1. In the lecture, a proof was sketched showing a $\Omega(n \log n)$ lower bound on the complexity of the sorting problem. It was also stated that a similar analysis could be used to establish the same bound for the Travelling Salesman Problem. Give a detailed sketch of such an argument. Can you think of a way to improve the lower bound?

2. Say we are given a set $V = \{v_1, \ldots, v_n\}$ of vertices and a cost matrix $c : V \times V \to \mathbb{N}$. For a set $S \subseteq V$, let $t_{S,i}$ denote the cost of the shortest path that starts at $v_1$, visits all vertices in $S$ and ends at $v_i$. Describe a dynamic programming algorithm that computes $t_{S,i}$ for all sets $S$ and all $i$. Show that your algorithm can be used to solve the Travelling Salesman Problem in time $O(n^2 2^n)$.

3. Consider the language $\text{Unary-Prime}$ in the one letter alphabet $\{a\}$ defined by $\text{Unary-Prime} = \{a^n \mid n \text{ is prime}\}$. Show that this language is in $\mathsf{P}$.

4. Suppose $S \subseteq \mathbb{N}$ is a set of natural numbers and consider the language $\text{Unary}-S$ in the one letter alphabet $\{a\}$ defined by $\text{Unary}-S = \{a^n \mid n \in S\}$, and the language $\text{Binary}-S$ in the two letter alphabet $\{0, 1\}$ consisting of those strings starting with a 1 which are the binary representation of a number in $S$. Show that if $\text{Unary}-S$ is in $\mathsf{P}$ then $\text{Binary}-S$ is in $\mathsf{TIME}(2^{cn})$ for some constant $c$.

5. We say that a propositional formula $\phi$ is in $2\text{CNF}$ 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 $2\text{CNF}$ 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 $(p \lor \neg q)$ is equivalent to $(q \rightarrow p)$ and $(\neg p \rightarrow \neg q)$, and $(p \lor q)$ is equivalent to $(\neg p \rightarrow q)$ and $(\neg q \rightarrow p)$.

For any formula $\phi$, define the directed graph $G_\phi$ to be the graph whose set of vertices is the set of all literals that occur in $\phi$, 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 $\phi$.

(a) If $\phi$ has $n$ variables and $m$ clauses, give an upper bound on the number of vertices and edges in $G_\phi$. 

1
(b) Show that \( \phi \) is unsatisfiable if, and only if, there is a literal \( x \) such that there is a path in \( G_\phi \) from \( x \) to \( \neg x \) and a path from \( \neg x \) to \( x \).

(c) Give an algorithm for verifying that a graph \( G_\phi \) 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?

6. A clause (i.e. a disjunction of literals) is called a Horn clause, if it contains at most one positive literal. Such a clause can be written as an implication: \((x \lor (\neg y) \lor (\neg w) \lor (\neg z))\) is equivalent to \(((y \land w \land z) \rightarrow x)\). 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 HORNSAT. (Hint: if a variable is the only literal in a clause, it must be set to \text{true}; if all the negative variables in a clause have been set to \text{true}, then the positive one must also be set to \text{true}. Continue this procedure until a contradiction is reached or a satisfying truth assignment is found).

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

(c) Conclude from (b) that the problem HORNSAT is \( \text{P} \)-complete under \( \text{L} \)-reductions.

7. We define the complexity class of quasi-polynomial-time problems Quasi-\( \text{P} \) by:

\[
\text{Quasi-}\text{P} = \bigcup_{k=1}^{\infty} \text{Time}(n^{(\log n)^k}).
\]

Show that if \( L_1 \preceq_p L_2 \) and \( L_2 \in \text{Quasi-} \text{P} \), then \( L_1 \in \text{Quasi-} \text{P} \).

8. In general \( k \)-colourability is the problem of deciding, given a graph \( G = (V, E) \), whether there is a colouring \( \chi : V \rightarrow \{1, \ldots, k\} \) of the vertices such that if \( (u, v) \in E \), then \( \chi(u) \neq \chi(v) \). That is, adjacent vertices do not have the same colour.

(a) Show that there is a polynomial time algorithm for solving 2-colourability.

(b) Show that, for each \( k \), \( k \)-colourability is reducible to \( k + 1 \)-colourability. What can you conclude from this about the complexity of 4-colourability?