Mateusz Biegański
mb385162

Computational Complexity - Homework 3

4 minutes of delay, I would be grateful if I weren't punished for that :):

Problem 3.1: Prove that there exists a deterministic Turing machine with oracle for SAT that works in polynomial time, and that given a positive integer n finds its decomposition into prime numbers: $n = p^{\alpha_1} \cdot ... \cdot p^{\alpha_k}$ where $p_1 < ... < p_k$ are prime numbers, and $\alpha_1, ..., \alpha_k$ are positive integers.

Solution:

Let's observe, that for given $n, n \in \mathbb{N}$, SAT problem can be used to find it's decomposition into positive numbers x, y, such that $n = x \cdot y$. That construction is described in Appendix.

Now, we simply do the following: Start an algorithm from n; check whether it is prime, using AKS algorithm. If it is, finish, otherwise find it's $n = x \cdot y$ decomposition, using algorithm described in Appendix. Now, do the same for both it's x, y factors.

This way, we obtain decomposition $n = p_1 \cdot p_2 \cdot ... \cdot p_m, \forall i \in \{1, ..., m\}$ p_i is prime. Now, to find all α_i numbers from problem description we just sort all p_i and group it.

Whole solution is polynomial-time in obvious way, thus we proved such TM exists.

Appendix:

Let's consider problem of determining decomposition of given $n \in \mathbb{N}$ into factors x, y, such that $x, y \geq 2, n = x \cdot y$ using SAT. We can simply omit that part, because SAT is NP-complete and finding multiplication decomposition is in NP, but I put that part for consistency, rather as a draft.

Let's say that n is given in binary representation $z_k, ..., z_2, z_1$, z_i representing each digit, 1 index means the lowest significant bit. We will be using variables $x_k, ..., x_1$ and $y_k, ..., y_1$ to mark representation of x, y numbers. Let's observe, that multiplication of binary numbers consists of addition of maybe shifted numbers, which is easy to obtain if we implement addition.

$z_i \implies XOR(x_i, y_i, z_{i-1})$, assuming $z_0 = 0$ implements addition.

Now, for multiplication 10101 * 1101, we do 10101(0) + 10101(2) + 10101(3) (number in parentheses means positions to shift left) = 10101000 + 1010100 + 10101 Remark, that we take advantage of fact, that XOR is implementable in boolean formulas language.

Now, when we do have machine determining whether formula is satisfiable, we can create polynomial-time algorithm that finds proper values (x, y in our case), which just substitutes $x_i = 1$, and check whether formula is still satisfiable, if no, it say's "then x_i must be 0" and do the same for x_{i+1}.

We need to remember that we only need to find factors ≥ 2, so in our formula we must have additional clauses that forces any of $x_1, ..., x_{k-1}$ being 1, same with $y_1, ..., y_{k-1}$.

Whole description above provides us algorithm that for given n finds such x, y, that $n = x \cdot y$.

Problem 3.2:

For a language $L \subseteq \{0,1\}*$, let $B(L,r) = \{u | \exists v \in L, d(u,v) \leq r\}$, where $d(u,v)$ is the Hamming distance,

$$d(u,v) = \begin{cases} |\{i : v_i \neq u_i\}| \ , & |u| = |v| \\ \infty, & \text{otherwise} \end{cases}$$

Show that for each $L \subseteq \{0,1\}$ and each $r \in \mathbb{N}$

3.2 a): if $L \in RP$ then $B(L,r) \in RP$ Let M be RP-TM that recognizes L.

To show $B(L,r)$ is in RP, for given word w we must construct RP-TM. Let's observe, that we seek for words $u \in L$, such that $d(w,u) \leq r$. Thus words u are copies of w, with $i \in \{0,1,...,r\}$ bits changed. There are $\sum_{i=0}^{i=r} \binom{n}{i} \leq n^{r+1} = poly(n)$ such words. What we do is simply run M for each word from above, waiting for TRUE answer. Let's observe that it is enough for RP-TM construction - if $w \notin B(L,r)$, we never return TRUE; if it is in $B(L,r)$, in particular $\exists u \ d(u,w) \leq r$, and we run M for that u, obtaining required $1/2$ success chance.

3.2 b): if $L \in coRP$ then $B(L,r) \in coRP$

$L \in coRP$ implies, that there exists coRP-TM, recognizing L. Let's name it M. We must show, that there exists coRP-TM recognizing $B(L,r)$. The computation is presented below:

Given word $w, |w| = n$ let's mark $W = \sum_{i=0}^{i \leq r} \binom{n}{i}$ (number of all considered words).

First generate all words v, such that $d(w,v) \leq r$. Now, run M on all such generated $v's$, accept if any was accepting. But, what is important, on each word run M machine W times (once is not enough; it will be needed to prove correctness). We accept if all of W runs are accepting.

Let's observe, that for words from $B(L,r)$ at least one word v is from L, thus we will

accept with probability equals to 1.

A bit more problematic is case, when $w \notin B(L, r)$. M rejects w with probability $p \geq 1/2$. We need to compute probability, that $\forall v, d(w, v) \leq r$ M rejects v W times. Probability of single v to be rejected during W-times-check is $1 - (1 - p)^W$ (it is enough one run to be rejecting), thus rejection of all $v's$ is $(1 - (1 - p)^W)^W \geq (1 - 2^{-W})^W \geq 1/2$, because $p \geq 1/2$.

Algorithm above belongs to coRP.