Complexity Theory

Supervision 1

1. Algorithms and problems

1. a) Explain the formal connections between the notions of *characteristic function*, *predicate*, *decision problem*, *subset* and *language*.

   b) What is the difference between a Turing machine *accepting* vs. *deciding* a language L? How does this distinction relate to the difference between *recursively enumerable* and *decidable* languages? (Note: instead of *accepting*, *decidable* and *recursively enumerable*, you will also often see the terms *recognising*, *recursive* and *semidecidable*, respectively).

   c) What set-theoretic object (what kind of function or relation) is implemented by a Turing machine accepting vs. deciding a language?

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 an index $i \in [1..n]$ and a subset $S \subseteq V$, let $T(S, i)$ denote the cost of the shortest path that starts at v_1, and visits all vertices in S, with the last stop being $v_i \in S$. Describe a dynamic programming algorithm that computes $T(S, i)$ for all sets $S \subseteq V$ and all $i \leq |V|$. Show that your algorithm can be used to solve the Travelling Salesman Problem in time $O(n^2 2^n)$.

3. The lectures define Turing machines to have a transition function of type $\delta : (Q \times \Sigma) \to (Q \cup \{ \mathtt{acc}, \mathtt{rej} \}) \times \Sigma \times D$, where $D = \{ L, R, S \}$ is the set of directions that the tape head can move in (left, right, stationary). In other literature you might find definitions that have $D = \{ L, R \}$, allowing only two directions and requiring that the tape head move at each transition. Can such a Turing machine simulate the one described in lectures? Is the complexity class P affected by this distinction?

2. Polynomial-time problems

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

2. Suppose $S \subseteq \mathbb{N}$ is a subset of natural numbers and consider the language Unary-S in the one-letter alphabet $\{ a \}$ defined by Unary-$S = \{ a^n \mid n \in S \}$, and the language 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 Unary-S is in P, then Binary-S is in TIME(2^{cn}) for some constant c.

3. We say that a propositional formula φ is in 2CNF if it is a conjunction of clauses, each of which contains exactly two 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 two literals can be written as an implication in exactly two ways. For instance $(P \vee \neg Q)$ is equivalent to $(Q \Longrightarrow P)$ and $(\neg P \Longrightarrow \neg Q)$, and $(P \vee Q)$ is equivalent

to $(\neg P \implies Q)$ and $(\neg Q \implies P)$. 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 P to literal Q if, and only if, the implication $P \implies Q$ 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 P such that there is a path in G_φ from P to $\neg P$ *and* a path from $\neg P$ to P.

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 2CNF-SAT is in P.

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

4. 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: $X \vee \neg Y \vee \neg W \vee \neg Z$ is equivalent to $(Y \wedge W \wedge Z \implies X)$. HORNSAT is the problem of deciding whether a given Boolean expression that is a conjunction of Horn clauses is satisfiable.

   Show that there is a polynomial time algorithm for solving HORNSAT.

3. Reductions

1. We define the complexity class of *quasi-polynomial-time* problems Quasi-P by:

$$\text{Quasi-P} = \bigcup_{k=1}^{\infty} \text{Time}\left(n^{(\log n)^k}\right)$$

   Show that if $L_1 \leq_P L_2$ and $L_2 \in$ Quasi-P, then $L_1 \in$ Quasi-P.

2. In general, k-colourability is the problem of deciding, given a graph $G = (V, E)$, whether there is a colouring $\chi : V \to \{1, \ldots, k\}$ 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. Does this, together with part (a), mean that 3-colourability is also in P?