KTH Computer Science and Communication

Homework II, Complexity Theory Fall 2011

Due on September 15 at 15.15, i.e. at the beginning of the lecture. The general rules on homework solutions available at the course home-page apply. In particular, discussions of ideas in groups of up to at most two people are allowed but solutions should be written down individually.

Some of the problems below are "classical" and hence their solutions is probably posted on the Internet. It is not allowed to use such solutions in any way. The order of the problems is "random" and hence do not expect that the lowest numbered problems are the easiest.

Any corrections or clarifications on this problem set will be posted under "homework" on the course home-page http://www.csc.kth.se/utbildning/kth/kurser/DD2446/kplx11/uppgifter.

1 (10p) This is information gathering problem and should be solved *individually* As discussed in class the standard way to prove that a computational problem, A, requires large amounts of time is to reduce the $T(n)$-bounded halting problem for some quickly growing function T to A. An equivalently to say this is that A is hard for $DTime(T(n))$, the complexity class of problems being solvable in time $T(n)$. Find two examples of problems being proved to take at least time $2^{\Omega(n)}$ by this method. The problems should be a little bit different from the problems we discussed in class and in particular they should not mention "Turing machine" or a similar notion. State the problems you find together with lower bound T on the amount of time T they require. You should supply a reference (text-book or original publication) to the claimed result but there is no need to repeat the proof.

2 (15p) Prove that $P \neq \text{SPACE}(O(n))$, i.e. that the set of languages accepted in polynomial time is not equal to the set of languages accepted in linear space, i.e. by Turing-machines that operate using at most $O(n)$ space on inputs of length n.

Hint: As we do not know which of the two is the bigger class or if they are incomparable, the proof has to be rather indirect. Assume equality of the two classes, use each assumption twice together with two hierarchy theorems.

Another ingredient of the proof which here is crucial but normally plays little role is the coding of the inputs. Padding in various forms is an artificial way to change the length of the input. In particular one can require that in an n-bit input, each bit is repeated n times. The classes P and $\text{SPACE}(O(n))$ behave very differently with this type of padding.

It is here important to note that, as dicussed in the notes available on the course home page, there is a hierarchy in for space complexity which is similar (and in fact more fine-grained) to the hierarchy for time complexity.

3 This problem is about coding of inputs. Let us consider problems of integers and graphs. Remember that P is defined as the set of functions that can be computed in time polynomial in the length of the inputs.

3a (5p) Let us consider three ways to code integers. In binary, decimal, or unary notation. The first two are hopefully well known and in the last the integer n is coded by a sequence of n ones. Determine to what extent these three codings give rise to the same functions in P.

Let us consider two ways to code undirected graphs on m nodes. The first coding is given by the number m in binary, followed by $\binom{m}{2}$ bits x_{ij} for $1 \leq i < j \leq m$ where $x_{ij} = 1$ if and only if the edge (i, j) is present in the graph. The second encoding is given by a number s in binary and a list $(i_1, j_1), (i_2, j_2) \ldots (i_s, j_s)$ of all s edges.

3b (5p) Prove that if we focus on graphs where each vertex has at least on edge adjacent to it, the two encodings give the same graph-functions in P.

3c (5p) What happens if we allow graphs with very few edges? Do we get the same set of functions in P?