Chapter 2 : Time complexity

In this chapter we study some basic results on the time complexities of computational problems. We concentrate our attention mostly on polynomial time complexities, deterministic and nondeterministic.

2.1 The complexity classes P and NP

P is the class of languages accepted by poly-time DTMs, whereas NP is the class of languages accepted by poly-time NTMs. More specifically for input size n we define:

$$\mathrm{P} := \bigcup_{k\in\mathbb{N}} \mathrm{TIME}(n^k), \qquad \text{and} \qquad \mathrm{NP} := \bigcup_{k\in\mathbb{N}} \mathrm{NTIME}(n^k).$$

These classes are invariant for all computational models that are polynomially equivalent to the single-tape single-head TM (deterministic for P, nondeterministic for NP).

We use encodings $\langle \ldots \rangle$ of (a finite number of) objects into a string. We assume only *reasonable* encodings. Examples of reasonable encodings include the binary (or decimal or hexadecimal) encoding of positive integers, adjacency matrix representation of a graph and so on. The unary encoding of positive integers is not reasonable, since it demands unreasonable (exponential) amount of space compared to reasonable encodings (like binary).

2.1 Example (1) The language

$$\mathrm{PATH} := \{\langle G, s, t\rangle \mid \text{ There is a path from vertex } s \text{ to vertex } t \text{ in a directed graph } G\}$$

is in P. The proof is based on search algorithms (like BFS or DFS) in the graph.

(2) The language

$$\mathrm{RELPRIME} := \{\langle x, y\rangle \mid \text{ The positive integers } x \text{ and } y \text{ are relatively prime}\}$$

is also in P. The proof is based on the Euclidean division and gcd algorithms.

(3) Every context-free language is in P. For the proof one may use a dynamic programming algorithm for context-free grammars in Chomsky-normal form.

NP comprises precisely the languages that are verifiable in poly-time, i.e., that have succinct certificates.

2.2 Definition A verifier for a language $L \subseteq \Sigma^*$ is defined to be a DTM V, such that

$$L = \{\alpha \mid \langle \alpha, \beta\rangle \in \mathcal{L}(V) \text{ for some } \beta \in \Sigma^*\}.$$

The string β is a certificate for the membership of α in L. V is called a poly-time verifier, if V runs in poly-time in the size of α. For a poly-time verifier a certificate β for α must be of polynomial length in $|\alpha|$. Such certificates are called succinct certificates.

2.3 Theorem $A \in \mathrm{NP}$ if and only if A has a poly-time verifier.

2.4 Example The following languages are in NP:

$$\begin{aligned}
\text{HAMPATH} &:= \{\langle G,s,t\rangle \mid \text{There is a Hamiltonian path from vertex } s \text{ to vertex } t \text{ in the directed graph } G\}\\
\text{UHAMPATH} &:= \{\langle G,s,t\rangle \mid \text{There is a Hamiltonian path from vertex } s \text{ to vertex } t \text{ in the undirected graph } G\}\\
\text{CLIQUE} &:= \{\langle G,k\rangle \mid \text{The undirected graph } G \text{ has a } k\text{-clique}\}\\
\text{INDEP-SET} &:= \{\langle G,k\rangle \mid \text{The undirected graph } G \text{ has an independent set of size } k\}\\
\text{VERTEX-COVER} &:= \{\langle G,k\rangle \mid \text{The undirected graph } G \text{ has a vertex cover of size } k\}\\
\text{COMPOSITE} &:= \{\langle x\rangle \mid \text{The positive integer } x \text{ is composite}\}\\
\text{SUBSET-SUM} &:= \left\{\langle S,t\rangle \mid \text{There is a subset } T \text{ of the set } S \text{ with } t=\sum_{x\in T} x\right\}\\
\text{SAT} &:= \{\langle \phi\rangle \mid \phi \text{ is a satisfiable Boolean formula}\}\\
\text{3SAT} &:= \{\langle \phi\rangle \mid \phi \text{ is a satisfiable Boolean formula in 3-cnf}\}
\end{aligned}$$

