Exercises and hand-ins

Advanced database technology

February 7, 2005

Consider the following classic internal sorting algorithm Selection Sort. It works as follows: For $i = 1, \ldots, N$ scan the whole input and output the ith smallest element (which, for $i > 1$, is the smallest element larger than the element just output). Some features of this algorithm:

- It does not change the input.
- Uses only memory to store a couple of elements.
- Outputs the sorted list sequentially.

This problem considers the performance of Selection Sort on external memory.

1. Suppose that we have a machine whose internal memory holds only the most recently read block of B elements (plus a few registers). Analyze the I/O complexity of Selection Sort.
2. How does the I/O complexity of Selection Sort change if we have internal memory for M elements, and the virtual memory system always keeps the M/B most recently accessed blocks in internal memory? Argue why your answer is correct.
3. Improve slightly on the above I/O complexity by using another caching strategy for virtual memory. Describe the caching strategy and analyze the I/O complexity when it is used.
4. Devise an external version of Selection Sort having the abovementioned features (i.e., it should not write anything other than the output to disk, but may make use of all internal memory). It should improve the I/O complexity of the internal version investigated in the first three questions by a factor of M.

There will be 5 mandatory hand-ins during the course to be solved individually or in groups of two students. Assignments are given in connection with the lectures (and announced on the home page) and are due at the **start** *of a lecture (normally two weeks later). Every assigment is graded on a scale from A-E (A=very good, B=good, C=acceptable, D=not good enough, E=not handed in/late hand-in), and students must achieve A, B, or C on at least 4 out of 5 assignments. If you are not able to hand in in time, for some reason, inform the teacher as soon as possible. It is allowed to discuss assignments with other students, but the solutions must be prepared in groups of size at most 2. Hand-ins may be in Danish or English, and are to be handed in on paper. More details on how to hand in will be posted on the course home page.*

Note: First hand-in (due February 21) is on the back of the sheet.

The following exercise is to be handed in at the latest February 21.

We have argued that bags can be more efficient that sets. This problem discusses some cases where bags may be less efficient. Assume in the following that a tuple fits in one block. Suppose that we have built a relation by simply adding N tuples one by one, not checking whether an added tuple was already present. Consider a sequence of queries that each require every tuple of the bag to be inspected.

1. Give an example of a bag of tuples for which these queries could take much longer than on the corresponding set of tuples.

2. Show that a bag may be turned into a set by sorting and performing one additional scan.

3. Suppose that whenever the number of I/Os spent on queries on a bag is about to exceed (during the next query) the number of I/Os needed to transform it into a set, we perform the transformation. Show that, no matter how many queries are performed, this strategy uses at most twice the number of I/Os compared to the best of:

   (a) Transforming to a set before performing the queries.

   (b) Using a bag representation for all the queries.