Institut for Matematik og Datalogi
Syddansk Universitet

September 8, 2004
JFB

# DM19 – Algorithms and Complexity – E04 – Lecture 2

## Lecture, September 6

We began with an introduction to the course. Quicksort, from sections 7.1 and 7.2, was quickly reviewed. Then, Randomized Quicksort, from sections 7.3 to 7.4 in the textbook, was presented. Randomized Quicksort can also be found in the course notes, those by Motwani and Raghavan. Counting Sort and Radix Sort from sections 8.2 and 8.3 were also presented.

## Lecture, September 13

Lower bounds from section 2.4 of the of the notes (by Baase) will be discussed (part of this is also in section 8.1 of the textbook). Then we will begin on adversary arguments from sections 3.1 and 3.2 of the same notes.

## Lecture, September 20

We will cover section 3.3 and 3.5 of the first set of notes, plus median finding from chapter 9 (sections 9.2 and 9.3) in the textbook.

## Problems to be discussed in week 38

- Do the following problems from the textbook: 8.1-1, 8.1-3 (just do the first part), 8.2-2. 8.2-4.
- Do problems 3.2 (use Stirling's approximation - formula 3.17 from the textbook, and compare your result to the upper bound in section 7.1 of the textbook, rather than to the lower bound mentioned) and 3.10 from page 140 and 141 of the notes.
- 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.

## Announcement

I lighed med tidligere år afholdes der obligatoriske, studieorienterende samtaler for alle vore studerende på studieretningerne matematik, mat.øk., datalogi og anvendt matematik, som læser på andet studieår og opefter. Samtalerne foretages af lærerrepræsentanter fra IMADA i ugerne 39 og 40. Der er tilmeldingslister på sekretariatet.