KTH Computer Science and Communication

Homework I, Complexity Theory 2008

Due on April 22 at 8.15, 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 three people but solutions should be written down individually. There is a bonus problem outside the standard total of 100 points. Points on this problem are counted as normal points. Try to be mathematically correct in your arguments.

1 (12p) Design explicitly a Turing machine that computes the product of two numbers written in binary and gives the answer also in binary. The details of the machine (details of input and output coding, the number of tapes, working alphabet, etc) are up to you and you can make any choice to ease your programming effort, but your choices should be formally specified.

2 (18p) Let us study properties of Turing machines. In this problem we are interested only in the input/output behavior of the machine. We consider "not halting" to be a particular type of output behavior. Properties that fall under this label are "Outputting 7 on the input 5", "Halting on all inputs", "Giving the output 0 on all input strings that end with a 1". Properties such that "Having 23 states" or "Any transition possible from state q_0 is also possible from q_1" are clearly not of this type.

A property P is nontrivial if there is both a machine that has the property and a machine that does not have the property.

2a (13p) Given any nontrivial property P of the above type show that it is non-recursive to decide whether a given machine has property P.

2b (5p) Is it true that for each property of the given type the set of machines it describes is recursively enumerable?

3 (20p) Prove that if a language, L, is recognized by a Turing machine in time $T(n)$ and space $S(n)$, then there is a different Turing machine that recognizes L in time $T(n)/2 + O(n)$ and space $S(n)/2 + O(n)$.

Hint: Code several symbols of the old machine as one symbol of new machine.

4 (20p) Suppose you are given a Boolean formula with $\wedge$, $\vee$, $\neg$ and Boolean variables. Prove (10p) that it is possible to convert the formula to CNF (i.e. as a conjunction of disjunctions) in the same variables but give an example (5p) where the blow-up in size must be exponential.

Explain (5p) how this relates to the fact that CNF-Sat is NP-complete and hence satisfiability of an arbitrary formula can be reduced to satisfiability of a formula in CNF in polynomial time.

5 (10p) Normally we pose NP-problems as decision problems, i.e. given a formula φ we ask if it is satisfiable. Usually, if the formula is satisfiable we also want to find an assignment satisfying φ. This is called the "search problem". This is the problem of returning a satisfying assignment in the case when φ is satisfiable and the statement "not satisfiable" when this is the case.

Prove (5p) that these two problems are equivalent in that if we can solve one in polynomial time then we can solve the other.

Is this a unique property for satisfiability or is the corresponding property true for any NP-complete problem? In other words is it true (5p) for any NP-complete language A that deciding $x \in A$ is polynomial time equivalent to finding a witness to this fact?

6 (23p) The aim of this problem is to show that two-tape Turing machines are significantly more efficient then one-tape Turing machines. Consider the language of palindromes

$$\{ww^R | w \in \Sigma^*\},$$

where R denotes reversal.

6a Show that this language can be recognized in linear time on a two-tape Turing machine.

6b Show that this language requires time $\Omega(n^2)$ to be recognized by a one-tape Turing machine.

Hint: Look at inputs of the form $y0^{2k}y^R$ for k about $n/4$. They must all be accepted. If the computation is too quick try to construct an accepting computation of an input that should not be accepted, by splicing together accepting computations.

7 **Bonus problem** (25p) This problems analyzes an algorithm that solves 3Sat on instances with n variables in time c^n where $c < 2$. The algorithm works as follows on a 3-CNF formula φ.

1. Pick a random assignment α.
2. Repeat the following $10n$ times. If α does not satisfy φ pick a clause not satisfied by α and randomly pick one of the three variables in α and change its value.
3. Repeat from 1.

Analyze this algorithm, ignoring polynomial factors!

Hint: Analyze the distance from α to a satisfying assignment x_0. Let α_0 be the assignment picked in step 1, and d be Hamming distance. The probability of landing in x_0 is

$$\sum_{d_0} Pr[\text{success} | d(x_0, \alpha_0) = d_0] Pr[d(x_0, \alpha_0) = d_0]$$

and, as we are ignoring polynomial factors, the best d_0 gives the answer.