Complexity Theory
Easter 2000
Suggested Exercises 3

1. Recall that a Boolean expression is in 3CNF if it is a conjunction of clauses, each of which contains at most 3 literals. The language 3SAT is the set of strings that are *satisfiable* Boolean expressions in 3CNF. In the lectures, an argument was sketched to show that 3SAT is NP-complete. Complete this argument by showing the following:

   > There is a polynomial time computable function that will map any CNF Boolean expression ϕ to a 3CNF expression θ, such that θ is satisfiable if, and only if, ϕ is.

2. We say that a propositional formula ϕ is in 2CNF if it is a conjunction of clauses, each of which contains exactly 2 literals. The point of this problem is to show that the satisfiability problem for formulas in 2CNF can be solved by a polynomial time algorithm.

   First note that any clause with 2 literals can be written as an implication in exactly two ways. For instance $(p \vee \neg q)$ is equivalent to $(q \rightarrow p)$ and $(\neg p \rightarrow \neg q)$, and $(p \vee q)$ is equivalent to $(\neg p \rightarrow q)$ and $(\neg q \rightarrow p)$.

   For any formula ϕ, define the directed graph G_ϕ to be the graph whose set of vertices is the set of all literals that occur in ϕ, and in which there is an edge from literal x to literal y if, and only if, the implication $(x \rightarrow y)$ is equivalent to one of the clauses in ϕ.

   (a) If ϕ has n variables and m clauses, give an upper bound on the number of vertices and edges in G_ϕ.

   (b) Show that ϕ is *unsatisfiable* if, and only if, there is a literal x such that there is a path in G_ϕ from x to $\neg x$ *and* a path from $\neg x$ to x.

   (c) Give an algorithm for verifying that a graph G_ϕ satisfies the property stated in (b) above. What is the complexity of your algorithm?

   (d) From (c) deduce that there is a polynomial time algorithm for testing whether or not a 2CNF propositional formula is satisfiable.

   (e) Why does this idea not work if we have 3 literals per clause?

3. A clause (i.e. a disjunction of literals) is called a *Horn* clause, if it contains *exactly one* positive literal. Such a clause can be written as an implication: $(x \vee (\neg y) \vee (\neg w) \vee (\neg z))$ is equivalent to $((y \wedge w \wedge z) \rightarrow x))$. $\mathsf{HORNSAT}$ is the problem of deciding whether a given Boolean expression that is a conjunction of Horn clauses is satisfiable.

   (a) Show that there is a polynomial time algorithm for solving $\mathsf{HORNSAT}$. (Hint: if a variable is the only literal in a clause, it must be set to `true`; if all the negative variables in a clause have been set to `true`, then the positive one must also be set to `true`. Continue this procedure until a contradiction is reached or a satisfying truth assignment is found).

   (b) In the proof of the NP-completeness of SAT it was shown how to construct, for every nondeterministic machine M, integer k and string x a Boolean expression ϕ which is satisfiable if, and only if, M accepts x within n^k steps. Show that, if M is deterministic, than ϕ can be chosen to be a conjunction of Horn clauses.

   (c) Conclude from (b) that the problem $\mathsf{HORNSAT}$ is P-complete.

4. $\mathsf{POLYLOGSPACE}$ is the complexity class

$$\bigcup_k \mathsf{SPACE}((\log n)^k).$$

   (a) Show that, for any k, if $A \in \mathsf{SPACE}((\log n)^k)$ and $B \leq_L A$, then $B \in \mathsf{SPACE}((\log n)^k)$.

   (b) Show that there are no $\mathsf{POLYLOGSPACE}$-complete problems with respect to $\leq_L$. (Hint: use (a) and the space hierarchy theorem).

   (c) Which of the following might be true: $\mathsf{P} \subseteq \mathsf{POLYLOGSPACE}$, $\mathsf{P} \supseteq \mathsf{POLYLOGSPACE}$, $\mathsf{P} = \mathsf{POLYLOGSPACE}$?