Exercises for Section 2.1

1. Prove that the following languages are in P:

$$\begin{aligned}
\text{CONNECTED} &:= \{\langle G\rangle \mid G \text{ is a connected undirected graph}\}\\
\text{BIPARTITE} &:= \{\langle G\rangle \mid G \text{ is an undirected bipartite graph}\}\\
\text{TRIANGLE-FREE} &:= \{\langle G\rangle \mid G \text{ is a triangle-free undirected graph}\}
\end{aligned}$$

2. Show that the language $\text{GRAPHISO} := \{\langle G_1, G_2\rangle \mid \text{The undirected graphs } G_1 \text{ and } G_2 \text{ are isomorphic}\}$ is in NP.

**** 3.** Show that the language $\text{PRIME} := \{\langle x\rangle \mid x \text{ is a prime positive integer}\}$ is in NP. (**Remark:** Of course, we now know that $\text{PRIME} \in \text{P}$ and so in NP as well. An independent proof that $\text{PRIME} \in \text{NP}$ is not as easy as proving $\text{COMPOSITE} \in \text{NP}$. You may use the fact that $p \in \mathbb{N}$ is prime if and only if $\mathbb{Z}_p^*$ is cyclic and of order $p-1$. You may require the prime factorization of $p-1$ as part of a certificate for primality of p.)

*** 4. (a)** Demonstrate that the class P is closed under union, intersection, complement, concatenation and Kleene star.

(b) Prove that the class NP is closed under union, intersection, concatenation and Kleene star. (**Remark:** It is widely *believed* that NP is *not* closed under complement.)

2.2 NP-complete problems

Complete problems in a complexity class constitute the set of most difficult problems in the class. The definition of completeness in a particular class depends on suitable reduction algorithms between problems.

2.5 Definition A poly-time computable function $f : \Sigma^* \to \Sigma^*$ is a function for which a poly-time DTM M exists with the property that M, on input α, halts with $f(\alpha)$ (and nothing else) on its tape. A language A is poly-time reducible to a language B, if there exists a poly-time computable function f such that for every α we have $\alpha \in A$ if and only if $f(\alpha) \in B$. In this case we write $A \leqslant_P B$.

2.6 Definition A language L is called NP-complete, if:

(1) $L \in \text{NP}$ and

(2) $A \leqslant_P L$ for every $A \in \text{NP}$.

2.7 Theorem If $A \leqslant_P B$ and $B \in \text{P}$, then $A \in \text{P}$.

2.8 Theorem If A is NP-complete and $A \in \mathrm{P}$, then $\mathrm{P} = \mathrm{NP}$.

2.9 Theorem If A is NP-complete and $A \leqslant_P B$, then B is NP-complete too.

The last theorem suggests that once we have proved certain problems to be NP-complete, we can reduce these problems to other problems in NP to prove the NP-completeness of these new problems. That makes the class of NP-complete problems bigger and allows more possibilities for reductions to newer problems. However, the challenge is to prove from the scratch a first NP-complete problem to be so. This is done by the following pioneering work.

2.10 Theorem [Cook-Levin] SAT is NP-complete.

A simple continuation of the proof of Cook-Levin's theorem implies:

2.11 Theorem 3SAT is NP-complete.

2.12 Example All the problems introduced in Example 2.4 (except COMPOSITE) can be proved to be NP-complete. One may use the following reductions: $3\mathrm{SAT} \leqslant_P$ CLIQUE, CLIQUE $\leqslant_P$ INDEP-SET, INDEP-SET $\leqslant_P$ VERTEX-COVER, $3\mathrm{SAT} \leqslant_P$ HAMPATH, HAMPATH $\leqslant_P$ UHAMPATH and $3\mathrm{SAT} \leqslant_P$ SUBSET-SUM.

Exercises for Section 2.2

1. Show that if $\mathrm{P} = \mathrm{NP}$ and $L \in \mathrm{P} \setminus \{\emptyset, \Sigma^*\}$, then L is NP-complete.

