Exercise Sheet 6
complete until Monday, December 5th, 9am

**do ONE of the following two exercises**

**Exercise A**

Read the paper *Optimal aggregation algorithms for middleware*, by Fagin, Lotem, and Naor, Journal of Computer and Systems Sciences 66:614–656, 2003. You find a PDF version in our papers database under http://search.mpi-sb.mpg.de/papers. The paper is quite long, but for the purpose of this exercise you will have to read only parts of it.

Prepare a *blackboard presentation* for the instance-optimality proof of either NRA (Theorem 8.5) or CA (Theorem 8.9). You can concentrate on the main case(s), and leave out special cases.

In your presentation, pay particular attention to get a good understanding of the **main idea/intuition behind the proof**, so that you can explain it to us very clearly. Understand the details, but do not get lost in them.

**Exercise B**

For two or three queries of your choice on your collection from Exercise Sheet 1, *compute a lower bound* on the cost of any top-$k$ algorithm as follows.

For a given query, iterate over all possible combinations of scan depths, e.g., for two query words with lists of sizes 2 and 3, respectively, consider $(0, 1), (0, 2), (0, 3), (1, 0), (1, 1), (1, 2), (1, 3), (2, 0), (2, 1), (2, 2), (2, 3)$. For each of these combinations, compute the essential number of random accesses, which is the number of document ids, for which the bestscore obtained by scanning up to the considered depths lies above the smallest of the final top-$k$ scores. This number times the cost of a random access plus the sum of the depths (which is exactly the number of sorted accesses done if scanning to these depths) times the cost of a sorted access gives a lower bound on any top-$k$ algorithm scanning to this combination of depths (and you should give a short explanation why this is so). Let $C$ be the set of these costs, one for each combination of scan depths. Then $\min C$ is a lower bound on the cost of any top-$k$ algorithm, that is, every algorithm, whatever clever things it does, has a cost of at least this much to compute the correct set of top-$k$ items.

There are a number of parameters here: the $k$, the cost of a random access, the cost of a sorted access, and the queries to look at. Choose them in some reasonable way. Note that the computation above can be quite time-intensive, so beware of queries with very long lists (the longer lists you can do, the better of course).

For each query, compute the ratio between your lower bound and the result that TA would produce on these queries. Compare this to the worst-case ratio of $m + m \cdot (m - 1) \cdot c_R/c_S$ proven by Fagin et al., where $m$ is the number of query words, and $c_R, c_S$ are the cost a random/sorted access, respectively.