Algorithms and Data Structures - Mandatory exercise #2

Hand in date: 25. Nov. 23:59.

The *selection* problem is a frequently used example in the textbook to describe various ways of solving a problem.

The task is quite simple. You have an unsorted list of elements, in our case integers, and you must find the kth smallest element in that list.

The obvious idea is of course to sort the list in ascending order and return the kth element. Using a trivial sorting algorithm, the time complexity would be quadratic, and with one of the more advanced sorting algorithms, you could achieve O(N log N).

There is, however, a way to achieve linear time complexity. That can be done by using the *quickSelect* method, which is shown in section 7.7.6 in the textbook.

Your tasks are as follows:

1. Write a method that solves the selection problem using a priority queue (you must write your own PQ), and conduct a series of experiments that indicate that the time complexity of your method is O(N log N).
2. Implement the *quickSelect* method and conduct a series of experiments that indicate that the time complexity of the method is O(N).
3. Solve the average case recurrence equation for *quickSelect*.

When you conduct your experiments, you should do it on a variety of numbers and a relatively big amount of numbers, i.e. 1000 or more, and you must find a way to 'count' the number of instructions that the methods perform, so that you can relate them to N, thus being able to estimate the time complexity.

Your experiments must be properly documented.

Hand in Details

- Hand in a zip archive with all the source and header files (.cpp and .hpp files) or Java files and any additionally documents required from the assignment.
- Do not include build files and project files from your IDE.
- Name the zip-archive with your student id. example: "fande14.zip".
- **Groups**: Put both student-ids with underscore as delimiter. example: "fande14_fande15.zip". Only one from each group should upload the assignment.
- **CLION/GCC Compiler**: Please note that the GCC compiler used in CLION accepts syntax errors like missing "returns" and attempts to access array indexes that doesn't exists.
- **If you use CLION/GCC:** please be sure to check your code and syntax is in order. Even though the compiler accepts syntax errors please ensure you uphold the minimum C++ standard and don't make syntax errors.
- **TEST-DRIVER:** Remember to make a good test driver with meaningful output.