2. Show that 2×2 windows are not sufficient in the proof of the Cook-Levin theorem.

** **3.** Let $2\mathrm{SAT} := \{\langle\phi\rangle \mid \phi \text{ is a satisfiable Boolean formula in 2-cnf}\}$. Show that $2\mathrm{SAT} \in \mathrm{P}$.

4. Prove that the following problems are NP-complete.

$$\begin{aligned} \text{HAMCYCLE} &:= \{\langle G\rangle \mid \text{There is a Hamiltonian cycle in the directed graph } G\} \\ \text{UHAMCYCLE} &:= \{\langle G\rangle \mid \text{There is a Hamiltonian cycle in the undirected graph } G\} \end{aligned}$$

5. [The traveling salesperson problem] Assume that a salesperson wishes to visit m cities with each city visited exactly once. Associated with each pair of cities a (positive) cost representing the overhead for inter-city travel (assumed symmetric with respect to the two cities). The objective of the salesperson is to reduce the total cost for the travel. Consider an undirected (complete) graph on m vertices with each vertex representing a city and with each edge labeled by the cost of the corresponding inter-city travel. The traveling salesperson problem can be reformulated as finding an (undirected) Hamiltonian cycle in the graph with the minimum sum of labels on the edges of the cycle. The following is the decision version of this problem:

$$\mathrm{TSP} := \{\langle G, k\rangle \mid G \text{ has a Hamiltonian cycle of (total) cost } \leqslant k\}.$$

Show that TSP is NP-complete.

6. Let $\text{DOUBLE-SAT} := \{\langle\phi\rangle \mid \phi \text{ has} \geqslant 2 \text{ satisfying assignments}\}$. Show that DOUBLE-SAT is NP-complete.

7. For a fixed k define $k\mathrm{COLOR} := \{\langle G\rangle \mid \text{ The undirected graph } G \text{ is } k\text{-colorable}\}$. Prove that:

* **(a)** $2\mathrm{COLOR} \in \mathrm{P}$.

** **(b)** $3\mathrm{COLOR}$ is NP-complete.

2.3 The class coNP

The complements of languages in NP constitute the class coNP:

$$\text{coNP} := \{L \mid \bar{L} \in \text{NP}\}.$$

Thus coNP consists of languages that have poly-time disqualifiers. It is not known if NP = coNP. The popular belief is that these two classes are different. One can define complete problems for the class coNP using poly-time reductions as in Definition 2.6. It turns out that:

2.13 Theorem L is NP-complete if and only if $\bar{L}$ is coNP-complete.

2.14 Theorem If P = NP, then NP = coNP.

However, one may have NP = coNP, even when P ≠ NP. We also have the following result:

2.15 Theorem If a coNP-complete problem is in NP, then NP = coNP.

2.16 Theorem $\text{P} \subseteq \text{NP} \cap \text{coNP}$.

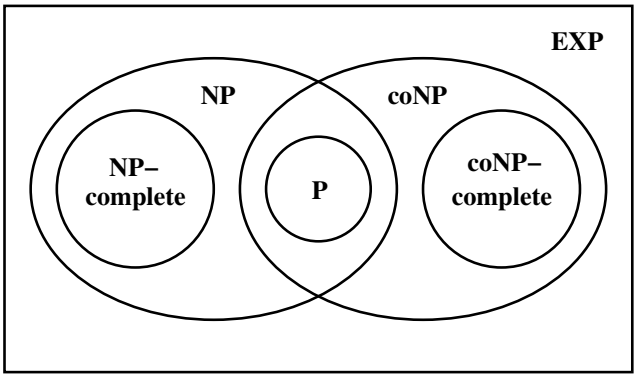
It is widely believed that $\text{P} \subsetneq \text{NP} \cap \text{coNP}$. The conjectured relationships among the complexity classes introduced so far are described in Figure 2.1. Here all containments are conjectured to be *proper*. We can only prove $\text{P} \subsetneq \text{EXP}$, where EXP is the deterministic exponential-time class defined for input size n as:

$$\text{EXP} := \bigcup_{k \in \mathbb{N}} \text{TIME}(2^{n^k}).$$

The exponential-time simulation of an NTM by a DTM implies:

2.17 Theorem $\text{NP} \subseteq \text{EXP}$.

Figure 2.1: Conjectured relationships among the time complexity classes



Exercises for Section 2.3

1. **(a)** Prove that the class EXP is closed under union, intersection, complement, concatenation and Kleene star.
(b) Deduce that $\text{coNP} \subseteq \text{EXP}$.

