Wendi Tan
wtan37@ucsc.edu

CSE13s winter 2021
Assignment 5: Sorting

1. Time complexity:

Sorting Algorithms	Best case	Average case	Worst case
Bubble sort	Ω(n)	Θ(n^2)	O(n^2)
Shell sort	Ω(n log(n))	Θ(n log(n))	O(n log(n))
Quick sort	Ω(n log(n))	Θ(n log(n))	O(n^2)
Heap sort	Ω(n log(n))	Θ(n log(n))	O(n log(n))

[The constant of time complexity of shell sort varies with gap sequence.]

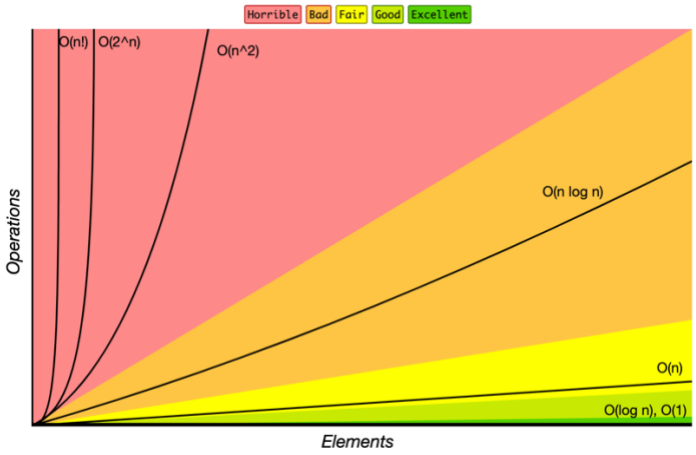
2. Things I've learnt

From these sorting algorithms, I learnt even inside of each algorithm, there are different methods for sorting. For example, there's more than one way finding a pivot. But if we do this by finding the rightmost element as a pivot, in case of an unsorted array, the worst case emerges.

3. How I experimented with the sorts.

I firstly search sorting methods for these sorts online combining with what I learned in the class. Then draw the arrays and implement sorts step by step, visualizing the theories. Then understand the pseudocode professor Long provided to us and translate those into C language.

4. Graphs



Graph cites from
https://www.bigocheatsheet.com/#:~:text=Array%20Sorting%20Algorithms%20%20%20%20Algorithm%20,%20O%20%28n%29%20%2010%20more%20rows%20

Red part represents horrible time complexity, orange is bad, yellow is fair, light green is good, and green is excellent. Combining with the table in question 1,

- For bubble sort, the inverse array would be the worst case, and the graph lies in the red part. As the number of elements grows bigger, the operations increase dramatically.
- For shell sort, the time complexity depends on gap sequence. According to research online, Ciura's sequence might be the most efficient gap sequence for time complexity.
- For quick sort, the worst case would lie on choosing the largest or the smallest element as pivot in each round. But in general, it is the most efficient sorting algorithm. And to decrease the possibility of encountering the worst cases, we choose the midst or random pivot each time.
- For heap sort, time complexity is the same for each case.