Introduction to Theoretical Computer Science

Tutorial Sheet – Week 7 Tutorials

These questions are fairly straightforward; the first one involves some mathematical equation shuffling.

(1) The basic claim was that polynomial problems are 'easy', and non-polynomial problems are hard. Consider $f(n) = n^{10^{10}}$, and $g(n) = 10^{n/10^{10}}$. Show that $f(n) \in o(g(n))$. (Recall this means that $\forall \epsilon > 0.\exists n_0.\forall n > n_0.|f(n)| \leq \epsilon|g(n)|$.) (Hint: take logs, and remember that you only have to care about large enough n.) Where does g catch up with f?

For the enthusiast: Where does the statement $f(n) \in o(g(n))$ fit in the arithmetical hierarchy that we discussed unofficially? (Trick question!)

(2) On slide 36, we defined the class P in terms of polynomially bounded machines. Explain how to implement this definition. That is, given a register machine M (taking input R in R_0 as usual), explain how to construct a machine M' which takes inputs R and k, and behaves like M except that it halts after $(\lg R)^k$ steps of M's execution.

(3) Show that the Halting problem is not NP-complete. (This is obvious … but can you prove it?)

This is a reasonably tricky algorithm design problem.

(4) 2-SAT is the following problem: given a set of boolean variables X_i, and a formula $\phi = \bigwedge_{1 \leq j \leq n}(\alpha_j \vee \beta_j)$, where each α_j, β_j is either a variable or a negated variable, is there a satisfying assignment for ϕ?

Show that 2-SAT is polynomial (unlike SAT). (Quite difficult. Hint: look for two clauses that contain a variable and its negation (e.g. $(X \vee Y)$ and $(Z \vee \neg Y)$), merge them into a single clause, and add it to the formula.)