# CTA Project

# Benchmarking Sorting Algorithms

## Introduction

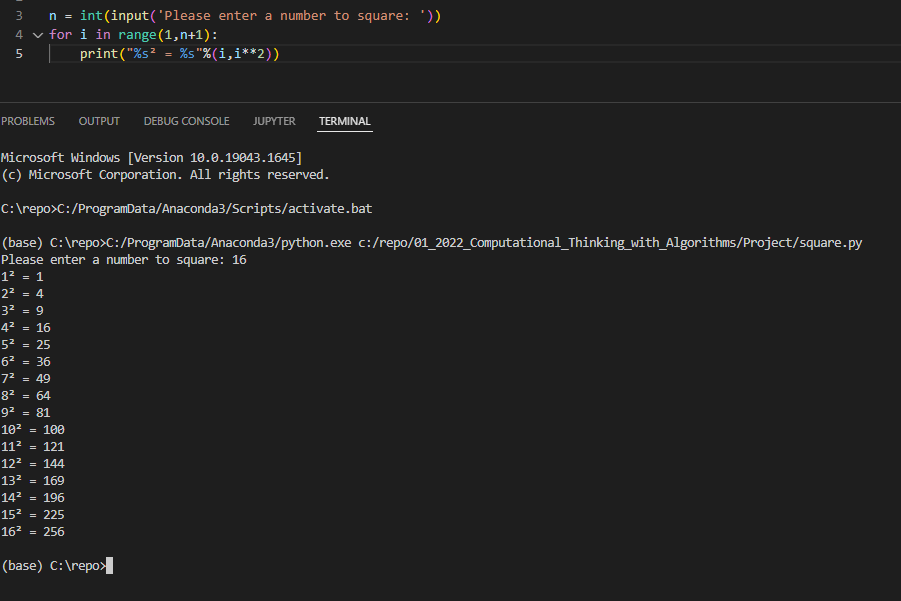
This section of the report is to introduce different concepts of sorting and the use of sorting algorithms. This introduction will hopefully demonstrate an understanding of concepts such as performance, complexity, comparator functions, comparison/non-comparison based sorts, n-place sorting.

This project outcome is to demonstrates a well designed algorithm, as an algorithm is to get an expected output from an input with a set of rules. The request is to investigate and demonstrate this using a sorting algorithms, producing the correct output by demonstrating the efficiency of the sorting algorithm over space/time and varying sizes the input (n=100 to n=10000).

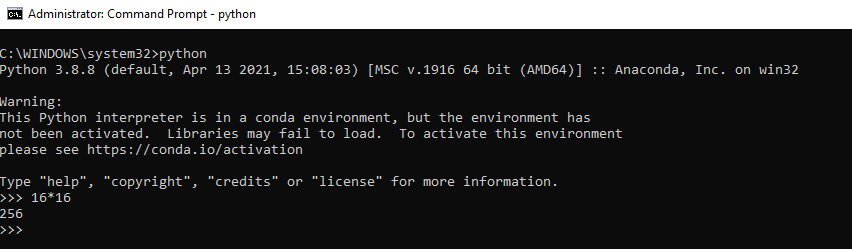
**What is algorithms time complexity?**

The time complexity of an algorithm is estimated by counting the number of steps performed till its final execution. By the number of steps which approach would you consider is the best solution? A python for loop or use a mathematical operator of \*

To demonstrate lets try two methods to output the square of 16. Example 1 uses a for loop to cycle from 1\*1 eventually reaching 16\*16 and printing out each step.

