**Part 1: Sorting and Searching: Algorithm Analysis (70 marks)**

1. Write a Bubble Sort algorithm that sorts the data using a column based on your student number. If two items have the same value sort based on column 1.

You will receive higher marks for optimal (low run-time) solutions. **Highlight in the submission the reason why you chose your sorting algorithm with reference to the run-time complexity.** The sorting algorithm must be your own implementation. You will receive 0 marks for using an imported library to complete this task.

it's a terribly long algorithm in my case, but overall it's useful for programming practice and can be used on a small data.

A screen shot of a computer code

Description automatically generated

in the worst case scenario there will be complexity O(N2)

but in the best case scenario, if all the elements are already sorted, the complexitie will be O(N)

we go through the array and compare pairs of elements each time, if the current one is greater than the next one, we do a swap, even if the collection is sorted, we will check all the elements at least once

Result of sorting

A screenshot of a computer program

Description automatically generated

Bubble Sort of 10000 elements took 45375 milliseconds

1. Experimentally analyse the time complexity of your sorting algorithm that you wrote for question 1 above. **Show your results by taking the average elapsed time for 10, 100, 1000, 5000 and 10000 records.**

better not run my code, haha, its too slow

A computer screen shot of a program

Description automatically generated

1. Write a Quick Sort algorithm that sorts the data using a column based on your student number. If two items have the same value sort based on column 1.

You will receive higher marks for optimal (low run-time) solutions. **Highlight in the submission the reason why you chose your sorting algorithm with reference to the run-time complexity.** The sorting algorithm must be your own implementation. You will receive 0 marks for using an imported library to complete this task.

Worst case is O(N2)

But average case is O(n log n) end best case also O(n log n) which means that each time the array will be divided into two, which, unlike a regular bubble sort, where the algorithm will go through each value at least once, will be executed much faster on a large array, We can see the big difference of time performance in results.

Quick Sort of 10003 elements took 5 milliseconds

Result of Quick Sort algorithm

A screen shot of a computer screen

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

Full results:

Volodymyr Rosinskyi

x23252065