Nottingham Trent University

**SOFT20091 – Software Design & Implementation 2**

**2017/18**

# Lab 5b – Sorting Algorithms

**Consider the following sequence:**

| 50 | 15 | 28 | 1 | 23 | 22 |
|---|---|---|---|---|---|

**1. Selection Sort Algorithm.**

a) Perform a Selection Sort on the sequence above, showing the sequence obtained after each iteration of the algorithm.

Use the following table or similar to keep track of each iteration:

| **Original Array** | 50 | 15 | 28 | 1 | 23 | 22 |
|---|---|---|---|---|---|---|
| **Pass #1** | | | | | | |
| **Pass #2** | | | | | | |
| **(...)** | | | | | | |

b) How many comparisons were made in sorting the array?

**2. Insertion sort.**

a. Perform an Insertion Sort on the sequence above, showing the sequence obtained after each iteration of the algorithm. Keep track of each iteration.

b. How many data moves were made to sort the sequence?

**3. Bubble Sort.**

a. Perform a Bubble Sort on the sequence above, showing the sequence obtained after each iteration of the algorithm. Keep track of each iteration.

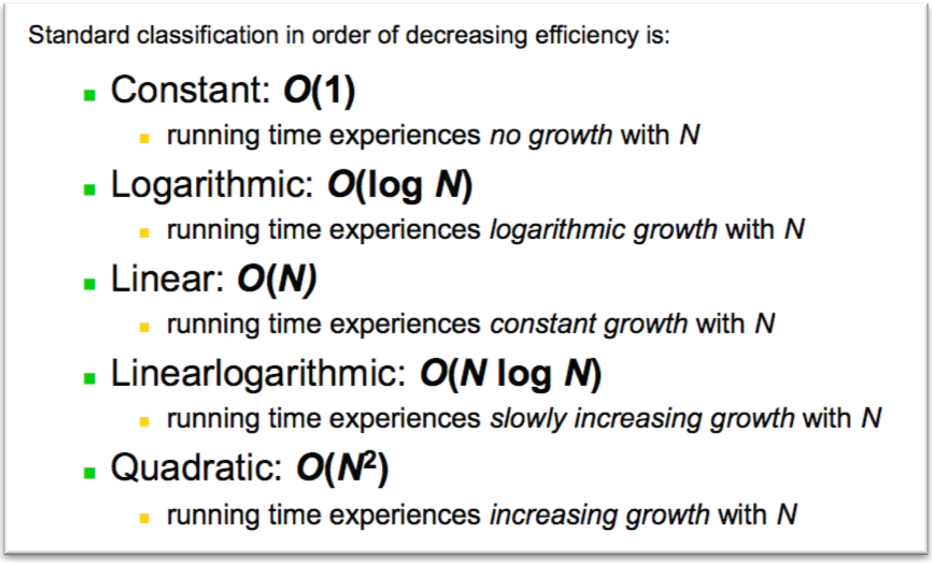
b. How many data moves were made to sort the sequence?

c. Implement in C++ the necessary classes and methods to sort and display the sequence after each pass of the algorithm.

**To study further:**

- See the algorithms in action: https://www.toptal.com/developers/sorting-algorithms/random-initial-order

- Computation Efficiency Analysis:




**Have a go and classify this lab's algorithms for their computational efficiency, the Big-O notation.**

- Sorting Algorithms Computational Speed *vs* Ease of Implementation:

| | Computational Speed | Ease of Implementation |
|---|---|---|
| **Bubble Sort** | Slow | Simple |
| **Selection Sort** | | |
| **Insertion Sort** | | |
| **Shell Sort** | | |
| **Merge Sort** | | |
| **Quick Sort** | Fast | Difficult |

- Have fun while studying SDI: https://visualgo.net/bn/sorting