Example 1 For loop *[3]*

The Example 2 uses a Mathematical operator of \*

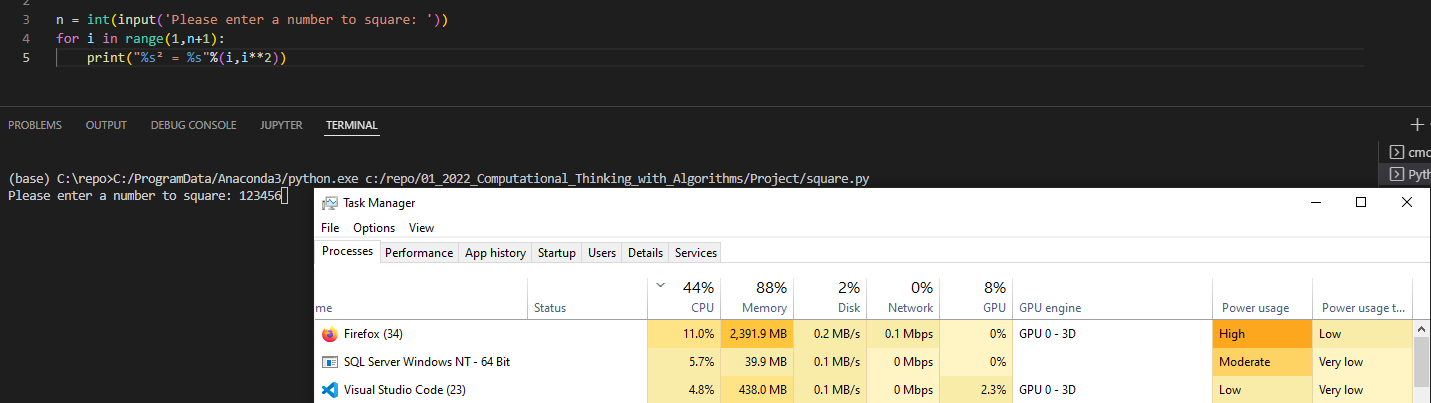
Example 2 Mathematical operator of \*

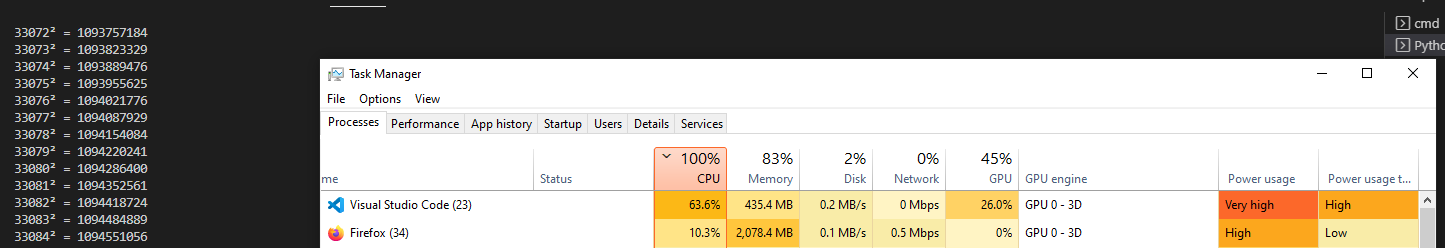
The time complexity of steps performed till its final execution using the mathematical operator of \* returning the result in one line [4] has a better solution.

**What is algorithms space complexity?**

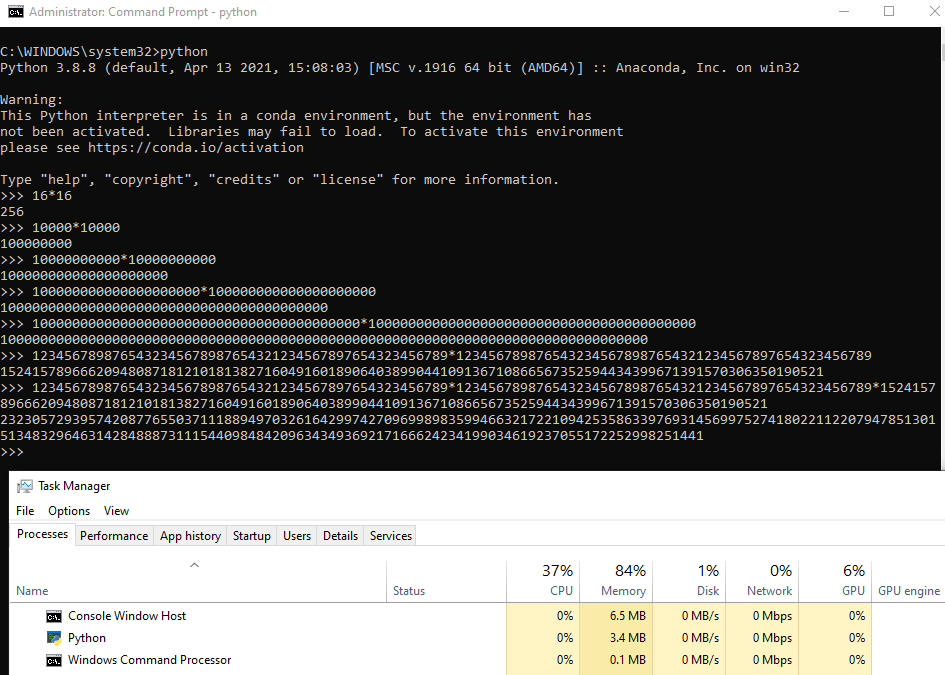
Space complexity is considered the amount of memory a computer requires to solve in relation to the input size, other aspect can also affect the outcome can include by the type of machine, programming language or how it the program is compiled. [1]

