

Exercises and hand-ins

Advanced database technology

February 27, 2003

Hand-in

To be handed in at the latest March 6 at 10.00 AM.

Consider a database that receives a sequence x_1, x_2, \dots, x_N of integers, and has to compute *statistics*, i.e., compute how many occurrences of each integer it has seen. For example, after seeing the sequence (11, 87, 33, 87, 22, 11, 11) it should be able to report the counts (11, 3), (22, 1), (33, 1), (87, 2). The goal of this problem is to investigate the I/O performance of different approaches with respect to:

- The average processing time for the N integers.
- How long it takes to report the counts when the sequence stops.

Use Z to denote the number of distinct integers in x_1, x_2, \dots, x_N . Analyze and compare the I/O performance of the following methods:

1. Initially write all N integers to disk. To compute the counts, sort the integers and scan to do the counting and reporting.
2. Keep a sorted list of counts on the disk. New integers are initially stored in internal memory, but whenever we have seen M new integers (and thus filled memory), or if the sequence has ended, we update the list of counts to include the M new integers (by scanning). Reporting is trivially done by scanning.
3. Insert the integers one by one in a B-tree, where leaves store integers and their current counts. At the end traverse the leaves to report the counts.

Other exercises for discussion on March 6

1. In the lecture of February 27, we discussed *global rebuilding* of a database. The problem of updating the pointers was left open. Assume that an entire database using D blocks of disk space has been rebuilt, except for its pointers. With every record, the previous position of that record is stored. Show how to efficiently update all pointers to old record locations to point to the new locations. What is the I/O complexity of updating the pointers? **Hint:** Use sorting (several times) to pair pointers and positions.
2. GUV 13.3.8.
3. Suppose that two keys K_1 and K_2 are Z keys apart in the sorted order of N keys. Show that a search for K_1 and K_2 can be done in $O(\log_n N + \log_n Z)$ I/Os, by using the B^+ -tree of GUV Exercise 13.3.8.
4. Suppose we want to search among N keys, that internal memory can hold M keys/pointers, and that a disk block can hold n keys/pointers. Further, suppose that the *only way* of accessing disk blocks is by following pointers. Show that a search takes at least $\lceil \log_n(N/M) \rceil$ I/Os in the worst case. **Hint:** Consider the size of the set of blocks that can be accessed in at most t I/Os.