Course 02249, DTU Compute
Carsten Witt

Computationally Hard Problems – Fall 2016
Assignment 2

Date: 06.09.2016, **Due date**: 12.09.2016, 21:00

The following exercises are **not** mandatory:

Exercise 2.1: Prove Proposition 2.14 in the lecture notes, which is also repeated below, and determine the constants c_{p} and c_{e}. You may assume that the running time for input size ℓ is ℓ^a for a positive constant $a > 0$ in the case of polynomial running time, and b^ℓ for a positive constant $b > 1$ in the case of exponential running time.

Proposition 2.14: Suppose an algorithm has been implemented on some machine. With the current hardware, this can solve problems up to size n in a fixed time t_0. Now suppose that the speed of the hardware is doubled. If the running time of the algorithm is polynomial then there is a constant $c_{\mathrm{p}} > 1$ such that one can now solve problems of size $c_{\mathrm{p}} \cdot n$ in time t_0. If the running time of the algorithm is exponential then there is a constant $c_{\mathrm{e}} > 0$ such that one can now solve problems of size $c_{\mathrm{e}} + n$ in time t_0.

End of Exercise 1

Exercise 2.2: Show how to convert a decision algorithm A_{d} for the following problem into a polynomial-time optimization algorithm A_{o}. The input to the problem consists of an undirected graph $G = (V, E)$ and a positive integer k. The decision algorithm A_{d} answers YES if there is a set $V' \subseteq V$ of cardinality k such that $\forall\, v, w \in V' \colon \{v, w\} \notin E$. This problem is called INDEPENDENTSET since there are no edges between the vertices in V' .
We now want to solve the optimization problem, that is, we are only given the undirected graph G and want to find in it an independent set V' of maximum cardinality.

a) Describe an algorithm A_{o} which solves the optimization problem as described above. The running time of A_{o} has to be polynomial in the input size and A_{o} may make calls to A_{d}. Recall that such a call counts as one step.

b) Argue that your algorithm is correct.

c) Show the running time of the algorithm.

End of Exercise 2

Continued on next page.

Exercise 2.3: Consider the following problem.

Problem: [MinimumTestSet]
Input: You want to test an electronic circuit. The circuit can have n different errors, $e_1, \ldots, e_n$. You can perform m different tests $T_1, \ldots, T_m$. Every test can detect some, but not necessarily all, errors.
Output for the optimizing version: A minimum set of tests that detects all possible errors.

We suggest the following algorithm for selecting a minimum test set: Select a test T_i which detects a maximum number of errors. Continue by selecting tests which detect a maximum number of errors, which have not already been detected by previously selected tests.
Prove or disprove that the output of this algorithm is a minimum test set.

End of Exercise 3

The following exercise is **mandatory**:

Exercise 2.4: We consider the following problem.

Problem: [Refutation] Given is a disjunctive form consisting of k monomials $m_1, \ldots, m_k$ over n boolean variables $x_1, \ldots, x_n$ (as on Exercise Sheet 1). The task is to decide if there is a truth assignment to the variables such that the truth value of the disjunctive form is *false.*

You are given a decision algorithm A_d that solves this problem, i. e., for each instance to Refutation, A_d answers YES if there is an assignment that makes the truth value of the disjunctive form *false*; otherwise it answers NO.

a) Describe an algorithm A_o which solves the optimization problem, that is, which finds a truth assignment making the disjunctive form *false* if one exists. The running time has to be polynomial in the input size and A_o may make calls to A_d. Such calls count as one basic computational step.

b) Argue that your algorithm is correct.

c) Prove that the running time of the algorithm is bounded from above by a polynomial. Any polynomial is sufficient; you need not look for a polynomial of minimal degree. Recall that a call to A_d counts one step.

Note: The input to Refutation is a disjunctive form of monomials over n boolean variables, nothing else. In particular, a legal input cannot specify specific settings of variables.

End of Exercise 4