Sample Exam [1]

Oct. 15 2009
kl 1400 – 1800
byggnad V

Ansvarig:
Devdatt Dubhashi Tel. 073 932 2226 Rum 6479 EDIT

Poäng:	60	
Betygsgränser:	Chalmers	5:48, 4:36, 3:24
	GU	VG:48, G:28
	Doktorander	G:36
Hjälpmedel:	kursboken, anteckningar	

- Recommended: First look through all questions and make sure that you understand them properly. In case of doubt, do not hesitate to ask.
- **Answer all questions in the given space on the question paper (the stapled sheets of paper you are looking at). The question paper will be collected from you after the exam. Only the solutions written in the space provided on the question paper will count for your points.**
- Use extra sheets only for your own rough work and then write the final answers on the question paper.
- Answer concisely and to the point. (English if you can and Swedish if you must!)
- Code strictly forbidden! Motivated pseudocode or plain but clear English/Swedish description is fine.

Lycka till!

[1] 2005 läsperiod 1, INN200 (GU) / TIN090 (CTH).

Problem 1 Finding the m smallest [10] An array T contains n numbers. You want to find the m smallest numbers where m is much smaller than n. Would you

(a) sort T and pick the first m,

(b) call `select(T,i)` for $i = 1, 2, \ldots, m$, or

(c) use some other method?

Justify your answer by comparing the time complexity of the different methods.

Problem 2 Changing Coins [10] Suppose that you have coins available in denominations (i.e. values) of $1, p, p^2, \ldots, p^n$, where $p > 1$ and $n \geq 0$ are integers (for example, $1, 5, 25$ with $p = 5$ and $n = 2$). You have an unlimited number of coins of each type. What algorithm will you use to find the minimum number of coins required to change a given amount? Give a short outline of a proof in justification.

Problem 3 Minimize Idle Time [10] Chalmers super-commputing center has its super computer running from 10 AM to 5 PM. Let's translate time and denote this interval $[0, M]$. At the start of the day it receives n requests where request i wants to reserve the supercomputer from time $0 \leq s_i$ to $f_i \leq M$. You want to schedule the jobs so that the supercomputer time is left idle as little as possible.

(a) Give counter-examples to show that neither of the follwing greedy heuristics gives the optimal solution: longest job first, earliest finishing job first.

Now we develop a dynamic programming algorithm. Let $OPT(i)$ denote the optimal solution considering only jobs $1 \cdots i$.

(b) First say why we may sort the jobs in order of finishing time and assume that the last finishing job has $f_i = M$.

(c) What is $OPT(1)$?

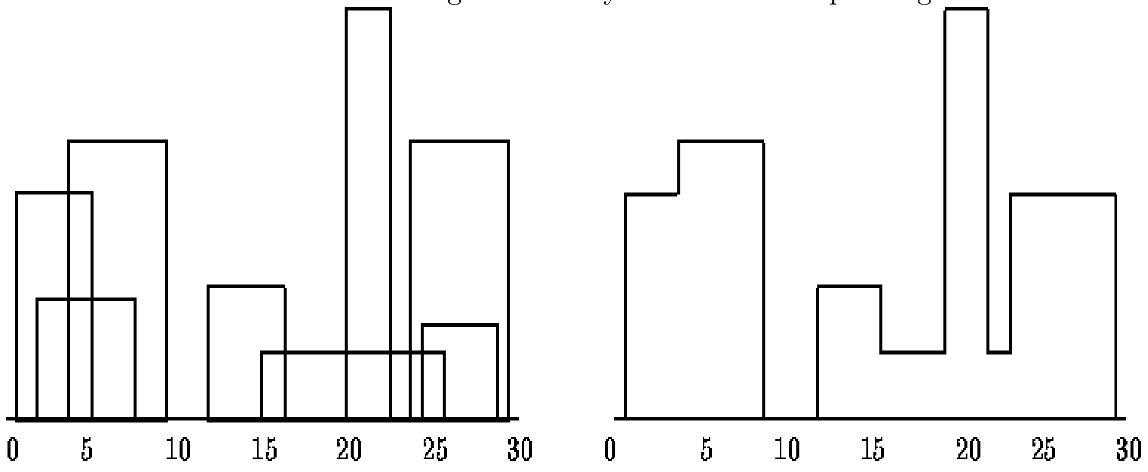
(d) Give a recurrence for $OPT(i)$ based on what we do with job i.

