Select versus Quicksort

Due Date

The report for this project is due **November 10**, 1999. You will also be asked to submit the source code for your programs online (details will be given later).

Objective

After reading the description of the linear-time SELECT algorithm, Professor X made the following statement:

> This algorithm is so complicated that if you were to implement it in real life it would be really slow. Even though the theoretical analysis gives it a $O(n)$ running time, it will probably be faster to just use QUICKSORT.

The purpose of your project is to either support or refute Professor X's statement.

Implementation Issues

For this project you will implement the following algorithms to find the median element in an array.

1. The deterministic linear time SELECT algorithm given in Chapter 10 of the textbook.
2. The randomized selection algorithm given in Chapter 10 of the textbook.
3. Using randomized QUICKSORT to first sort the set of numbers, the picking the middle element of the sorted array.

We fully expect that the randomized selection algorithm will the fastest of the three — it will be used to check the other two algorithms.

In order for this to be a fair comparison, you must make each algorithm as fast as you can. The following is a minimal list of issues that you should address to make your implementation run as fast as possible:

1. The SELECT algorithm in the textbook groups the items into segments of 5 items each. The number 5 was chosen purely for analysis purposes. Segments of $7, 9, 11, \ldots$ would also work. You should run experiments to determine the optimal segment size.
2. The textbook does not describe exactly what happens when you recursively call SELECT to find the median of the medians (Step 3, page 190). You should **not** copy the medians of the segments into a new array. This is unnecessary and wastes a lot of time. Your implementation of SELECT should be flexible enough that you can use it to find the median of an array from one index to another considering only every kth item.

3. For small arrays it will be faster to use Insertion Sort to sort the items. You should run some experiments to find the optimal size to switch over to insertion sort and use this in your running time trials.

Other implementation issues will come up as you write your programs. You must document in your report the steps you have taken to make the implementations as fast as you can.

Data

For this project you must use "real" data. Both QUICKSORT and the randomized selection algorithm will use a pseudo-random number generator on your system. It is not a fair experiment to also use this generator to produce your data. Furthermore, your arrays must be large enough to be meaningful. Your arrays should contain at least tens of thousands of items. Hundreds of thousands is preferable. Your experiments should include several trials for each size.

The internet is a useful source of "random" data. For example, you can download text or images and treat every four bytes as a 32-bit integer. Alternatively you can treat the binary code of an application (e.g., Microsoft Word) as a sequence of integers. Your report should fully document how you obtained the data for testing and give some indication of why you think this is a fair way to test your implementations.

Reporting Requirements

You should report your results as a technical report of roughly 5 to 10 pages. The report counts as 20% of the project grade. The report will be graded on its quality, not its length. A good report should present your case clearly and convincingly. The English will be judged according to the standards of a term paper submitted in a liberal arts course. Grammar counts.

The report should contain the following parts:

Abstract: An abstract is a short description of the contents of the report. The abstract should be written in the third person and should not be longer than half a page.

Introduction: Describe the project, your approach and summarize the conclusions. A person who understands computer science but has not read this project description should understand this section.

Implementation: Document how you implemented the algorithms. Report how the implementation issues described above were addressed.

Experiments: Describe how you generated the data, which experiments you performed, how the running times were collected, etc. Report the timing results of your experiments in tables and graphs. The purpose of this section is to give enough information for the reader to repeat your experiments.

Conclusion: State and justify your conclusions. Between SELECT and QUICKSORT, which algorithm is faster? Do you think your conclusions valid in general or just for the data and systems that you used?

Grading

Your project will be graded on 5 parts weighted equally.

Report: The report will be graded according to presentation, as described above.

Implementation: This portion of the project grade depends on how well you implemented each algorithm. See the implementation issues discussed above.

Data: The quality of the data used in your experiments count for 20% of your grade. If you have doubts about your approach in collecting data, ask your instructor well before the project deadline.

Correctness: The implementation, reporting and experimental methodology should conform to this project description.

Conclusion: You will not be graded on the statement of your conclusion, but on how convincing your conclusions are.

Note that coding is only one part of this project. You will not do well if you simply turned in 3 working programs. You must leave sufficient time after you code to test the code, run your experiments and write up the report.