Complexity Theory

Exercises

2021

Contents

1. Algorithms and problems 1
2. Polynomial-time problems 1
3. Reductions 2
4. NP-completeness 2
5. NP-complete problems 3

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 $\mathrm{TIME}(2^{cn})$ 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 $(\neg P \Longrightarrow Q)$ and $(\neg Q \Longrightarrow P)$. 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 $P \Longrightarrow Q$ 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 $(Y \wedge W \wedge Z \Longrightarrow X)$. 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?

4. NP-completeness

1. Show that the identity function is a poly-time reduction, and composition of poly-time reductions is a poly-time reduction.

2. A problem in NP is called NP-*intermediate* if it is neither in P nor NP-complete.

   a) Are there any problems that are known to be NP-intermediate?

   b) Research and briefly summarise *Ladner's theorem*.

3. Suppose that a language $L_1 \subseteq \Sigma_1^*$ is polynomial-time reducible to a language $L_2 \subseteq \Sigma_2^*$ with the underlying function $f : L_1 \leq_P L_2$. Prove or disprove the following claims, or state if the answer is unknown and explain why:

a) If $L_2 \leq_P L_1$, then f is a bijection.

b) If f is a bijection, then $L_2 \leq_P L_1$.

c) If f is a bijection, then L_2 is in NP.

d) If f is a bijection and L_1 is in NP, then L_2 is in NP.

e) If L_1 is NP-complete, then $L_2 \leq_P L_1$.

f) If L_2 is NP-complete, then $L_2 \leq_P L_1$.

5. NP-complete problems

1. Given a graph $G = (V, E)$, a set $C \subseteq V$ of vertices is called a *vertex cover* of G if, for each edge $(u, v) \in E$, either $u \in C$ or $v \in C$. That is, each edge has at least one end point in C. The decision problem V-COVER is defined as:

   Given a graph $G = (V, E)$, and an integer K, does G contain a vertex cover with K or fewer elements?

   a) Show a polynomial time reduction from IND to V-COVER.

   b) Use a) to argue that V-COVER is NP-complete.

2. The problem of *four-dimensional matching*, 4DM, is defined analogously with 3DM:

   Given four sets, W, X, Y and Z, each with n elements, and a set of quadruples $M \subseteq W \times X \times Y \times X$, is there a subset $M' \subseteq M$ such that each element of W, X, Y and Z appears in exactly one tuple in M'?

   Show that 4DM is NP-complete.

3. Given a graph $G = (V, E)$, a source vertex $s \in V$ and a target vertex $t \in V$, a *Hamiltonian path* from s to t in G is a path that begins at s, ends at t, and visits every vertex in V exactly once. We define the decision problem HamPath as:

   Given a graph $G = (V, E)$ does G contain a Hamiltonian path?

   a) Give a poly-time reduction from the Hamiltonian cycle problem to HamPath.

   b) Give a poly-time reduction from HamPath to the Hamiltonian cycle problem.

   c) Consider the following, modified statement of the Hamiltonian path problem:

      Given a graph $G = (V, E)$ and vertices $s, t \in V$, does G contain a Hamiltonian path from s to t?

      Explain how this differs from the problem above, and comment on whether your reductions in parts a) and b) can be simplified for this version.

4. We know from the Cook–Levin Theorem that every problem in NP is reducible to SAT. The proof worked for a general nondeterministic Turing-machine, but for some problems it is easy to give an explicit reduction. Describe how to obtain, for any graph $G = (V, E)$, a Boolean expression φ_G such that φ_G is satisfiable if and only if:

a) G is 3-colourable.

b) G contains a Hamiltonian cycle.

Hint: By analysing the search space of the problem, determine a collection of Boolean variables that can encode the relevant properties of the graph (cf. $S_{i,q}$, $T_{i,j,\sigma}$ and $H_{i,j}$ in the Cook–Levin Theorem proof). Give the constraints on the variables which are required to make the encoded graph an instance of the given problem (cf. expressions (1)-(7) in the CLT proof). Combine these with the constraints that would decide whether a potential instance is a member of the problem or not (cf. expression (8) in the CLT proof).

5. An instance of a *linear programming* problem consists of a set $X = \{ x_1, \ldots, x_n \}$ of variables and a set of constraints, each of the form

$$\sum_{1 \leq i \leq n} c_i x_i \leq b,$$

where each c_i and b is an integer.

The 0-1 Integer Linear Programming Feasibility problem 01-ILP is defined as follows:

Given an instance of a linear programming problem, determine whether there is an assignment of values from the set $\{ 0, 1 \}$ to the variables in X so that substituting these values in the constraints leads to all constraints being simultaneously satisfied.

Prove that this problem is NP-complete.

6. *Self-reducibility* refers to the property of some problems in $L \in \mathrm{NP}$, where the problem of finding a witness for the membership of an input x in L can be reduced to the decision problem for L. This question asks you to give such arguments in three specific instances.

a) Show that, given an oracle (i.e. a black box) for deciding whether a formula φ over a set of variables $\mathcal{V} = \{ x_1, x_2, \ldots, x_n \}$ is satisfiable, there is a polynomial-time algorithm that gives a variable assignment which satisfies a formula over $\mathcal{V}$.

b) Show that, given an oracle for deciding whether a given graph $G = (V, E)$ is Hamiltonian, there is a polynomial-time algorithm that, on input G, outputs a Hamiltonian cycle in G if one exists.

c) (Harder) Show that, given an oracle for deciding whether a given graph G is 3-colourable, there is a polynomial-type algorithm that, on input G, produces a valid 3-colouring of G if one exists.

Optional exercises

1. The problem E3SAT is defined as follows:

Given a set of clauses, each clause being a disjunction of exactly three distinct literals and containing exactly three distinct variables, determine whether it is satisfiable.

Prove that $\mathrm{E3SAT}$ is NP-complete. *Hint:* introduce new variables to the set by adding a tautological clause.

2. We use $x;0^n$ to denote the string that is obtained by concatenating the string x with a separator ; followed by n occurrences of 0. If $[M]$ represents the string encoding of a non-deterministic Turing machine M, show that the following language is NP-complete:

$$S = \{\,[M];x;0^n \mid M \text{ accepts } x \text{ in } n \text{ steps}\,\}$$

Hint: Rather than attempting a reduction from a particular NP-complete problem, it is easier to show this from first principles, i.e. construct a reduction for any NDTM M and polynomial bound p.