**Technische Universität München (I7)**
**Prof. J. Křetínský / Dr. C.H. Broadbent**

**Summer term 2016**
**28. April 2016**

# Computational Complexity – Homework 3

Discussed on 02.05.2016.

## Exercise 3.1

Is **NP** closed under intersection, resp. union?

## Exercise 3.2

Prove that DOUBLE-SAT = $\{\langle\Phi\rangle \mid \Phi$ is a Boolean formula with at least two satisfying assignments $\}$ is **NP**-complete.

## Exercise 3.3

(a) Let $M$ be a Turing machine which decides SAT, and let $\phi$ be a CNF formula with $n$ variables.

Design a recursive algorithm which computes a satisfying assignment for $\phi$ (if $\phi$ is satisfiable) using at most $2n+1$ calls to $M$ plus some additional polynomial-time computation.

(b) Assume that $L \subseteq \{1\}^*$ is a *unary* language which is also **NP**-complete.

Show that then SAT $\in$ **P**.

*Hints*:

- Again write a recursive program but limit the number of recursive calls by using a hash map. Use as hash function a polynomial-time reduction $f$ of SAT to $L$.
- Consider then the call tree of your program for a given input. Show that two nodes $v, v'$ which do not lie on a common path from the root to a leaf correspond to formulae $\phi_v, \phi_{v'}$ with $f(\phi_v) \neq f(\phi_{v'})$.

## Exercise 3.4

In the lecture, you have seen the definition of "polynomial-time reducible" $\leq_p$:

> For two languages $A, B \subseteq \{0,1\}^*$ we write $A \leq_p B$ if there is a function $f : \{0,1\}^* \to \{0,1\}^*$ computable in polynomial time such that $x \in A \Leftrightarrow f(x) \in B$ for all $x \in \{0,1\}^*$.

Similarly, the notion of "log-space reducible" $\leq_{\log}$ is defined but this time the function $f$ has to be computable by a Turing machine using at most $\mathcal{O}(\log n)$ space.

(a) Show that $A \leq_{\log} B$ implies $A \leq_p B$.

(b) Show that for any two languages $A, B$ in **P** *with* $B \neq \emptyset, \{0,1\}^*$ we have $A \leq_p B$.

*Remark*: Using $\leq_{\log}$ one can also define **P**-complete problems in a meaningful way.

(c) Argue that $\leq_{\log}$ is also transitive, i.e., if $A \leq_{\log} B \leq_{\log} C$, then also $A \leq_{\log} C$.

*Hint*: This is not as straightforward as for polynomial-time reductions. Why?

## Exercise 3.5

(a) Show that **NP**=co**NP** if and only if 3SAT and TAUTOLOGY are polynomial-time reducible to each other.

(b) A *strong* nondeterministic Turing machine (sNDTM) is a NDTM which has three possible outputs: "1", "0", "?". An sNDTM $M$ decides a language $L$ if: (i) for $x \in L$ every computation of $M$ on $x$ yields "1" or "?" and there is at least one computation of $M$ on $x$ which yields "1". (ii) for $x \notin L$ every computation of $M$ on $x$ yields "0" or "?" and there is at least one computation of $M$ on $x$ which yields "?".

Show that $L$ is decided by an sNDTM in polynomaial time iff $L \in$ **NP** $\cap$ co**NP**.

**Exercise 3.6**

*Notation*: For $n$ a natural number let $[n]$ be the set $\{1, 2, \ldots, n\}$.

The KNAPSACK problem is defined as follows:

We are given $n$ items where item $i$ has both a weight $w_i \in$ and a value $v_i$. We are also given a maximal weight $W$ the knapsack can hold and a target value $V$. (All numbers are assumed to be positive integers.) A selection $S \subseteq [n]$ then has total weight $w(S) := \sum_{i \in S} w_i$ and total value $v(S) := \sum_{i \in S} v_i$. A selection $S$ is a *solution* if $w(S) \leq W$ and $v(S) \geq S$ hold.

(a) Give a reasonable encoding of KNAPSACK and show that KNAPSACK is in **NP**.

(b) Assume you are given an algorithm for deciding KNAPSACK running in polynomial time.

Construct from it a polynomial-time algorithm which computes the maximal $V_{\max}$ for which a given instance of KNAPSACK has a solution.

(c) Give an algorithm for deciding KNAPSACK in time $\mathcal{O}(nW)$.

*Hint*: Use dynamic programming to produce a table $V(w, i)$ where

$$V(w, i) := \max \{v(J) \mid J \subseteq [i] \text{ and } w(J) = w\}.$$

*Remark*: Note that $W$ is exponential in the size of the representation of $W$.

(d) We define MULTI-KNAPSACK to be the problem where for every item $i \in [n]$ we are given $M$ values $v_i^p$ ($p \in [M]$) and $N$ weights $w_i^q$ ($q \in [N]$) with corresponding target values $V^p$ and total weights $W^q$. (All numbers are assumed to be positive integers.) A selection $S \subseteq [n]$ is then a solution of the MULTI-KNAPSACK instance if

$$\forall p \in [M] : \sum_{i \in S} v_i^p \geq V^p \text{ and } \forall q \in [N] : \sum_{i \in S} w_i^q \leq W^q.$$

Show that MULTI-KNAPSACK is also in **NP** and give a reduction 3SAT $\leq_p$ MULTI-KNAPSACK .

*Hint*: The reduction is quite similar to 3SAT $\leq_p$ 0/1-IPROG: Given a 3CNF formula $\phi$ with $M$ clauses and $N$ variables, generate a MULTI-KNAPSACK instance with $n = 2N$ items, i.e., one for every literal, and $v_i^p, w_i^q \in \{0, 1\}$ for $i \in [n], p \in [M + N], q \in [N]$. An truth assignment of $\phi$ should correspond to the selection of those literals which evaluate to true.

(e) Give a reduction 3SAT $\leq_p$ KNAPSACK .

*Hint*: Start from your reduction of 3SAT to MULTI-KNAPSACK and set $w_i := v_i := v_i^1 \ldots v_i^{M+N}$ for $i \in [2N]$ and $W := V := 1^N 3^M$ with all strings interpreted as numbers in *decimal* representation. A satisfying assignment should then yield a selection of total weight/value in $[1^N 1^M, 1^N 3^M]$. Introduce $2M$ additional items which allow to extend every selection induced by a satisfying assignment to a solution of the KNAPSACK instance.

**Exercise 3.7**

We define SUDOKU to be the following problem: You are given a $n^2 \times n^2$ grid where every entry is either blank or contains a numbers from $\{1, 2, \ldots, n^2\}$. The goal is to decided whether the remaining blank entries of the grid can be labeled by numbers from $\{1, 2, \ldots, n^2\}$ in such a way that every number of $\{1, 2, \ldots, n^2\}$ appears exactly once in (i) every row, (ii) every column, and (iii) in each of the $n^2$ subgrids.

- Give a reduction SUDOKU $\leq_p$ SAT.

In particular, apply your reduction to the following SUDOKU instance:

| | | | |
|---|---|---|---|
| 1 | | 2 | |
| | | | 4 |
| | 3 | | |
| | | 1 | |

*Remark*: One can show that SUDOKU is also **NP**-complete. The adventurous might like to attempt this!