Institut for Matematik og Datalogi
Syddansk Universitet

October 2, 2009
Peter Schneider-Kamp

# Introduction to Computer Science
# E09 – Week 6

## Lecture: Monday, September 28

Rolf Fagerberg lectured on files, merging, and hashing.

## Lecture: Wednesday, September 30

Rolf Fagerberg lectured more on files, merging, and hashing.

## Lecture: Monday, October 5, 12-14 (U20)

Lene Monrad Favrholdt will give a brief introduction to cryptography and lecture on online algorithms.

## Discussion section: October 6, 12:15-14 (U52)

Note: The fourth exercise below involves programming. It should be done before you come to the discussion section. You may use Java or Maple.

Discuss the following problems in groups of three or four (the page numbers and problem numbers are from the textbook):

1. Sequential files: Problem 54 on page 478.
2. Hashing: Question 6 on page 469.
3. Hashing: Actually, any hash function has inputs which are very bad for it, making the hash storage system be little more than a sequential file (of course, the hope is we don't meet these inputs in practice—with a well-chosen hash-function, this is also the experience in general). However, to prove the point, find a type of input for which the hash function $h(x) = x \bmod 41$ from page 466 sends *all* elements to the same bucket.

4. Hashing: Write a program to compute the probability of having no collisions when hashing is used for inserting $m$ records into $n$ buckets. Make the assumption that the hash function distributes records randomly to buckets. (Hint: See the calculation on page 468 and generalize it.) Use your program to answer problem 7 on page 469 and problem 57 on page 478. Explain how your program works.

5. Merging: Question 1 on page 469.

6. Merging: Assume sets of numbers are represented by sequential files sorted on element value. For example, the set $\{4, 7, 13, 9, 2\}$ is represented by a sequential file containing $< 2, 4, 7, 9, 13 >$.

   Describe algorithms for constructing $A \cup B$ and $(A \cup B) \cup C$ from $A$, $B$ and $C$ (Hint: $A \setminus B$ was done in class). Note that $(A \cup B) \cup C$ can be done by first computing $A \cup B$ and computing the union of this with $C$. Instead of giving this solution, process the three files simultaneously, as you do with two files.

7. Merging: Assume the database relations $A$ and $B$ each are stored as sequential files of tuples, ordered according to attribute $X$ (which is an attribute of both relations).

   Sketch (details not necessary) an algorithm based on merging for executing the statement

$$C \leftarrow \text{JOIN } A \text{ and } B \text{ where } A.X = B.X$$

8. Hashing: Assume again that the database relations $A$ and $B$ each are stored as sequential files, but now no longer ordered on the $X$ attribute.

   Describe an algorithm based on nested loops for executing the statement

$$C \leftarrow \text{JOIN } A \text{ and } B \text{ where } A.X = B.X$$

   How many comparisons between tuples are performed (as a function of $|A|$ and $|B|$, the numbers of tuples in each relations)?

   Describe how to speed up the algorithm by first using hashing on each relation.

## Discussion section: October 8, 14:15-16 (U26)

Discuss the following problems in groups of 3 to 4. (Think about these problems before coming to discussion section.)

1. The List Scheduling algorithm (LS) places each new job on the machine with lowest load (currently). We showed in class that the algorithm has a competitive ratio of $2 - \frac{1}{m}$, where $m$ is the number of machines. (Note the competitive ratio is the worst case ratio, over all possible input sequences, of the value the on-line algorithm achieves on the input sequence to the value the optimal off-line algorithm achieves on the same input sequence.) In order to show that the competitive ratio was this high, we showed that if the algorithm gets many small jobs ($m(m-1)$ of length 1) followed by one large job (a job of length $m$), LS will pack them so each machine has the same number of the small ones, so the last job will be placed on some machine that already has a large load. One might hope that holding a single machine free for such large jobs could help.

   Consider the following algorithm: Machine $M_1$ is kept mostly free for large jobs. The other machines are $M_2, M_3, ..., M_m$. Suppose you are trying to obtain a competitive ratio of $C$. When handling a job $J$, place it on the least loaded of machines $M_2, M_3, ..., M_m$ if placing it there does not bring the competitive ratio above $C$ (if $J$ was the last job). Otherwise, place it on machine $M_1$. Show that this strategy cannot lead to a better competitive ratio than $2 - \frac{1}{m}$.

2. If the number of machines is $m = 2$, then LS is 3/2-competitive and is optimal, i.e., no on-line algorithm has a better competitive ratio.

   Assume now, that one of the two machines is twice as fast as the other. Consider a variant of LS, LS2, which places a job $J$ on the machine where it will finish first. This algorithm is 3/2-competitive.

   (a) Show that LS2 is optimal. (You can do it using a sequence with two jobs.)

   (b) There are also other optimal algorithms for this problem. What other (very simple) algorithm also has competitive ratio 3/2?

3. Consider the dual bin-packing problem, which is the problem of putting (as many as possible) items into $n$ bins, which all have size 1. First-Fit (FF), is the algorithm which puts an item (which arrived on-line) in the first bin in which it fits (no bin may be filled to more than 1). An *accommodating sequence* is a sequence which can fit in the $n$ bins.

   (a) Show that with an accommodating sequence, FF has a competitive ratio between $\frac{1}{2}$ and $\frac{4}{5}$, i.e., with such a sequence, FF packs at least half of the items in the bins, and there exists such a sequence where FF only packs $\frac{4}{5}$ of the items. You can do this using only items of size $\frac{1}{2}$ and $\frac{1}{3}$.

   (b) Can you come further down than 4/5 by using items of size $\frac{1}{5}$, too?

4. In the classical bin-packing problem, you are not given a limit on how many bins there are and must pack all items. The goal is to use as few bins as possible.

   (a) Show that FF has a competitive ratio between $\frac{5}{3}$ and 2. (Hint: you can show that it is at least $\frac{5}{3}$ using a sequence consisting of items of sizes $\frac{1}{7}+\frac{1}{2000}, \frac{1}{3}+\frac{1}{2000}, \frac{1}{2}+\frac{1}{2000}$.)

   (b) Improve the $\frac{5}{3}$ to about 1.69 by using items of sizes $\frac{1}{43}+\frac{1}{8000}, \frac{1}{7}+\frac{1}{8000}, \frac{1}{3}+\frac{1}{8000}, \frac{1}{2}+\frac{1}{8000}$).

## Assignment due 14:15, October 21

Late assignments will not be accepted. Working together is not allowed. You may write this either in English or Danish. Write clearly if you do it by hand. Even better, use LaTeX.

When answering, describe your methods and calculations. If you use a program to compute something, please include the code or a good description of what it does.

1. Assume sets of numbers are represented by arrays sorted on element value. For example, the set $\{4, 7, 13, 9, 2\}$ is represented by an array of length 5 containing $[2, 4, 7, 9, 13]$. Write a program in Java (or Maple) for constructing $A \cap (B \cup C)$. Use an algorithm similar to that in Figure

9.15 (which goes through each list once). As in problem 6 above, process the three arrays simultaneously (i.e., you should not first calculate $B \cup C$ and then intersect with $A$). Test your algorithm. Turn in both the program code (commented) and your test results.

2. If a hash function distributes into 12 buckets, what is the probability that no collisions occur when three records are inserted into the hash table? (Make the assumption that the hash function distributes records randomly to buckets.)

   For a hash function as above, how many records must be stored in the hash table until it is more likely for collisions to occur than not?