Lesson 1: Basic complexity classes

Theme: Review of some introductory material.

1 The big-Oh notations

Let $\mathbb{N}$ denote the set of natural numbers $\{0, 1, 2, \ldots\}$. Let f and g be functions from $\mathbb{N}$ to $\mathbb{N}$.

- $f = O(g)$ means that there is c and n_0 such that for every $n \geqslant n_0$, $f(n) \leqslant c \cdot g(n)$.
  It is usually phrased as "there is c such that for (all) sufficiently large n," $f(n) \leqslant c \cdot g(n)$.
- $f = \Omega(g)$ means $g = O(f)$.
- $f = \Theta(g)$ means $g = O(f)$ and $f = O(g)$.
- $f = o(g)$ means for every $c > 0$, $f(n) \leqslant c \cdot g(n)$ for sufficiently large n.
  Equivalently, $f = o(g)$ means $f = O(g)$ and $g \neq O(f)$.
  Another equivalent definition is $f = o(g)$ means $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.
- $f = \omega(g)$ means $g = o(f)$.

To emphasize the input parameter, we will write $f(n) = O(g(n))$. The same for the Ω, o, ω notations. We also write $f(n) = \mathsf{poly}(n)$ to denote that $f(n) = c \cdot n^k$ for some c and $k \geqslant 1$.

Throughout the course, for an integer $n \geqslant 0$, we will denote by $\lfloor n \rfloor$ the binary representation of n. Likewise, $\lfloor G \rfloor$ the binary encoding of a graph G. In general, we write $\lfloor X \rfloor$ to denote the encoding/representation of an object X as a binary string, i.e., a 0-1 string. To avoid clutter, we often write X instead of $\lfloor X \rfloor$.

We usually use Σ to denote a finite input alphabet. Often $\Sigma = \{0, 1\}$. Recall also that for a word $w \in \Sigma^*$, $|w|$ denotes the length of w. For a DTM/NTM $\mathcal{M}$, we write $L(\mathcal{M})$ to denote the language $\{w : \mathcal{M} \text{ accepts } w\}$.

We often view a language $L \subseteq \Sigma^*$ as a boolean function, i.e., $L : \Sigma^* \to \{\mathsf{true}, \mathsf{false}\}$, where $L(x) = \mathsf{true}$ if and only if $x \in L$, for every $x \in \Sigma^*$.

2 Time complexity

Definition 1.1 Let $\mathcal{M}$ be a DTM/NTM, $w \in \Sigma^*$, $t \in \mathbb{N}$ and let $f : \mathbb{N} \to \mathbb{N}$ be a function.

- $\mathcal{M}$ *decides w in time t (or, in t steps)*, if every run of $\mathcal{M}$ on w has length at most t. That is, for every run of $\mathcal{M}$ on w:
  $$C_0 \ \vdash\ C_1 \ \vdash\ \cdots \ \vdash\ C_m \qquad\qquad \text{where } C_m \text{ is a halting configuration,}$$
  we have $m \leqslant t$.
- $\mathcal{M}$ *runs in time* $O(f(n))$, if there is $c > 0$ such that for sufficiently long word w, $\mathcal{M}$ decides w in time $c \cdot f(|w|)$.
- $\mathcal{M}$ *decides/accepts a language L in time* $O(f(n))$, if $L(\mathcal{M}) = L$ and $\mathcal{M}$ runs in time $O(f(n))$.
- $\textsc{Dtime}[f(n)] \stackrel{\mathsf{def}}{=} \{L : \text{there is a DTM } \mathcal{M} \text{ that decides } L \text{ in time } O(f(n))\}$.

- $\textsc{Ntime}[f(n)] \stackrel{\mathsf{def}}{=} \{L : \text{there is an NTM } \mathcal{M} \text{ that decides } L \text{ in time } O(f(n))\}$.

Note that Definition 1.1 applies in similar manner for both DTM and NTM. The only difference is that a DTM has one run for each input word w, whereas NTM can have many runs for each input word w.

We say that $\mathcal{M}$ runs in *polynomial* and *exponential time*, if there is $f(n) = \mathsf{poly}(n)$ such that $\mathcal{M}$ runs in time $O(f(n))$ and $O(2^{f(n)})$, respectively. In this case we also say that $\mathcal{M}$ is a polynomial/exponential time TM.

