Instructions

- Classroom Problems C4.1–C4.3 will be discussed and solved at the tutorial session on Wed 5 Feb, 14–16, Room T4 (A238). No credit is given for these problems.
- Homework Problems H4.1–H4.3 you should solve on your own, and submit your solutions via the MyCourses interface by the deadline of Tue 11 Feb, 23:59, These problems will be individually graded on a scale of 0–2 points per problem.
- In preparing your solutions to the Homework Problems:
  1. Justify your solutions, be precise, and provide sufficient detail so that it is easy to follow your reasoning.
  2. Submit your solutions as an easily readable, single pdf file, which is either typeset or written in full sentences and clean handwriting.
  3. **[Code of Conduct]** You can discuss the problems with your colleagues and the course's teaching staff, but you must write the presentations of your solutions *independently* and *individually*, without any notes from such discussions.

Classroom Problems

C4.1 Show that for any proper complexity function $f(n) > n$, all languages in complexity class TIME$(f(n))$ reduce[1] to the language

$$U_f = \{M; x \mid M \text{ accepts } x \text{ in } f(|x|) \text{ steps}\}.$$

In such a case one would say that U_f is TIME$(f(n))$-*hard*.

C4.2 Language $A \subseteq \Sigma^*$ is *polynomial-time (many-one) reducible* (or "Karp-reducible") to language $B \subseteq \Gamma^*$, denoted $A \leq_m^p B$ or $A \leq_P B$, if there is a function $R : \Sigma^* \to \Gamma^*$, computable by a deterministic Turing machine in time $O(n^k)$ for some k, such that for all strings $x \in \Sigma^*$,

$$x \in A \quad \textit{iff} \quad R(x) \in B.$$

(Note that by Proposition 8.1 on the lecture slides, if $A \leq_L B$, then $A \leq_P B$.) Show that every language $A \subseteq \Sigma^*$ in the complexity class $\mathbf{P}$, except for $A = \emptyset$ and $A = \Sigma^*$, is also $\mathbf{P}$-complete with respect to polynomial-time reductions.

[1] Unless otherwise noted, in this course "reduction" means log-space reduction. In the present case the reductions can in fact be made constant-space or even 0-space.

C4.3 Consider the following SET COVER decision problem:

> INSTANCE: A family $F = \{S_1, \ldots, S_n\}$ of subsets of a finite set U and an integer K.
> QUESTION: Is there a subfamily of at most K sets in F whose union is U?

(i) Show that SET COVER $\in$ **NP**.

(ii) Design a reduction from the **NP**-complete VERTEX COVER decision problem to SET COVER, thus proving that also SET COVER is **NP**-complete.

Homework Problems

H4.1

(a) Show that the special case of SAT, in which each clause has either exactly two literals or at most one negative literal, is **NP**-complete.

(b) Show that the problem 1-IN-3SAT is **NP**-complete, where the input is a 3cnf-formula as in ordinary 3SAT, but the satisfiability condition is that *exactly one* literal in each clause is set to be true. (2 points)

H4.2 Consider the following DOMINATING SET decision problem:

> INSTANCE: An undirected graph $G = (V, E)$ and an integer K.
> QUESTION: Does the graph G contain a set $U \subseteq V$ of at most K vertices, such that for any $v \in V \setminus U$ there is an edge connecting it to some vertex $u \in U$?

(i) Show that DOMINATING SET $\in$ **NP**.

(ii) Design a reduction from the NP-complete VERTEX COVER decision problem to DOMINATING SET, validate the reduction conditions and explain why your reduction in logspace-computable. (2 points)

H4.3 The goal of this problem is to prove that $\mathbf{P} \neq$ SPACE(n) [C. Wrathall 1978]. The proof is amazingly simple, considering that these types of separation results are usually beyond reach of present-day proof techniques.

Say that a set A is *linear-complete* for a complexity class $\mathbf{C}$, if:

(i) $A \in \mathbf{C}$, and

(ii) for every $B \in \mathbf{C}$, there exists a function f computable by a Turing machine in linear time and constant workspace such that $x \in B \Leftrightarrow f(x) \in A$ holds for all input strings x.[2]

[2] Actually, by using the Turing machine's finite state control, we could assume the constant to be 0.

Now consider the following "padded universal set":

$U = \{M; x; 1^{|M|\cdot|x|} \mid$ Turing machine M accepts input x in (work)space $|x|\}$.

(i) Show that for any constant $k \geq 0$, the mapping $x \mapsto 1^{k|x|}$ can be computed in linear time and constant workspace.

(ii) Show that U is linear-complete for the class SPACE(n).

(iii) Show that if $\mathbf{P} =$ SPACE(n), then U is linear-complete for $\mathbf{P}$ and $U \in$ TIME(n^r) for some $r \geq 0$.

(iv) Based on the previous, show that if $\mathbf{P} =$ SPACE(n), then $\mathbf{P} \subseteq$ TIME(n^r), which is a contradiction. (Why?) (2 points)