Institut for Matematik og Datalogi
Syddansk Universitet

April 24, 2014
JFB

DM508 – Algorithms and Complexity – 2014
Lecture 1

Textbook and notes

Introduction to Algorithms, 3rd edition, by T. Cormen, C. Leiserson, R. Rivest, and C. Stein, MIT Press, 2009.

Extra notes (available in the bookstore): *DM508 Algoritmer og kompleksitet, Noter 4. kvartal 2014*. This is identical to the notes from 2010, 2011, 2012, and 2013 and is from the following sources:

- *Computer Algorithms: Introduction to Design and Analysis*, second edition, by S. Baase, Addison-Wesley, 1987.
- *Combinatorial Optimization: Algorithms and Complexity*, by C.H. Papadimitriou and K. Steiglitz, Prentice-Hall, 1982.

Format

Lectures (and discussion sections if there are foreign students) will be in English. Please read the appropriate sections in the textbook or notes before coming to class and bring your textbook with you. There will both be assignments which you are required to turn in and other assignments which you should be prepared to discuss in the discussion sections (øvelserne/træningstimerne), usually shortly after the relevant lecture. The lectures will be on Tuesdays in U156, 8:15–10 (weeks 15,17–21), and on Wednesdays in U49, 14:15–16 (weeks 15, 17, 19, 21), but one of these will be cancelled. The discussion sections move around more. Ten are scheduled, but one will be cancelled. The "instruktor" for the course is Christian Kudahl.

The required assignments will be graded on a Pass/Fail basis, and satisfactory completion of all 3 assignments is required for a Pass. The assignments must all be turned in on time using the Blackboard system, submitted via the menu item "SDU Assignment". Turn in each assignment as a single PDF file. Do not use any Danish letters or other non-ASCII symbols in the name of the file. Keep the receipt it gives you proving that you turned your assignment in on time. You may work in groups of 2 to 3 students if you wish. These 3 assignments must be approved in order for you to take the oral exam, so cheating on these assignments is viewed as cheating on an exam. You are allowed to talk about course material with your fellow students, but working together on assignments with students not in your group is cheating. (You can, however, talk with Christian or me.) Using solutions you find elsewhere, such as on the Internet, is also cheating. You may do the assignments

in either English or Danish, but if you write them by hand, please do so very neatly. You will be allowed to redo one of the first two assignments if it is not approved the first time (if one of your assignments is late, then you will have used up your only chance to redo any assignment).

The weekly notes and other information about the course are available through the World-wide Web. Use Blackboard or the URL:

http://www.imada.sdu.dk/∼joan/dm508/

I have office hours 9:00–9:45 on Mondays and 8:30–9:15 on Thursdays.

There will be an oral exam on June 16, 17 and 18, 2014 (probably only 2 of those 3 days). Previous exam questions are available on DM508's homepage; an updated set of exam questions will be available later in the course (probably the same as last year). You may do your exam in Danish if you wish (in most cases it is advisable to do it in Danish).

Lecture, April 8

We will begin with an introduction to the course. Lower bounds from section 2.4 of the first part of the notes will be discussed (part of this is also in section 8.1 of the textbook). We will also begin on sections 3.1 and 3.2 of those notes.

Lecture, April 9

Magnus Gausdal Find will lecture, finishing sections 3.2, 3.3 and 3.5 of the DM508 notes, plus median finding from chapter 9 (section 9.3) in the textbook.

Problems to be discussed in U155 on April 10, 14-16

Do problems:

1. Do problems 3.2 and 3.10 from pages 140 and 141 of the notes. For problem 3.2, use Stirling's approximation - formula 3.20 (2nd ed. 3.19) from the textbook, and compare your result to the upper bound for merging.

2. Prove a lower bound for merging two lists of lengths n and m which meets the upper bound of $n + m - 1$ (assume $n = m$).

3. From the following pages of that book by Baase:

   > Consider the problem of determining if a bit string of length n contains two consecutive zeros. The basic operation is to examine a position in the string to see if it is a 0 or a 1. For each $n = 2, 3, 4, 5$ either give an adversary strategy to force any algorithm to examine every bit, or give and algorithm that solves the problem by examining fewer than n bits.

and

a. You are given n keys and an integer k such that $1 \leq k \leq n$. Give an efficient algorithm to find *any one* of the k smallest keys. (For example, if $k = 3$, the algorithm may provide the first-, second- or third-smallest key. It need not know the exact rank of the key it outputs.) How many key comparisons does your algorithm do? (Hint: Don't look for something complicated. One insight gives a short, simple algorithm.)
b. Give a lower bound, as a function of n and k, on the number of comparisons needed to solve this problems.

