Institut for Matematik og Datalogi
Syddansk Universitet

Assignment 4 — Introduction to Computer Science 2015

This is your fourth assignment in DM534/DM558. The assignment is due at **8:15 on Thursday, November 19**. You may write this either in Danish or English. It must be made in LaTeX. Write your full name, your section number (D1, D2, or D3), and your "instruktor"s name (Kristine Vitting Klinkby Knudsen, Mathias W. Svendsen, or Jesper With Mikkelsen) clearly on the first page of your assignment (on the top, if it's not a cover page). You should turn it in as a PDF file via Blackboard through your DM534/DM558 course. The assignment hand-in is in the menu for the course and is called "SDU Assignment". Choose the correct one for your section number, D1, D2 or D3. Keep the receipt it gives you proving that you turned your assignment in on time. Blackboard will not allow you to turn in an assignment late.

Cheating on this assignment is viewed as cheating on an exam. You are allowed to talk about course material with your fellow students, but working together on this assignment is cheating. If you have questions about the assignment, come to Joan Boyar or your "instruktor" for DM534/DM558.

Please note that you must have this assignment approved in order to pass DM534/DM558. If it is not turned in on time, or if you do not get it approved, it will count as one of your two retries in the course, and you must have it approved on your single allowed retry for this assignment. Note that you have only two retries in total for the assignments in DM534.

Assignment 4

Do the following problems and write your solutions in LaTeX. Write clear, complete answers, but not longer than necessary. Do not include the statements of the problems or other information not asked for in the problems.

1. Do problem 15b on page 452 of the textbook. Note that the problem asks for a solution using relational operations, not SQL.

2. Assume a hash table is partitioned into 12 buckets. Assume that the hash function is such that a randomly chosen record is equally likely

to hash to any of the buckets. In these problems, you may use a program to check your answer, but show the intermediate steps of your calculations, so it is clear what is being calculated and why.

(a) What is the probability of at least two of three arbitrary records hashing to the same bucket?

(b) How many records must be stored in the table in order for it to be more likely for at least one collision to occur than for no collisions to occur? Give the smallest possible number for which this holds.

3. Do one of the following two problems. The second is the more challenging.

    (a) Assume sets of numbers are represented by sequential files, sorted on element value. For example, the set $\{14, 27, 13, 9, 32\}$ is represented by a sequential file of length 5 containing $[9, 13, 14, 27, 32]$. Since, by definition, sets do not have duplicate elements, these files do not either. Write a procedure in pseudocode for constructing $A \cap (B \cup C)$, where A, B, and C are sorted sequential files. Use an algorithm similar to that in Figure 9.15 (which goes through each file only once, never reading any element more than once). Or you can use the API from the slides

    http://www.imada.sdu.dk/∼joan/intro/15slide9.pdf

    using `open, close, isEndOfFile, readNext, writeNext`. As in problem 3 (from the discussion section described on the note for week 44), process the three sequential files simultaneously (do not first calculate $B \cup C$ and then the intersection with A). Make sure your algorithm works correctly in all cases, and explain why it does.

    (b) Consider the problem of merging four sorted files, each of length n. Call the files A, B, C, and D. There are two obvious ways to do this. One is to first merge the files A and B, then merge the files C and D, and then merge the two resulting files. The second processes the four files simultaneously (as in the above problem, never reading any element more than once). Analyze both of these algorithms (worst case), calculating the number of reads, the number of writes, and the number of comparisons, separately. Express your results in the form $i \cdot n + j$ in all cases. The value i

should be as low as possible, and both i and j should always be constants, not functions of n. When counting comparisons, you should be able to get the same value for i for both algorithms. This requires being careful in designing the second algorithm. Specify exactly which comparisons are done by that second algorithm (this may be clearest with pseudocode). Explain your answers.

4. Include your LaTeX code for this assignment at the end.