Exercise sheet 2 – Solutions

COMP6741: Parameterized and Exact Computation

Serge Gaspers

Semester 2, 2017

Exercise 1. Prove the following generalization of Lemma 3 [Lawler '76]: For any graph G on n vertices, if G has a k-coloring, then G has a k-coloring where one color class is a maximal independent set in G of size at least n/k.

Solution. Start from a k-coloring of G. Consider the largest color class C (it has $\geq n/k$ vertices), and let S be a maximal independent set of G with $C \subseteq S$. Replace C by S (and remove the vertices in S from all other color classes). The new coloring uses a minimum number of colors and one color class is a maximal independent set of size at least n/k.

Exercise 2. In the Meeting Most Deadlines problem, we are given n tasks $t_1, \ldots, t_n$, and each task t_i has a length ℓ_i, a due date d_i, and a penalty p_i which applies when the due date of task t_i is not met. The problem asks to assign a start date $s_i \geq 0$ to each task t_i so that the executions of no two tasks overlap, and the sum of the penalties of those tasks that are not finished by the due date is minimized.

Meeting Most Deadlines

Input: A set $T = \{t_1, \ldots, t_n\}$ of n tasks, where each task t_i is a triple (ℓ_i, d_i, p_i) of three non-negative integers.

Output: A schedule, assigning a start date $s_i \in \mathbb{N}_0$ to each task $t_i \in T$ s.t.

$$\sum_{i \in \{1,\ldots,n\} \,:\, s_i + \ell_i > d_i} p_i$$

is minimized, subject to the constraint that for every $i, j \in \{1, \ldots, n\}$ with $i \neq j$ we have that $s_i \notin \{s_j, s_j + 1, \ldots, s_j + \ell_j - 1\}$.

(a) Show that the Meeting Most Deadlines problem can be solved in $O^*(n!)$ time by reformulating it as a permutation problem.

(b) Design an algorithm solving the Meeting Most Deadlines problem in $O^*(2^n)$ time.

Hints: there are many ways to solve this exercise. Several dynamic programming algorithms solve the problem in $O^*(2^n)$ time and space. The following hints guide you towards a polynomial-space algorithm.

- Show that there is an optimal *tightly packed* solution, i.e., with no gaps in the schedule.
- Show that there is an optimal tightly packed solution where no task t_j is scheduled before a task t_i, where the deadline of task t_j is not met, but the deadline of task t_i is met.
- Show that there is an optimal tightly packed schedule where the tasks whose deadlines are met are ordered by non-decreasing deadlines.