Assignment 4 Writeup

Pehara Vidanagamachchi

October 2022

1 Questions

- **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? -** Throughout this assignment, I have been able to familiarize myself with the several different sorting algorithms I encountered. I have been able to learn the different methods of sorting such as bubble sort which iterates through all the elements and swaps them around till they're in order from smallest to biggest. Following that there is heap sort where it searches for the smallest element and orders them from the desired parameters. It iterates through the code from the minimum to the maximum. There is also the quick sort which is also known as the partition-exchange sort is a sorting algorithm that follows a divide-and-conquer principle, it is executed by spitting larger arrays into small arrays and running shell sort if the value is below eight. Finally, the shell sort algorithm works by sorting out elements far apart from each other so it moves elements in front until they are all in the right place, by working between the gaps it is able to organize all the elements. I have found that sorts perform best when given data that puts it in the desired order. Each algorithm, however, has its own set of preferences such as bubble sort which works well with large data sets where they are able to be sorted in a single iteration. So it runs fastest on a small or close to the sorted set. Quick sort on the other hand runs best when the data known is similar and it is able to travel sequentially. While Quick sort is able to run much faster than the other algorithms and works better with arrays. The heap sort algorithm is by far the most inefficient one and is 2-3 times slower than quick sort and is generally seen as an unstable sorting algorithm. And given that all the sorting algorithms run better under different circumstances I was able to conclude that based on the data given some programs will function better than others, and vice versa.

- **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 large number of elements.**

Your graphs must be produced using either gnuplot or matplotlib. You will find it helpful to write a script to handle the plotting. As always, awk will be helpful for parsing the output of your program. - I found that.

- **Analysis of the graphs you produce. Here is an example graph produced by gnuplot with a different set of sorts for reference (axes are log-scaled): -** I do not think.

2 Main Program

- **Bubble Sort -** For this assignment.
- **Heap Sort -** For this assignment.
- **Shell Sort -** For this assignment.
- **Quick Sort -** For this assignment.
- **Sorting -** For this assignment.