Practical 1 - Sorting

Updated: 18th July 2023

Aims

- To implement and test Bubble Sort, Insertion Sort and Selection Sort.

Before the Practical

- Read this practical sheet fully before starting.
- Download Sorts.java or DSAsorts.py from Blackboard. Use this as a starting pointto write the sorting methods.
- Download `SortsTestHarness.java` or `SortsTestHarness.py` to test your code.

Note:
Java Students: Please adhere to the `Java Coding Standard` document located under *Links and Resources on Blackboard*.

Python Students: Please adhere to the `PEP8` Coding Standard.

Activities

1. Bubble Sort Implementation

Time to write some code:

- In `Sorts.java`/ DSAsorts.py, implement the `bubbleSort()` method using the pseudocodefrom the lecture slides as a guide.
- Don't forget to include a check to stop `bubbleSort()` if it does not do any swaps during a pass (*i.e.,* The array has finished being sorted).
- Note: The method in the `Sorts.java`/ DSAsorts.py file works on an int[] array.
- Test your code using the `SortsTestHarness.java/py`

```
java SortsTestHarness n xy ...
python3 SortsTestHarness.py n xy ...
    where:
            n is the number of integers to sort
            x is one of:
                    b - bubble sort
                    i - insertion sort
                    s - selection sort
            y is one of:
                    a - 1..n ascending
                    d - 1..n descending
                    r - 1..n in random order
                    n - 1..n nearly sorted (10% moved)
    Hint: You can run multiple test cases by adding additional
    Command Line Arguments.
```

2. Selection Sort Implementation

Do the same for Selection Sort as you did for Bubble Sort.

- Selection Sort differs from Bubble Sort in that it only swaps *once* per pass. Instead, what it does is search for the smallest value, update the *index* of the smallest value until the end of the pass. Only then, does it swap the smallest value with the first value.
  - Remember that in the second pass, the first value has already been sorted. Therefore, don't include the first value in the second pass
  - The same is true for subsequent passes (*i.e.,* The third pass should ignore the first two values.).
- Test your code using the `SortsTestHarness.java/py`.

3. Insertion Sort Implementation

Do the same for Insertion Sort as you did for Bubble Sort and Selection Sort.

- Test your code using the `SortsTestHarness.java/py`.

4. Exploring Sorting Runtimes

`SortsTestHarness.java/py` lets you easily test your sorts code for various types of data. (Refer to Activity 1).

- Create a table of runtime results in a document, using at least four array sizes and at least three of the ascending/descending/random/nearly sorted options across each of theimplemented sorting algorithms.
- Write a paragraph discussing the results in terms of time complexity and other characteristics of the sorting algorithms.

- Hint: Identify the average/best/worst cases and investigate the scalability of the sorting algorithms.

Submission Deliverable

- Allows group of 1/ 2 members.
- Your code, data and document are due 2 weeks from your current tutorial session.
  - You will demonstrate your work to your tutors during that session.
  - If you have completed the practical earlier, you can demonstrate your work during the next session.
  - Every group member needs to be present during the demonstration.
- You must **submit** your code and any test data that you have been using **electronically via Blackboard** under the *Assessments* section before your demonstration. (Individual blackboard submission for each group member)
  - Java students, please do not submit the *.class files

Marking Guide

Your submission will be marked as follows:

- [5] Bubble Sort is implemented properly and tests correctly.
- [5] Selection Sort is implemented properly and tests correctly.
- [5] Insertion Sort is implemented properly and tests correctly.
- [5] Sorts performance investigation.

End of Worksheet