Algorithms and Data Structures (ADS2)

Assessed Exercise 1

This exercise is for submission using Moodle and counts for 10% of the total assessment mark for this course.

This exercise is worth a total of 20 points.

The deadline for submission is Friday 21 February 2025 at 4:30pm.

Exercise

This exercise has two parts. In the first part, you are asked to run an empirical study to compare the performance of several sorting algorithm, and define an efficient algorithm to solve a practical problem in the second part.

Submission

Submit the Java sources of your implementations and a short (maximum 3 pages) report briefly describing what you have done in each part of the exercise. Your report should include a heading stating your full name and matriculation number and clear instructions on how to run your code.

Part 1

Compare empirically the performance of the sorting algorithms you implemented in the Labs:

- INSERTION-SORT
- SELECTION-SORT
- SHELL-SORT
- MERGE-SORT (and variants: hybrid INSERTION-SORT and bottom up)
- QUICKSORT (and variants: Median of 3, Duch flag, hybrid INSERTION-SORT)

Use the datasets provided on Moodle and produce the following:

- A plot comparing all the algorithms in which the running time is on the y-axis and the datasets on the x-axis (from int10.txt to intBig.txt)
- A table with the running times for each algorithm on dataset dutch.txt
- A table with the running times for each algorithm on dataset bad.txt

Perform each experiment 10 times and report only the averages. Briefly explain your findings.

Hints:

- Use `TestSortingAlgorithms` from Lab 1 to test the correctness of your algorithms.
- Set appropriate cut-off times to terminate earlier slow runs. [10]

Part 2

You are part of a team responsible for running a successful video streaming service with millions of views daily. For marketing and research reasons, you have been asked to implement an algorithm that **efficiently** finds the k most viewed videos daily. You should expect $k<<n$, that is the total number of videos n is much bigger than the number of videos you are interested in. Explain in the report how your algorithm operates, describe its running time using big-O notation (including worst case), and implement it in Java. [10]