So if we re-run the example 3 of the for loop with a bigger number (123456) this will take more space in memory and resources, making the PC unusable until the program finishes executing.



Example 3 for loop Heavy on CPU and memory consumption.

Now lets run this example 4 using the mathematical operator of \*

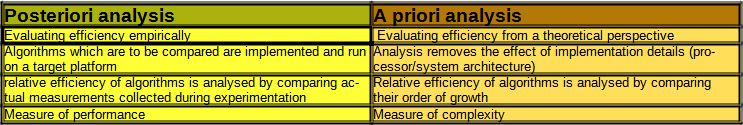
Example 4 negligible CPU and memory consumption

The second squared example of a mathematical operator \* require little to no memory to achieve the same result and even with larger numbers did not reach the same requirements in example 3.

**What methods to measure complexity?**

With an algorithm there are two methods of calculate this with Fig 1 gives a further breakdown.

* A Priori analysis looks at from a theoretical, device agnostic and measure of complexity.
* A Posteriori analysis measure the performance on the same platform

Fig 1 Difference between Posteriori and Priori [1]

These methods indicated performance versus complexity and if we refer to the Example 1 and 3 python for loops, performance never affects complexity but as the complexity increased input size n it affected the performance.

**What is the Big O notation?**

From a theoretical perspective the Big O notation is the most common metric used for calculating time complexity. Referring to example Example 1 and 3 python for loops increasing the size of the input n, increased the number of operation took. Big O notation measures the growth rate of a function increase or decrease in a worst case scenario representing the lower bounds

If Example 2 has a less complex Big 0 notation then Example 1 it can be inferred that Example 2 is more efficient in terms of space/time requirements. [5]

**What is Ω (omega) notation?**

This is basically the best case scenario with a linear growth rate in execution time as n is increased, representing the upper bounds [5]

**What is Θ (theta) notation?**

Θ (theta) notation is the upper and lower bounds of the runtime and is used for analysing the metric used for calculating average time of complexity.[5]

**What is a sorting algorithm?**

A sorting algorithm is to take an input (an array or list) and to arrange the output in a particular order and have a desirable properties of stability, efficiency and in-place sorting.[1] There are several sorting algorithms each with it own efficiency in complexity by the number of operation or time it takes to complete known as Time and the amount of memory required to run also known as Space.

**Why do we need sorting algorithms?**

It is said that 25 percent of the running time of computer in the 1960’s was spent sorting with some tasks responsible for more than half of the computing time. [2]

While us humans deal with sorting information on a day to day for example your shopping list and finding food in a supermarket. We generally use technology to simplify tasks for us, this is usually through sorting, how we search the web and the results and the order of the return results, how to find a car make, model, by year, by mileage, etc.

**What are the different sorting techniques?**

## References

[1] Mannion, P., 2022. *Sorting Algorithms Part 1*

*[2]* *John D. Cook | Applied Mathematics Consulting. 2022. Sorting. [online] Available at: <https://www.johndcook.com/blog/2011/07/04/sorting/>*

*[3]* *square, P. and Parikh, J., 2022. Python for loops program with square. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/41876325/python-for-loops-program-with-square>*

*[4]* *Studytonight.com. 2022. Time Complexity of Algorithms | Studytonight. [online] Available at: <https://www.studytonight.com/data-structures/time-complexity-of-algorithms>*

*[5] Mannion, P., 2022. Sorting Algorithms Part 2*