

## Asgn3 Write Up

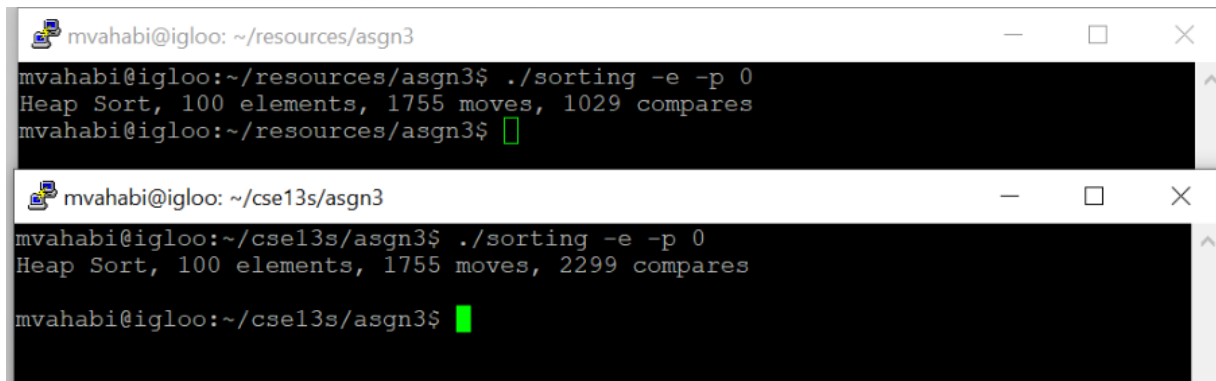
Mahyar Vahabi

### Analysis:

- In this week's project, we learned the different ways of sorting elements in an array, using different methods or procedures. Putting items into a sorted order is one of the most common tasks in Computer Science. As a result, we used the given pseudocodes in the asgn3 pdf to code in C and also understand how it is done; in addition, it pressured us to understand how various sorting algorithms work.

### Errors:

- The output of our program has a subtle difference between the computed comparisons using the heap sorting algorithm.
- It returns the difference between the computed value vs the actual value which is lower than my functions calculations.



```
mvahabi@igloo: ~/resources/asgn3
mvahabi@igloo:~/resources/asgn3$ ./sorting -e -p 0
Heap Sort, 100 elements, 1755 moves, 1029 compares
mvahabi@igloo:~/resources/asgn3$

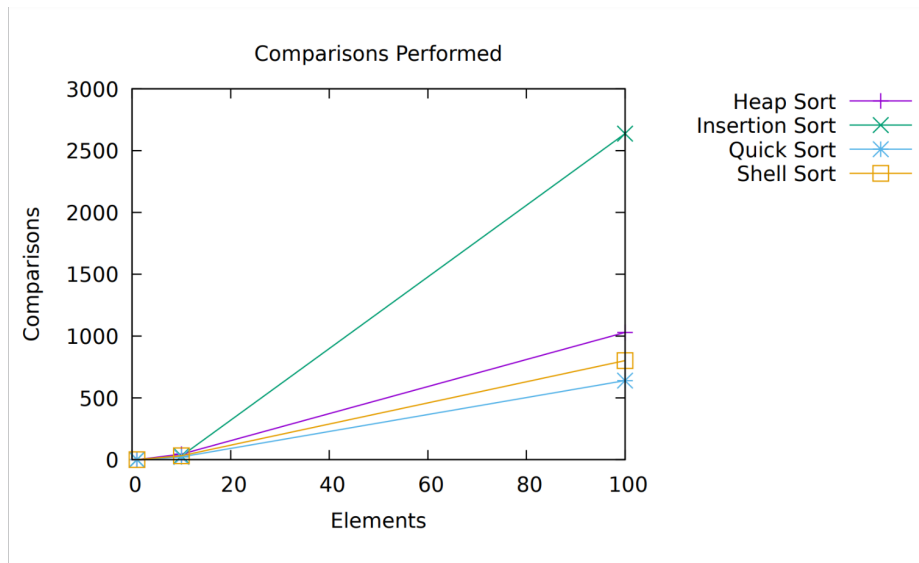
mvahabi@igloo: ~/cse13s/asgn3
mvahabi@igloo:~/cse13s/asgn3$ ./sorting -e -p 0
Heap Sort, 100 elements, 1755 moves, 2299 compares
mvahabi@igloo:~/cse13s/asgn3$
```

### Reasoning:

- In my program, I attempted to shorten the Heap sort file by combining a few of the pseudocode functions into 2 functions only. Even though it sorts the elements correctly and returns the right amount of moves, it gives an incorrect comparison value.

- I believe I accidentally made it work a bit more than the one in the pseudocode, but I knew it would work well enough to do the sorting. My guess is that I made one extra pair of comparisons in my `build_heap` function.

### Data:



- You can observe the difference in the amount of moves it took for each algorithm to sort the elements in the arrays.
- The insertion sort was predicted to be the longest one, because it would individually look at the first element and would go up till the end; once it recognized an element that is smaller than the one before it, it would extract it and place it back however many moves into its corresponding position.
- The quick sort works the best, with less amount of comparisons. Quick sort doesn't require any extra storage and since it works by dividing the array each time to sort the elements into the correct partition, it takes less comparisons.