TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Department of Computer Science
Sebastian Wild

Issue Date: 20.04.2015
Version: 2017-04-20 13:03

1st Exercise Sheet for Advanced Algorithmics, Summer 17

Hand In: *Until Wednesday,* ***03.05.2015****, 12:00 am, hand-in box in 48-4 or via email.*

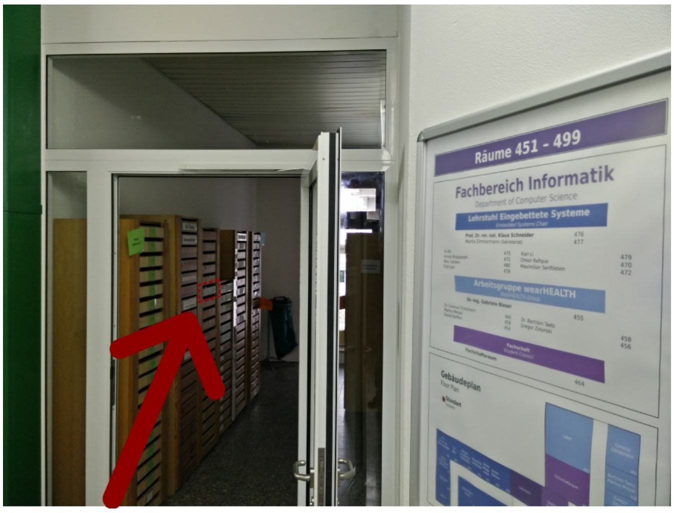
Exercise Policy

- The exercise problems take up the material from lecture. Working on them deepens understanding and increases proficiency, and I consider them an integral part of the course.
- You will require at least a $1/e$ fraction of the total points to take the oral exam.
- You may collaborate on the problems and hand in your solutions as a group of 2–3 students.
- If you make use of external sources, make sure to clearly mark the used parts as such, including unambiguous references to the original sources. In case of websites, at least add the access date and use permanent links whenever possible (e. g., for Wikipedia). You may cite any publicly accessible sources, but take their credibility into account: you are responsible to verify what you cite.

  Collaboration between groups on conceptual solutions to the exercises is fine, but each group must work out a detailed submission individually. In these cases, add a quick statement with which groups you joined forces to work on a given problem.

  Plagiarism is not tolerated in academia, be it external sources or other groups' submissions. Severe cases can cause expulsion from university.

N.B.: The hand-in boxes have been moved to the 4th floor:



Problem 1

15 + 15 + 40 + 40 points

Consider the following problems. Prove that they are $\mathcal{NP}$-complete, respectively.

You may use that the standard problems 3SAT, Hamilton Path, Clique, Knapsack, Subset Sum, Vertex Cover and Traveling Salesperson are $\mathcal{NP}$-complete.

a) **Subgraph Isomorphism:**

> **Input:** Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$.
>
> **Question:** Does G_1 have a subgraph which is isomorphic to G_2?

b) **Longest Path:**

> **Input:** A graph $G = (V, E)$ with edge weights $w : E \to \mathbb{N}$, nodes $s, t \in V$ and $k \in \mathbb{N}$.
>
> **Question:** Is there a simple path from s to t in G that has weight – w. r. t. w – at least k?

c) **Optimal Composition:**

> **Input:** A finite set A, a set $C \subseteq 2^A$ of subsets of A and $J \in \mathbb{N}$.
>
> **Question:** Is there a sequence of $j \leq J$ unions
>
> $$\langle x_1 \cup y_1, x_2 \cup y_2, \ldots, x_j \cup y_j \rangle \ ,$$
>
> so that
>
> i) $x_i \cap y_i = \emptyset$ for all $i \in [1..j]$,
>
> ii) $x_i, y_i \in \big\{\{a\} \mid a \in A\big\} \cup \big\{x_k \cup y_k \mid k < i\big\}$ for all $i \in [1..j]$ and
>
> iii) there is an i with $x_i \cup y_i = c$ for every $c \in C$?

d) **Scheduling with Missed Deadlines:**

Input: A set $T = \{t_1, \ldots, t_n\}$ of jobs, each with duration 1, deadlines $d : T \to \mathbb{N}$, a partial order $\lessdot$ on T and a natural bound $k \le n$.

Question: Is there a mapping $\sigma : T \to \{1, 2, \ldots, n\}$ which schedules the jobs according to the following conditions?

i) Only one job is scheduled per time slot, that is

$$i \neq j \implies \sigma(t_i) \neq \sigma(t_j)$$

for all $i, j \in [1..n]$.

ii) Schedule σ is *happens-before* consistent with $\lessdot$, that is

$$t_i \lessdot t_j \implies \sigma(t_i) < \sigma(t_j)$$

for all $i, j \in [1..n]$.

iii) Not too many jobs are late, that is

$$|\{t \in T \mid \sigma(t) > d(t)\}| \le k.$$

Advice: Remember what you know from basic courses! What are the necessary steps?

Problem 2

20 + 30 points

Design a solution algorithm for the following search problems using a polynomial number of calls to a decider for the corresponding decision problem, e.g., if the search problem under consideration is SAT, assume there is a procedure `boolean isSAT(`ϕ`)` that checks if a given formula ϕ is satisfiable or not. Use calls to `isSAT` to compute a satisfying variable assignment for ϕ, if ϕ is satisfiable, or output "No-instance" otherwise.

Counting calls to the decider as a single instruction, your algorithm must run in polynomial time.

a) SAT:
Given a boolean formula ϕ over variables $x_1, \ldots, x_n$ (not in any specific normal form, but given in a reasonable encoding[1] of your choice), compute a satisfying assignment $v : [n] \to \{0, 1\}$ of the variables if one exists, or output "not satisfiable".

b) Hamilton-Cycle:
Given an undirected (unweighted) graph $G = (V, E)$ (without self-loops and parallel edges), compute a Hamiltonian cycle (i.e., a simple closed path visiting each vertex in V exactly once) if one exists, or output "not Hamiltonian" otherwise.

Bonus Question: Can you find a search problem whose decision version is in $\mathcal{NP}$, but for which a similar solution of the search problem is (or seems) impossible?

[1] i.e., with length of order $\log n$ times the total number of variable occurrences and operators in ϕ.