Institut for Matematik og Datalogi
Syddansk Universitet

January 30, 2007
JFB

DM508 – Algorithms and Complexity – F07
Lecture 2

Lecture, January 30

We began with an introduction to the course. Randomized Quicksort, from sections 7.3 to 7.4 in the textbook, was presented. Counting sort from section 8.2 was also presented. We began on lower bounds using information theoretic arguments, presenting the concept of the decision tree model.

Lecture, February 1

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 6

We will cover sections 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 amortized analysis from chapter 17.

Problems to be discussed on February 7

Do problems:

1. Do problems 3.2 (use Stirling's approximation - formula 3.17 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. 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.

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

   - 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.)
   - Give a lower bound for this problem.

4. Design an algorithm for finding the second largest item in an array, which is similar to the standard algorithm for finding the largest. Keep track of the largest and second largest at each step. How many comparisons does your algorithm do in the worst case?

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

Assignment due Thursday, February 15, 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 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. Do problem 7-6 in the textbook. Read both parts of the problem before beginning.

2. Consider the problem of sorting where all the keys are either zero or one. Thus, all records with key value zero should be placed before all records with key value one. Rather than comparison, the fundamental operation will be checking if a key has the value zero or one.

   (a) Use an information theoretic argument to prove a lower bound on the worst case number key checks necessary to solve this problem. Give the best bound you can.

   (b) Use an adversary argument to prove a lower bound on the worst case number key checks necessary to solve this problem. Give the best bound you can.

(c) Give an algorithm for this problem. Analyze it, showing that the algorithm uses no more key checks than is necessary in the worst case.