The following are some of the important classes in complexity theory.

$$\begin{aligned}
\mathbf{P} &\stackrel{\mathsf{def}}{=} \bigcup_{f(n)=\mathsf{poly}(n)} \textsc{Dtime}[f(n)] \\
\mathbf{NP} &\stackrel{\mathsf{def}}{=} \bigcup_{f(n)=\mathsf{poly}(n)} \textsc{Ntime}[f(n)] \\
\mathbf{coNP} &\stackrel{\mathsf{def}}{=} \{L : \Sigma^* - L \in \mathbf{NP}\} \\
\mathbf{EXP} &\stackrel{\mathsf{def}}{=} \bigcup_{f(n)=\mathsf{poly}(n)} \textsc{Dtime}[2^{f(n)}] \\
\mathbf{NEXP} &\stackrel{\mathsf{def}}{=} \bigcup_{f(n)=\mathsf{poly}(n)} \textsc{Ntime}[2^{f(n)}] \\
\mathbf{coNEXP} &\stackrel{\mathsf{def}}{=} \{L : \Sigma^* - L \in \mathbf{NEXP}\}
\end{aligned}$$

Theorem 1.2 (Padding theorem) *If* $\mathbf{NP} = \mathbf{P}$, *then* $\mathbf{NEXP} = \mathbf{EXP}$.
Likewise, if $\mathbf{NP} = \mathbf{coNP}$, *then* $\mathbf{NEXP} = \mathbf{coNEXP}$.

Proof. We will only prove the first statement, i.e., "if $\mathbf{NP} = \mathbf{P}$, then $\mathbf{NEXP} = \mathbf{EXP}$."

Suppose $\mathbf{NP} = \mathbf{P}$. We will show that $\mathbf{NEXP} \subseteq \mathbf{EXP}$. Let $L \in \mathbf{NEXP}$. Let $\mathcal{M}$ be an NTM that decides L in time $2^{p(n)}$, where $p(n) = \mathsf{poly}(n)$. Consider the following language:

$$L' \stackrel{\mathsf{def}}{=} \{w0\underbrace{11\cdots 1}_{m} \ : \ w \in L \text{ and } m = 2^{p(|w|)}\}$$

We will first show that $L' \in \mathbf{NP}$. Consider the following algorithm that we denote by Algorithm 1.

Algorithm 1

Input: $u \in \Sigma^*$.
Task: Decide if $u \in L'$.

Check if u is of the form $w0\underbrace{11\cdots 1}_{m}$ for some m. and that $m = 2^{p(|w|)}$.
If not, REJECT. Otherwise, continue.
Run $\mathcal{M}$ on w.
ACCEPT if and only if $\mathcal{M}$ accepts w.

Since $\mathcal{M}$ is non-deterministic, Algorithm 1 is also non-deterministic. We can show that Algorithm 1 runs in polynomial time (in the length of the input u). Thus, $L' \in \mathbf{NP}$. By our assumption that $\mathbf{NP} = \mathbf{P}$, we obtain that $L' \in \mathbf{P}$. Let $\mathcal{M}'$ be a DTM that decides L' in polynomial time.

To show that $L \in \mathbf{EXP}$, consider the following algorithm that we denote by Algorithm 2.

Algorithm 2

Input: $w \in \Sigma^*$.

Task: Decide if $w \in L$.

1: Compute $m \stackrel{\text{def}}{=} 2^{p(|w|)}$.

2: Run $\mathcal{M}'$ on input $w01^m$.

3: ACCEPT if and only if $\mathcal{M}$ accepts w.

Note that by the definition of L', Algorithm 2 decides the language L. It is deterministic because $\mathcal{M}'$ is deterministic. Moreover, it runs in exponential time in the length of the input word w. Therefore, $L \in \mathbf{EXP}$, as desired. This completes the proof that $\mathbf{NP} = \mathbf{P}$ implies $\mathbf{NEXP} = \mathbf{EXP}$. ▮

3 Space complexity

Definition 1.3 Let $\mathcal{M}$ be a DTM/NTM, $w \in \Sigma^*$, $t \in \mathbb{N}$ and let $f : \mathbb{N} \to \mathbb{N}$ be a function.

- $\mathcal{M}$ decides w in t space (or, using t cells/space), if for every run of $\mathcal{M}$ on w:

  $$C_0 \vdash C_1 \vdash \cdots \vdash C_N \qquad \text{where } C_N \text{ is an accepting/rejecting configuration,}$$

  the length $|C_i| \leqslant t$, for each $i = 0, \ldots, N$.