2. A strong nondeterministic Turing machine has three possible outcomes: "accept", "reject" and "notsure", and decides a language L as follows: If $\alpha \in L$, all branches of computation on input α end up in outcomes "accept" or "notsure", with at least one branch providing an "accept" answer. On the other hand, if $\alpha \notin L$, all possible branches of computation on α end up in "reject" or "notsure", with at least one "reject" answer. Deduce that L is decided by a poly-time strong NTM if and only if $L \in \mathrm{NP} \cap \mathrm{coNP}$.

*** 3.** Define the decision version of the integer factoring problem as:

$$\mathrm{FACTORING} := \{\langle n, k\rangle \mid n \in \mathbb{N} \text{ has a factor } d \text{ with } 1 < d < k\}.$$

Show that $\mathrm{FACTORING} \in \mathrm{NP} \cap \mathrm{coNP}$. (**Hint:** For the proof of $\mathrm{FACTORING} \in \mathrm{coNP}$ you may assume that $\mathrm{PRIME} \in \mathrm{NP}$.)

**** 4.** The nondeterministic exponential-time class NEXP is defined as:

$$\mathrm{NEXP} := \bigcup_{k \in \mathbb{N}} \mathrm{NTIME}(2^{n^k}),$$

where n is the input size. Clearly, $\mathrm{EXP} \subseteq \mathrm{NEXP}$. It is not known if (but popularly believed that) this containment is proper. Prove that if $\mathrm{P} = \mathrm{NP}$, then $\mathrm{EXP} = \mathrm{NEXP}$. (**Hint:** Let $L \in \mathrm{NEXP}$ and N an NTM that decides L in time 2^{n^k} for some $k \in \mathbb{N}$. Define the language $L' := \{\alpha \sqcup^{2^{|\alpha|^k} - |\alpha|} \mid \alpha \in L\}$, where $\sqcup$ is a 'quasiblank' symbol. First show that $L' \in \mathrm{NP}$. By hypothesis, there is a DTM M' that decides L' in poly-time. Convert M' to an exponential-time DTM M to accept L.)

2.4 Function problems

So far we have considered only decision problems, i.e., problems that have yes/no answers. In practice, however, we deal with problems that *compute* some quantities from the input string. For example, SAT talks about the satisfiability of a Boolean function. We now frame the problem: if ϕ is satisfiable, compute a satisfying assignment for ϕ. We call this new problem FSAT. It may turn out that these "compute something" problems are more difficult than the corresponding decision versions. However, we have strong results that justify concentrating on studies of decision problems only.

2.18 Theorem FSAT can be solved in (deterministic) poly-time if and only if $\mathrm{SAT} \in \mathrm{P}$.

We now formally introduce function problems.

2.19 Definition Let $\rho \subseteq \Sigma^* \times \Sigma^*$ be a relation (on Σ^*). ρ is said to be polynomially decidable, if $\{\langle \alpha, \beta\rangle \mid (\alpha, \beta) \in \rho\}$ is in P. ρ is said to be polynomially balanced, if whenever $(\alpha, \beta) \in \rho$, we have $|\beta| \leqslant |\alpha|^k$ for some constant $k \in \mathbb{N}$ (depending on L but independent of α or β).

2.20 Theorem Let $L \subseteq \Sigma^*$. Then $L \in \mathrm{NP}$ if and only if there exists a polynomially decidable and polynomially balanced relation $\rho_L \subseteq \Sigma^* \times \Sigma^*$ with $L = \{\alpha \mid (\alpha, \beta) \in \rho_L \text{ for some } \beta\}$.

