Institut for Matematik og Datalogi
Syddansk Universitet

January 27, 2010
JFB

DM508 – Algorithms and Complexity – 2010
Lecture 1

Textbook and notes

Introduction to Algorithms, 2nd edition, by T. Cormen, C. Leiserson, R. Rivest, and C. Stein, MIT Press, 2001. (There is a newer edition, but we will use this older one since most students have it. If you don't have it, you can get the newer edition from the bookstore and refer to the older edition in the library for correct page numbers, section numbers, and problem numbers. I will sometimes put these numbers for the newer edition in parentheses.)

Extra notes (available in the bookstore): *DM508 Algoritmer og kompleksitet, Noter 3. kvartal 2010*. This is identical to the notes from 2009 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 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 are scheduled to be in U140 on Mondays, 8:15–10 (weeks 5–11), Wednesdays, 8:15–10 (weeks 6, 8, and 10), and Tuesday, 10:15–12 (week 5), but two of these will be cancelled. The discussion sections will be on Tuesdays, 10:15–12 (weeks 6–11) in U140, Wednesdays, 8:15–10 (weeks 5, 7, 9 and 11), in U140, and Thursday, 12:15–14 (week 11) in U20, but two of these will also be cancelled. The "instruktor" for the course is Magnus Find. We expect that the discussion section will be in English, since there are some foreign exchange students in the class.

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. To do this, login to Blackboard and find DM508. Choose "Course Tools" from the "Tools" menu on the left. Choose "Assignment hand in", fill out the form and end with "Submit". You can print out your receipt and you will get an e-mail. 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. 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 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 13:30–14:15 on Tuesdays and 10:30–11:15 on Wednesdays.

There will be an oral exam on April 9 and 10, 2009. 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, February 1

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 2

We will finish sections 3.2, 3.3 and 3.5 of the DM508 notes, plus median finding from chapter 9 (sections 9.2 and 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 (use Stirling's approximation - formula 3.19 (3rd ed. 3.20) from the textbook, and compare your result to the upper bound for merging, rather than to the lower bound mentioned) and 3.10 from pages 140 and 141 of the notes.

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

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 nth 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 Wednesday, February 17, 8:15

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. 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.

1. Consider the problem of determining whether or not an $n \times m$ matrix contains a 9×9 submatrix with all ones. For example, a 12×12 matrix A with $A[5,5] = 0$ does not contain such a submatrix, but the matrix

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

does.

Assume that both n and m are at least 9 and that there are no values in the matrix other than zero and one. The operations we will be counting are checking to see if an entry in the matrix is zero or one. Clearly, an algorithm can solve this problem by checking all $n \cdot m$ entries. We will consider if it is possible to do better than this.

a. What is the minimum number of zeros in any matrix where the correct answer is that the matrix does not contain a 9×9 submatrix with all ones? Justify your answer by (1) describing a placement of zeros which gives such a matrix and (2) assuming that the matrix has one fewer zero and showing that it must contain such a submatrix.

b. Give an algorithm which solves this problem by checking less than $n \cdot m$ entries in the matrix when $n = m = 11$.

c. Prove that any algorithm needs at least 90 operations in the worst case when $n = m = 11$.

Hint: First let your adversary answer zero whenever a corner is checked. Consider the 9×9 submatrix in the center and its top and bottom row plus its left and right column. Case 1 is when all nine entries in one these rows and columns are checked before the nine non-corner entries in any of the outermost rows and columns. Suppose the adversary makes this last entry a zero. Find a submatrix which the adversary can force the algorithm to continue checking until its last entry is checked. What should the adversary answer outside that submatrix?

In Case 2, the nine non-corner entries of some outermost row or column is finished before any of these outside rows and columns in the center submatrix are finished. Suppose the adversary makes this last entry a zero. Find a submatrix

which the adversary can force the algorithm to continue checking until its last entry is checked. What should the adversary answer outside that submatrix?

2. Consider the following algorithm:

Input: An integer array A of length n

for $i = 2$ to n **do**
    $j = i$;
    while $j > 1$ and $A[j] < A[j-1]$ **do**
        temp $\leftarrow A[j]$;
        $A[j] \leftarrow A[j-1]$;
        $A[j-1] \leftarrow$ temp;
    end while
end for
return(A)

Give two proofs that this is not an algorithm for sorting an array. One proof should refer to a lower bound result you already know. The other should not (what should an adversary do?).