4. From Baase's textbook: Suppose $L1$ and $L2$ are arrays, each with n keys sorted in ascending order.

   a. Devise an $O((\lg n)^2)$ algorithm (or better) to find the ith smallest of the $2n$ keys. (For simplicity, you may assume the keys are distinct.)

   b. Give a lower bound for this problem.

5. Design and analyze an efficient algorithm to find the third largest item in an array.

6. Consider the problem of Sorting by Reversals. You are given a permutation of the numbers from 1 to n in an array, A. The operation you have is to choose two indices, i and j, and to reverse the elements in the subarray from $A[i]$ to $A[j]$, inclusive. The objective is to end with a sorted array. For example, given $A = [8, 6, 4, 2, 7, 5, 3, 1]$, the first operation could be $(3, 6)$, resulting in $A = [8, 6, 5, 7, 2, 4, 3, 1]$. Then doing the operations $(2, 4), (3, 4), (5, 7), (5, 6), (1, 8)$ would finish sorting the array.

   a. Give an algorithm which sorts the array in $O(n)$ operations.

   b. Prove that any algorithm needs at least $\Omega(n)$ operations in the worst case.

   c. Why doesn't the information-theoretic lower bound of $\Omega(n \log n)$ apply here?

Assignment due Monday, April 28, 12:15

Note that this is part of your exam project, so it must be approved in order for you to take the exam in June, and you may not work with or get help from others not in your group (though you may talk with Christian Kudahl or myself). You may work in groups of two or three. You may write your solutions in English or Danish, but write very neatly if you do it by hand. Submit the assignment via Blackboard's "SDU Assignment" as one PDF file, with no Danish letters in the file name. Remember to keep a receipt. Turn in one assignment per group.

1. Consider the following game, which we will call "Narrow Cave". In this game, you start at the entrance to a long, but narrow, cave. The cave is divided up into 2 meter

blocks, with lines separating the blocks. The number of blocks is ℓ, and the blocks are numbered from 1 to ℓ, in order, with the block closest to the entrance being block 1. There is one large basket in each block, each one containing a large amount of one type of fruit (a different type for each basket). The player initially knows which type of fruit is found in each block.

As long as the game continues (it can stop at any time, but the player cannot decide when it stops), whenever the player is at the entrance to the cave, a new round starts and the player is told which type of fruit to get. The player must go into the cave, find the basket with that type of fruit, and bring back exactly one piece of that fruit. (Assume that no basket is ever empty.)

During one round, the player, is only allowed to take the type of fruit, which was asked for. He/she is only allowed to take one piece of it.

On returning from getting a piece of fruit in basket i, the player can move that basket i to any block closer to the cave entrance, by repeating switching basket i with the basket currently in front of it. This is an easy operation which has no cost. No other switching of baskets is allowed.

The cost of a round is the block number where the correct piece of fruit was found. The score for the game is the sum of the costs for each round. A low score is best.

For example, suppose that originally block 1 has bananas, block 2 has oranges, and block 3 has apples. Suppose that the order of the fruits the player gets is $\langle$apple, apple, orange$\rangle$. If the player does not move any baskets, the total score is $3 + 3 + 2 = 8$. However, if the player moves the apple basket past the orange and banana baskets during the first round, the cost is $3 + 1 + 3 = 7$, which is better.

If the player breaks any rules, he/she get the total score of ∞, which is worse than any other score possible.

(a) Suppose that originally block 1 has bananas, block 2 has oranges, and block 3 has apples. Suppose that the order of the fruits the player gets is

$$\langle\text{apple, orange, apple, orange, apple}\rangle.$$

Show that it is possible to get a score of only 10.

(b) In this subproblem, the player/algorithm does not know in advance which fruits will be asked for. When executing one round, the player does not know which type of fruit it will need to get in the next round or any other future rounds. Use an adversary argument to show that, no matter what algorithm is used, if there are n rounds and the algorithm never knows what will be asked for in future rounds, the score can be as high as $n \cdot \ell$.

(c) Design a strategy for a player which gets a score of at most $\ell^2 + \frac{n \cdot (\ell+1)}{2}$ whenever there are at most n rounds. (The player here knows at the beginning what fruit will be asked for in which round.)

2. Consider algorithms which should do all of the following for a list of $2n+3$ elements: find the largest element, find the smallest element, find the median element, and partition the remaining $2n$ elements into two sets, each of size n, such that the smallest n elements are all in the first set and the largest n elements in the second set. Suppose the algorithm can be modelled by decision trees which have as their basic operations a comparison operation which can compare two elements, telling which is larger.

   (a) Use an information theoretic argument to prove a lower bound on the number of these comparisons any such algorithm would need to make. How many leaves does your tree have and what is the degree of each node in the tree? (You will need to approximate – give a lower bound.)

   (b) Give a linear time algorithm for solving this problem (finding the required three elements and doing the partitioning) and analyze your algorithm.