Algorithms and Complexity
2015
Mästarprov 2: Complexity

Mästarprov 2 should be solved individually in written form and presented orally. No collaboration is allowed.

Written solutions should be handed in latest on **Tuesday, April 21th 17.00,** to Johan in written or printed form (personally or his mailbox). Be sure to save a copy of your solutions. Mästarprov 1 is a mandatory and rated part of the course. The test consists of four tasks. The test is roughly graded as follows: Two task correctly solved give an E. Three tasks correctly solved give a C and all tasks correctly solved give an A. The report should be written in English.

1. Super connectors

We have a computer network. We fix a number C and say that a computer is a *super connector* if it has at least C connections. (Connections are edges in the network.) A group S of super connectors is a set of super connectors all connected to each other. We can formulate a decision problem like this: Given a network in form of a graph G and integers C, K, is there a group of size K of super connectors in G? Show that this problem is NP-Complete by reducing the known problem CLIQUE to this problem.

2. A restricted version of CNF-SAT

In CNF-SAT we have a formula ϕ on CNF-form. The clauses can be arbitrarily long (i.e. the don't have to be of length 3). We know that this is an NP-Complete problem. We might believe that part of the problem is that a variable can occur many times in a formula. Let us formulate a problem called CNF$_3$. In this problem we have a formula with each variable occuring *at most* 3 times. The problem is then to decide if a formula on this form is satisfiable or not. It turns out that even this problem is NP-Complete. Your task is to show this by giving a method for translating any CNF-formula ϕ to a CNF$_3$-formula ϕ' such that ϕ is satisfiable if and only if ϕ' is satisfiable.

Here are some hints:

1. You will have to introduce some *new* variables in ϕ.
2. Two variables x, y are equivalent if $(x \vee \bar{y}) \wedge (\bar{x} \vee y)$ is true.

3. Rectangle puzzle

In this problem we will study a very simple type of puzzle. Imagine that we have a rectangular grid of squares of size 1×1. In this grid we have a rectangle of size $a \times b$ where a and b are positive integers. The rectangle is oriented to fit the grid and therefore consists of ab squares. We also have n small rectangles (pieces of the puzzle) of sizes $h_1 \times b_1, h_2 \times b_2, ..., h_n \times b_n$. We want to place all these pieces into the big rectangle so that they are aligned with the grid and do not overlap. If $h_i \neq b_i$ there are two possible orientations of rectangle i and so on. Is it possible to fit in all the pieces, i.e., lay the puzzle? Note that the pieces do not have to cover all of the big rectangle. We can formulate this as the problem RECTANGLE PUZZLE:

Input: Positive integers a, b. A set of n pairs $\{(h_1, b_1), (h_2, b_2), ..., (h_n, b_n)\}$, all with positive integer values.

Goal: Given a rectangle with corners $(0,0), (0,b), (a,0), (a,b)$ in the coordinate plane, is it possible to place n small rectangles of sizes h_i, b_i into the rectangle so that the corners of the rectangles are placed in integer positions and do not overlap? (The small rectangles can be rotated $90°$.)

Decide if this problem can be solved efficiently or not.

4. Finding vertex covers

In this problem we will use the technique of reducing construction problems to optimization problems.(Compare with examples from lecture notes and exercise problems.) What we will do is to assume that we have an algorithm F such that it, given a graph G, computes the size of a minimal vertex cover of G. Write an algorithm F' such that F' uses calls to F and finds an optimal vertex cover. If the time-complexity of F is $T(n)$ where n is the size of the input, then the time-complexity of G should be $O(p(n)T(n))$ where p is a polynomial of low degree.