(e) Implement the recurrence efficiently in pseudocode and analyse its time and space complexity.

(f) How would you modify your algorithm to also produce a list of the jobs scheduled?

Problem 4 Manhattan Skyline [10] With the advent of high speed graphics workstations, CAD (computer-aided design) and other areas (CAM, VLSI design) have made increasingly effective use of computers. One of the problems with drawing images is the elimination of hidden lines – lines obscured by other parts of a drawing.

You are to design a program to assist an architect in drawing the skyline of Manhattan given the locations of the buildings in the city. As we know, in Manhattan, all buildings are rectangular in shape and they share a common bottom (the city is very flat). The city is also viewed as two-dimensional. There are n buildings specified by three arrays $L[1 \dots n], R[1 \dots n]$ and $H[1 \dots n]$. Building i has left side at coordinate $L[i]$, right coordinate at $R[i]$ and height $H[i]$. In the figure below, buildings are shown on the left with $L = (12, 2, 14, 19, 3, 1, 24, 23)$ $R = (16, 7, 25, 22, 9, 5, 28, 29)$ and $H = (7, 6, 3, 18, 13, 11, 4, 13)$. The skyline, shown on the right, is represented by the two arrays: $(1, 3, 9, 12, 16, 19, 22, 23, 29)$ and $(11, 13, 0, 7, 3, 18, 3, 13, 0)$ where the first array gives the x coordinate and the second the height of the skyline at the corresponding co-



ordinate.

(a) Give the overall strategy of a Divide-and-Conquer algorithm.

(b) Explain clearly what the recursive subproblems should return.

(c) Explain clearly the Conquer step: how will you combine the solutions to the subproblem to get a solution for the original problem? This is the most crucial part!

(d) Write a recurrence for the running time and give a short justification. Deduce the running time of the algorithm in $O()$ notation.

Problem 5 Taxi Göteborg [10] You are working for Taxi Gẗeborg and are dveloping software for an automated response system. At a given point in time, there are m taxis at different locations in the city and there are n customers requesting a taxi, also from various parts of the city. Customer i needs the taxi within t_i minutes. Using some other nifty software using Dijkstra with GPS, you can determine if taxi j can reach customer i within t_i minutes in Gẗeborg traffic. Now you need to determine the maximum number of customers you can satisfy. Give an efficient algorithm and analyse its running time.

Problem 6 Multiple Interval Scheduling [10] We've seen the Interval Scheduling problem in class; here we consider a computationally much harder version of it that we'll call MULTIPLE INTERVAL SCHEDULING.As before, you have a processor that is available to run jobs over some period of time. (E.g. 9 AM to 5 PM.)
People submit jobs to run on the processor; the processor can only work on one job at any single point in time. Jobs in this model, however, are more complicated than we've seen in the past: each job requires a *set* of intervals of time during which it needs to use the processor. Thus, for example, a single job could require the processor from 10 AM to 11 AM, and again from 2 PM to 3 PM. If you accept this job, it ties up your processor during those two hours, but you could still accept jobs that need any other time periods (including the hours from 11 to 2).
Now, you're given a set of n jobs, each specified by a set of time intervals, and you want to schedule as many jobs as possible. Call this problem OPT-MS. Consider the decision version of the problem. Y/N-MS: For a given number k, is it possible to accept at least k of the jobs so that no two of the accepted jobs have any overlap in time?

(a) Suppose someone gives you a black box to solve the decision version Y/N-MS in polynomial time. Show that then you could use it to solve OPT-MS also in polynomial time.

(b) Show that Y/N-MS is in NP.

(c) State in one line the significance of showing that this problem is NP-complete.

(d) Show that Y/N-MS is NP-complete. (Hint: reduction from INDEPENDENT-SET: think of the nodes of a graph as job requests with the incident edges as its associated time intervals.).