2.21 Definition Let $L \in \mathrm{NP}$ and ρ_L be as in Theorem 2.20. The function problem associated with L is the following: If $\alpha \in L$, compute some β with $(\alpha, \beta) \in \rho_L$; if $\alpha \notin L$, return "no". (Of course, such a problem does not compute a "function" in the mathematical sense, since there may be several β for a given α.) Let us denote by $\mathrm{F}L$ the function problem associated with $L \in \mathrm{NP}$. We can now define the class:

$$\mathrm{FNP} := \{\mathrm{F}L \mid L \in \mathrm{NP}\}.$$

FP is the subclass of FNP consisting of function problems that can be solved in deterministic poly-time.

It is not known (but believed) that FP $\subsetneqq$ FNP. The problem FSAT of Theorem 2.18 is in FNP, but unlikely to be in FP. One can define reduction between problems in FNP and also complete problems for the class FNP. It can be shown that FSAT is FNP-complete. This implies that:

2.22 Theorem FP = FNP if and only if P = NP.

Some function problems have language Σ^* and are not interesting in terms of their decidability. Such problems are typically meant for computing some quantity and this computation makes sense for every possible input. In this case, the relation ρ_L is important (and not the trivial language L).

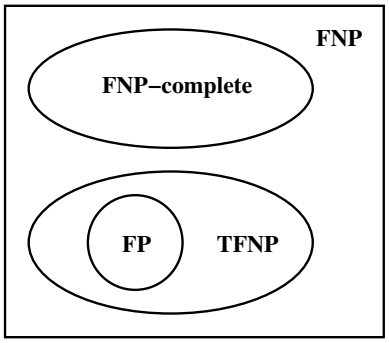
2.23 Definition A problem ρ_L in FNP is called total, if for every $\alpha \in \Sigma^*$ there exists β with $(\alpha, \beta) \in \rho_L$. TFNP is the subclass of FNP consisting of total function problems.

2.24 Example (1) The total version TFACTORING of the integer factoring problem deals with the computation of the prime factorization of the positive integer represented by the input α. Note that every positive integer has a (unique) prime factorization. Thus the problem of deciding if a factorization exists is not important; what is important is to compute it.

(2) The HAPPYNET problem on an undirected graph $G = (V, E)$ is defined as follows. The vertices of G are thought of as human beings and each edge $(u, v) \in E$ is labeled by an integer (positive or negative) weight $w(u, v)$ indicating the extent to which the vertices u and v like (or dislike) one another. A *state* is a map $\sigma : V \to \{1, -1\}$. A vertex $u \in V$ is said to be *happy* in a state σ, if $\sigma(u) \sum_{(u,v)\in E} w(u, v)\sigma(v) \geqslant 0$. For any undirected graph G there exists a state in which *every* vertex is happy. The HAPPYNET problem computes one such state given the graph G.

Problems in TFNP are believed to be easier than FNP-complete. The conjectured relationships among the function problems are depicted in Figure 2.2, where all containments are believed to be proper.

Figure 2.2: Conjectured relationships among the complexity classes for function problems



Exercises for Section 2.4

**** 1.** Consider the total version TTSP of TSP (Exercise 2.2.5), in which an optimal tour of the salesperson is to be computed. Demonstrate that if TSP $\in$ P, then TTSP has a poly-time algorithm.

*** 2.** Consider the total version TFACTORING of the integer factoring problem (Exercise 2.3.3), in which the prime factorization of the input is to be computed. Deduce that if FACTORING $\in$ P, then TFACTORING has a poly-time algorithm.

3. Let $a_1, \ldots, a_n$ be n positive integers with $\sum_{i=1}^{n} a_i < 2^n - 1$. Show that there exist *disjoint* nonempty subsets S and T of $\{a_1, \ldots, a_n\}$ with $\sum_{x\in S} x = \sum_{y\in T} y$. The problem of computing such subsets S and T is thus in TFNP. This problem has no known poly-time algorithm.