Uni Freiburg, Web Science Group
Prof. Peter Fischer
Systems Infrastructure for Data Science - Winter 2014/15

Exercise Sheet #4: Query Processing

Due/discussion: Nov 20, 2014

Exercise 4.1 : External Sorting

A. Assume you have a memory buffer that allows you to store at most three frames at any given time and each page of memory can contain at most two tuples. You wish to sort the following sequence:

```
67, 12, 45, 84, 58, 29, 76, 7, 91, 81, 39, 22, 5, 47, 15, 80, 99, 34, 32, 8
```

Perform external sorting on this input. How many page-reads and writes are required during the sorting operation?

B. Suppose you upgraded to a larger memory system which allows 5 frames to be stored in memory at the same time. How will you optimize the sorting operation in [A] to minimize I/O costs?
Show the runs produced and the merge phases. How many disk seeks are required in this case?

Exercise 4.2 : Nested Loops Joins

Suppose you want to perform a join of two relations R and S, where R is 1,000 pages and S is 100,000 pages in size. Each page contains 20 tuples. The relations are located on a disk with 10ms average access time and 10,000 pages/sec transfer speed. For your calculations, assume that CPU cost is negligible and ignore the cost of join output.

A. Given a block size of 100 pages, how long does it take to perform a Block Nested Loops Join of R and S? Note that you may choose either R or S as the "outer" relation. Explain the the best choice.

B. Suppose there is a memory-resident index on S which matches the join predicate. Assume a join selectivity $\theta = 1\%$, i.e. each tuple in R matches on average 1% of the tuples in S. How long does it take to perform an Index Nested Loops Join (INLJ) of R and S?

Exercise 4.3 : Sort-Merge Join vs. Hash Join

Using the same setup for relations R and S as in the previous exercise, you are now given a buffer which can hold 256 pages.

A. Estimate the I/O cost of joining R and S using a Sort-Merge Join algorithm.

B. Estimate the I/O cost of joining R and S using a Hash Join algorithm. Assume you may partition the relations uniformly.