Introduction

**Sorting**

Sorting is the arrangement of data (values or items) according to pre-defined ordering rules[[1]](#footnote-1).

A collection of values or items are deemed to be sorted if each item in the collection is less than or equal to its successor. [[2]](#footnote-2)

Sorting is one of the most thoroughly studied algorithms in computer science[[3]](#footnote-3), mainly because it has been claimed that up to 25% of all CPU cycles are spent sorting. In the context of this course sorting data is one of the most fundamental parts of Data Analysis. In order to gain insights into the data and sometimes visualise it in a coherent way the data may require sorting in a specific manner.

When I think of sorting, a list of jumbled numbers is the first thing that comes to my mind, but it is vital to understand that although this project will be using numbers (integers and floats) to demonstrate the different types of sorting algorithms, many other data types such as strings and bytes can also be sorted.

It is a very human trait to put things in some type of order, so normally we have a good concept of basic sorting – however in computing and Data Analysis in particular, sorting the data by hand is often an impossible task, mainly because the volume of data is just so massive. Sorting 10 numbers, even 50-100 numbers manually may sometimes be more efficient, however what happens if you have 10,000 or 10,000,000 values – manual sorting is just not viable.

Since sorting is such a cornerstone of computing, thousands and thousands of hours have gone into studying the most efficient (fastest) ways to sort, while using the least amount of processing power. These two elements are often in contradiction as sometimes speed is the priority, other times salvaging processing power is the greater need – an example of this would be on a production line.

**Example**

If a machine is packing units of a product into a box it may want to capture and sort the serial numbers of each unit. We’ll say it does this sort once the box is full and there is 1000 units per box. If the cycle time on the line is tight (that process is a bottleneck), then the time taking for the sorting algorithm to run would be the priority as you would not want to be waiting for the sorting to happen and adding to the cycle time of the process. However if the previous process was the bottleneck – say it was doing a final test on the units before packing and this was slower than the packing process, then time would not be as significant a factor and you may decide that the processing power is more valuable as the machine interface may also be doing some other work that requires heavy processing power (e.g. using an automated robot).

Because of the different requirements for different scenarios many sorting algorithms have been created over the years. Some focused on time, others focused on processing power and some balanced as best as possible. It is important to know which algorithms suit your specific need. Here I will go through some of the factors that are taken into account when choosing a sorting algorithm.

**Comparison and Comparator Functions**

Above I stated that *“A collection of values or items are deemed to be sorted if each item in the collection is less than or equal to its successor*”. What do I mean by less than here? Well that depends on the data type. Numeric sorting is the easiest to wrap your head around as each number has a predefined set value that is clear to the observer – but what if it is a list of words

**Time and Space complexity**

Time complexity of an algorithm describes the amount of time an algorithm takes to run in terms of the characteristics of the input. This is an approximation of the total number of operations executed in the sort program. It is understood that each operation takes a set time to be performed.[[4]](#footnote-4)

There are a number of notations used for expressing time complexity

* Big O Notation: O – The notation O(n) espresses the upper

Performance

In-place Sorting

Stable Sorting

Non-comparison based sorts

https://www.interviewkickstart.com/learn/time-complexities-of-all-sorting-algorithms

https://realpython.com/sorting-algorithms-python/

1. Mannion, P., n.d. *Computational Thinking with Algorithms – Sorting Algorithms Part 1*. GMIT. [↑](#footnote-ref-1)
2. Mannion, P., n.d. *Computational Thinking with Algorithms – Sorting Algorithms Part 1.*GMIT. [↑](#footnote-ref-2)
3. Python, R., 2022. *Sorting Algorithms in Python – Real Python*. [online] Realpython.com. Available at: <https://realpython.com/sorting-algorithms-python/> [Accessed 4 May 2022]. [↑](#footnote-ref-3)
4. Gupta, S., 2022. *Time and Space Complexities of all Sorting Algorithms*. [online] Interviewkickstart.com. Available at: <https://www.interviewkickstart.com/learn/time-complexities-of-all-sorting-algorithms> [Accessed 4 May 2022]. [↑](#footnote-ref-4)