Programming assignment 3

Getting started

All your submissions should be implemented in Java. If you do not have a Java development environment, we recommend that you either use the Visual Studio Code Java extensions or Jetbrains IntelliJ. If you use IntelliJ, do not forget to register as a student, so you get access to the full versions. If you run Linux, macOS, or WSL, and want more control over your JDK, we recommend SDKMAN!

Note that you must solve all of the programming problems in Java. You are allowed to use Python (or any other suitable tool) to analyse the results, plot graphs, and so on. If you have discussed an alternative language with Morgan, it's fine to ignore the Java requirement and use, e.g., C++ instead.

Problems

Problem 1

Implement a generic (in the Java sense) hash table that uses quadratic probing to handle conflicts. Use strings to test it. Using your own (poor) hash function is probably easier to ensure that conflict handling works as expected.

Your hash table should support operations to `insert`, `find`, and `delete` keys. You do not need to consider growing the table when it becomes full, just make sure to set it to a reasonable size.

Problem 2

Use the hash table from Problem 1 to store information about vehicles. Create a class that contains at least the license plate number, year, color, and make. Your class should implement a `hashCode` function that uses that number to determine which bucket the object should be placed in.

Conduct an experiment to analyze how well your hash function works. You can, for example, study the number of conflicts and the offset.

Problem 3

Implement Insertsort.

Problem 4

Implement heapsort.

Problem 5

Implement Quicksort. Use median of three to determine the pivot value. Your implementation should take a parameter, `depth`, which determines at which recursion depth Quicksort should swap to Heap- or Insertsort.

Conduct an experiment to determine recommended for `depth` and whether Heap- or Insertsort should be used.

Problem 6 (optional, for higher grade)

Implement an iterative version of Mergesort, i.e., a version that does not use recursion. Compare it to a recursive implementation. Which is faster on average.

Problem 7 (optional, for higher grade)

Implement Shellsort and experiement with different Gap sequences. Implement at least 2-3 differen seqences. Use Wikipedia's article as a starting point for various gap sequences.

Submission guidelines

Submit your solutions as a single zip-file via Moodle no later than 17:00 on October 27, 2023 (cutoff 08:00 October 30). This is a group assignment that can be done in groups of one or two students. Your submission should contain well-structured and organized Java code for the problems with a README.txt (or .md) file that describes how to compile and run the Java programs and a report in PDF format that describes the findings from problems 2 and 5. If you solve problem 6 and 7, the report should also contain your experiment setup and findings from that problem. It is sufficient if one person in the group submits the work to Moodle, but make sure that the submissions contains all your names.