Technische Universität München
Fakultät für Informatik
Lehrstuhl für Algorithmen und Komplexität
Prof. Dr. Susanne Albers
Dr. Arindam Khan
Maximilian Janke
Luisa Peter

Winter Semester 2018/19
Problem Set 10
January 15, 2019

Advanced Algorithms

Due January 22, 2019 at 10:00

Note: You are welcome to submit in groups of two. If you wish to submit individually, you should solve one of the Exercises 1 and 2 as well as one of the Exercises 3 and 4.

Exercise 1 (Greedy strategy for interval scheduling – 10 points)

Suppose that instead of always selecting the interval with the earliest finish time, we select the interval with the latest start time that is compatible with all previously selected intervals. Adjust the algorithm *Greedy** from the lecture appropriately and, just as in the lecture, prove using induction that it yields an optimal solution.

Exercise 2 (Scheduling unit-time tasks with deadlines – 10 points)

Let $S = \{1, 2, \ldots, n\}$ be a finite set of n unit-time tasks, i.e. jobs that require exactly one unit of time to complete. The tasks should be scheduled on a single processor, which means that the ith task in the schedule begins at time $i-1$ and completes at time i. Each task i is supposed to finish by its deadline $1 \leq d_i \leq n$ and a penalty $w_i \geq 0$ is incurred if task i is not finished by time d_i. No penalty is incurred if a task finishes by its deadline. The goal is to find a schedule for S that minimizes the total penalty incurred by missed deadlines.

Consider the following algorithm. Let all n time slots be initially empty, where time slot i is the unit-length slot of time that finishes at time i. We consider the tasks in order of monotonically decreasing penalty. When considering task j, if there exists a time slot at or before j's deadline d_j that is still empty, assign task j to the latest such slot. If there is no such slot, assign task j to the latest of the as yet unfilled slots.

Prove that this algorithm always gives an optimal answer.

Exercise 3 (Dynamic programming – 10 points)

A square Q with side length $n \in \mathbb{N}$ is subdivided into n^2 squares of side length 1 that are all parallel to Q. Each one of the small squares is either colored grey or white. The goal is to determine the side length and the position of the biggest square in Q that is completely colored in white and parallel to Q. In case there is more than one such square, any one of those may be chosen. See Figure 1 for an example. Design an algorithm that solves the problem and employs dynamic programming. Describe your algorithm and argue that it is correct. Include its complete recursive equation.

(Advanced version: If you want, you can design an algorithm that finds the biggest rectangle in Q that is parallel to Q and completely colored white instead of the biggest

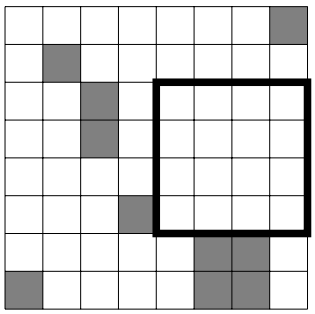

Figure 1: The bold edging marks a biggest square that is completely colored in white.

square. This version of the exercise is optional. We won't provide a sample solution and you won't receive bonus points for solving it.)

Exercise 4 (Optimizing space in a library – 10 points)

A library bought a new bookcase. The shelves all have the same fixed width $S > 0$ but the vertical distances between shelves can be adjusted individually according to the heights of the books placed on them. The order in which the n books are placed on the shelves is fixed by the cataloging system of the library: $b_1, b_2, \ldots, b_n$. A book b_i has width $w_i > 0$ and height $h_i > 0$. Using dynamic programming, design an algorithm that places the books on the shelves such that the total space usage, i.e. the sum of the heights of the largest book on each shelf, is minimized. Argue that your algorithm always finds an optimal solution and state its time complexity.