following example to compare each of the sorting algorithms:

- Number of integers = 500
- Seed = 446944

Bubble Sort

500 elements, 182442 moves, 124750 compares

Shell Sort

500 elements, 8496 moves, 179149 compares

Quick Sort

500 elements, 3177 moves, 7151 compares

Binary Insertion Sort

500 elements, 182442 moves, 3808 compares

Analysis

From ranking each sorting algorithm from the fastest to the slowest, it seems that the order is:

1. Quick Sort
2. Shell Sort
3. Binary Insertion Sort
4. Bubble Sort

Quick Sort

In comparison to the other sorting algorithms, Quick Sort was able to sort the array of 500 integers with less moves and less comparisons. From research, it appears that Quicksort has an average time complexity of O(n log n). These results make sense, as the other sorting algorithms have, on average, worse time complexities in comparison to quick sort. Quite rarely, the time complexity of Quick sort could become O(n^2) because it is possible for the pivot to become either the smallest or largest index, thus creating more comparisons and movement inside the

array. Again, this case is somewhat rare, thus Quick sort would rarely reach the worst time complexity of O(n^2).

Shell Sort

Using the Shell Sort algorithm appears to be faster than Bubble Sort and Binary Insertion Sort, but slower than Quick Sort. Based on research, Shell sort seems to have an average time complexity of O(n log n), but this is generally variable depending on the interval chosen for the gap. It appears that Quick sort has a better time complexity than Shell sort, but changing around the gap function and the algorithm to reduce the amount of comparisons may make the sorting algorithm faster.

Binary Insertion Sort

The Binary Insertion Sort algorithm appears to be slower than Shell Sort and Quick Sort, but nonetheless, is faster than Bubble sort. From research it appears that Binary Insertion sort has a worst case time complexity of O(n log n), but "The algorithm as a whole still has a running time of O(n^2) on average because of the series of swaps required for each insertion." (Source: https://en.wikipedia.org/wiki/Insertion_sort). This makes sense, as Bubble Sort has an average time complexity of O(n^2), but has way more comparisons than Binary Insertion sort.

Bubble Sort

The worst sorting algorithm to sort the array of 500 elements appears to be Bubble Sort, with overall worse comparisons and amount of moves in comparison to the other sorting algorithms. Though, Bubble Sort appears to have the same amount of moves as Binary Insertion sort, most likely due to the algorithm in moving each element from each index of the array being somewhat similar. Bubble sort has more comparisons in comparison to Binary Insertion sort because the algorithm will check each element to check if the current index element is sorted correctly in space. From research, Bubble Sort has an average time complexity of O(n^2).

Conclusion

In conclusion, I can deduce that Quick sort appears to be the more reliable and faster sorting algorithm in comparison to the others. This is due to the fact that Quick sort has a very good average time complexity and would rarely reach its worst case time complexity of O(n^2) due to the nature of it not reaching extreme indexes for the pivot. Testing the runtime of the sorting algorithms in other experiments, I achieve the same results in which sorting algorithm is the fastest vs. which are the slowest. There are ways to improve the other sorting algorithms, such as

Shell Sort by changing the gap size and gap interval, and these changes may have similar results in runtime when comparing it to Quick sort. It seems that using Bubble Sort is very inefficient because of its average time complexity of O(n^2). Overall, when experimenting with these sorting algorithms, I noticed major differences in time complexity and runtime, but these runtimes can be further improved with updates to their algorithm, which would reduce the amount of comparisons and moves.