Institut for Matematik og Datalogi
Syddansk Universitet

February 7, 2012
JFB

# DM508 – Algorithms and Complexity – 2012
# 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 3. kvartal 2012*. This is identical to the notes from 2010 and 2011 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), usually shortly after the relevant lecture. The lectures and discussion sections are scheduled to be in U26. The lectures will be on Mondays, 8:15–10 (weeks 5–10), on Fridays, 10:15–12 (weeks 6, 8, and 10), and on Wednesday, 10:15–12 (week 5), but one of these will be cancelled. The discussion sections will be on Wednesdays, 10:15–12 (weeks 6–11) and Fridays, 10:15–12 (weeks 5, 7, 9 and 11), but one of these will also be cancelled. The "instruktor" for the course is Magnus Find.

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. If you turn in a later, improved version, mark on the first page, near your name, which version it is (but we hope the system handles this). To do this, login to Blackboard and find DM508. Note that in the upper left hand corner of the screen, there is an icon which you can click on to expand the the menu for the course. It is just to the left of the code for and name of the course. The assignment hand-in is through "SDU Assignment". 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 Magnus 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 Tuesdays and Wednesdays.

There will be an oral exam on March 26 and 27, 2012. 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, January 30

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, February 1

We will finish sections 3.2, 3.3 and 3.5 of the DM508 notes, plus median finding from chapter 9 (section 9.3) in the textbook. We may also begin on NP-completeness, from chapter 34 in the textbook and the section by Papadimitriou and Steiglitz from the course notes.

## Problems to be discussed on February 3

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 $i$th 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 Friday, February 10, 10:15

Note that this is part of your exam project, so it must be approved in order for you to take the exam in March, and you may not work with or get help from others not in your group (though you may talk with Magnus Find 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, and turn in a paper copy.

1. Consider the following solitaire game: Use a standard deck of 52 cards. In all that follows, suits are irrelevant, only the value of the card matters.

   The *foundation* consists of four stacks, where initially, each has exactly one card. The first stack starts with an Ace, the second with a 2, the third with a 3, and the fourth with a 4. The object of the game is to place all 52 cards on the foundation, building all stacks up to kings as follows: The first stack must have the cards A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, in that order. The second must have 2, 4, 6, 8, 10, Q, A, 3, 5, 7, 9, J, K, in that order. The third must have 3, 6, 9, Q, 2, 5, 8, J, A, 4, 7, 10, K, in that order. The fourth must have 4, 8, Q, 3, 7, J, 2, 6, 10, A, 5, 9, K, in that order.

   There are also four *reserve columns*. These are initially empty. As cards are put in these columns, they are placed as the top card, but are placed so that all cards below are visible. There are no rules as to which cards can be placed on top of which, but cards can only be moved from these columns to the foundation, and only the top card can ever be moved. Of course, when one card is moved from a reserve column to the foundation, the card under it becomes the top one in that reserve column.

   After initializing the foundation, with the bottom card of each stack (an Ace, a 2, a 3, and a 4), shuffle the remaining cards. The cards are dealt one at a time, from the top of the deck. When a card is dealt, it must be placed either on a foundation stack, or as the top card on one of the four reserve columns. After it is placed, one may move other cards from the reserve columns to the foundation, if this is possible, following the required order in the foundation.

   (a) Prove that it can be necessary to have as many as 34 cards in the reserve columns at the same time. Write your proof as an adversary argument.

   (b) Prove that in a game which can be won, it can be necessary to play at least 3 cards from the reserve columns to the foundation (at least once), between dealing two cards from the deck.

   (c) Suppose that in every reserve column you had both a Q and a K, and the Q was always below the K. Is it possible to win? Why or why not? Can you generalize this to the K and other cards than the Q? How?

2. Consider algorithms for merging three sorted lists, each of length $n$, which can be modelled by decision trees which have as their basic operations a comparison operation which can compare either 2 and 3 items in one operation and give the ordering between them (the operation will tell which is largest and which is smallest, and this tells which is middle when there are three items). 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?