Institut for Matematik og Datalogi
Syddansk Universitet – Odense Universitet

September 7, 1999

JFB

DM19 – E99 – Lectures 3 and 4

Lecture, September 1

We began with an introduction to the course, touching on the topics covered in the first 4 chapters in the textbook. Merge-Sort was the running example. The "Master Theorem" for solving recurrence relations was presented. Randomized Quicksort was analyzed.

Lecture, September 3

We discussed randomized algorithms in general. Radix sort was quickly reviewed, and counting sort was presented. Then lower bounds from section 2.4 of the first set of notes were discussed. We introduced decision trees and finished the lower bound on the worst case of any algorithm which sorts via comparisons. We began on the lower bound for the expected running time of any algorithm which sorts via comparisons.

Lecture, September 8

We will discuss the information theoretic lower bounds from section 2.4 from the first set of notes.

Lecture, September 10

We will cover sections 3.1, 3.2, and 3.3. If there is time, we will begin on section 3.5 from the first set of notes and median finding from chapter 10 in the textbook.

Lectures, Week 37

We will continue with median finding and then briefly go over some material from sections 5.4, 5.5 and 23.1 on graphs. Then we will review minimum spanning tree algorithms from chapter 24 and introduce matroids from chapter 17.

Problems to be discussed in week 38

9.1-3 (just do the first part), 9.1-6, 9.2-5., and 9-1f.

3.2 (use Stirling's approximation - formula 2.11 from the textbook) from page 140 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.

Studieorienterende samtaler

I lighed med tidligere år afholdes der obligatoriske, studieorienterende samtaler for alle vore studerende på studieretningerne matematik, mat.øk., datalogi, anvendt matematik og datateknologi, som læser på andet studieår og opefter. Samtalerne foretages af lærerrepræsentanter fra både IMADA og MIP i ugerne 38 og 39. Der er tilmeldingslister på sekretariaterne. Se det detaljerede opslag på WWW via Studieudvalgets hjemmeside:

`http://www.imada.sdu.dk/Structure/StudiesCommittee/`