- $\mathcal{M}$ *uses* $O(f(n))$ *space*, if there is $c > 0$ such that for sufficiently long word w, $\mathcal{M}$ decides w using $c \cdot f(|w|)$ space.

- $\mathcal{M}$ *decides/accepts a language* L *in space* $O(f(n))$, if $L(\mathcal{M}) = L$ and $\mathcal{M}$ uses $O(f(n))$ space.

- $\text{DSPACE}[f(n)] \stackrel{\text{def}}{=} \{L : \text{there is a DTM } \mathcal{M} \text{ that decides } L \text{ using } O(f(n)) \text{ space}\}$.

- $\text{NSPACE}[f(n)] \stackrel{\text{def}}{=} \{L : \text{there is an NTM } \mathcal{M} \text{ that decides } L \text{ using } O(f(n)) \text{ space}\}$.

Again, note that the notion of $\mathcal{M}$ uses space $O(f(n))$ is the same for DTM and NTM. The only difference is that a DTM has only one run for each input word w, whereas NTM can have many runs for each input word w. In both cases, we can only say that $\mathcal{M}$ uses space $O(f(n))$, if for each input word w, for every run of $\mathcal{M}$ on w, the length of each configuration in the run is always $\leqslant cf(|w|)$.

We say that $\mathcal{M}$ uses *polynomial* and *exponential* space, if there is $f(n) = \mathsf{poly}(n)$ such that $\mathcal{M}$ runs in time $O(f(n))$ and $O(2^{f(n)})$, respectively. In this case we also say that $\mathcal{M}$ is a polynomial/exponential space TM. The following are some of the important classes in complexity theory.

$$\mathbf{PSPACE} \stackrel{\text{def}}{=} \bigcup_{f(n)=\mathsf{poly}(n)} \text{DSPACE}[f(n)]$$

$$\mathbf{EXPSPACE} \stackrel{\text{def}}{=} \bigcup_{f(n)=\mathsf{poly}(n)} \text{DSPACE}[2^{f(n)}]$$

4 Logarithmic space complexity

Another interesting classes are **L** and **NL**. We say that a language L is in **L**, if there is a 2-tape DTM $\mathcal{M}$ that decides L and a constant $c > 0$ such that for every input word w:

- The first tape always contains only the input word w, i.e., $\mathcal{M}$ never changes the content of the first tape.
- $\mathcal{M}$ uses $c \cdot \log(|w|)$ space in its second tape.

Likewise, we say that a language L is in **NL**, if there is a 2-tape NTM $\mathcal{M}$ that decides L such that the above two conditions are satisfied.

5 Some classic complexity results

Obviously, we have $\mathbf{L} \subseteq \mathbf{NL}$, $\mathbf{P} \subseteq \mathbf{NP}$, and $\mathbf{PSPACE} \subseteq \mathbf{NPSPACE}$.

Proposition 1.4

- $\mathbf{L} \subseteq \mathbf{P}$.
- $\mathbf{NP} \subseteq \mathbf{PSPACE}$.

Deterministic/non-deterministic time/space hierarchy theorem states that for every $k \geqslant 1$, the following holds.

$$\textsc{Dtime}[n^k] \subsetneq \textsc{Dtime}[n^{k+1}] \qquad \textsc{Dspace}[n^k] \subsetneq \textsc{Dspace}[n^{k+1}]$$
$$\textsc{Ntime}[n^k] \subsetneq \textsc{Ntime}[n^{k+1}] \qquad \textsc{Nspace}[n^k] \subsetneq \textsc{Nspace}[n^{k+1}]$$

Some classic results in complexity theory are: (We will prove all these results later on.)

- $\mathbf{NL} \subseteq \mathbf{P}$.
- If $L \in \textsc{Nspace}[n^k]$, then $\Sigma^* - L \in \textsc{Nspace}[n^k]$.
- $\textsc{Nspace}[n^k] \subseteq \textsc{Dspace}[n^{2k}]$.

The third bullet is the reason why we only have the class **PSPACE**.

Combining all these inclusions together, we obtain:

$$\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE}$$

From the deterministic/non-deterministic space hierarchy, it is also known that $\mathbf{L} \subsetneq \mathbf{PSPACE}$ and $\mathbf{NL} \subsetneq \mathbf{PSPACE}$. So, we know that at least one of the inclusions must be strict, but we don't know which one.