Short lecture notes on computational complexity[1].

(Outline of lectures at the University of Montpellier, December 2016.)
Last update: 14.12.2016.

December 7.

1 Time hierarchy theorem

Definition. A function $f : \mathbb{N} \to \mathbb{N}$ is called ***time constructible***, if it is computable and, moreover, the mapping

$$\underbrace{11\ldots1}_{n} \longmapsto \underbrace{11\ldots1}_{f(n)}$$

can be computed in time $O(f(n))$.

Exercises 1.1. *Show that the functions n, $n\log n$, n^{10}, 2^n, 2^{2^n} are time-constructible.*

Theorem 1.1. *Let $f : \mathbb{N} \to \mathbb{N}$ be a time constructible function such that $f(n) \geq n$ (for all n), and $g : \mathbb{N} \to \mathbb{N}$ be a function such that $f(n)$ is much less than $g(n)$, e.g.,*

$$(f(n))^3 = o(g(n)).$$

Then there exists a language $L \subset \{0,1\}^*$ that belongs to $\mathrm{DTIME}[g(n)]$ *but not to* $\mathrm{DTIME}[f(n)]$.

(Proven in the class.)

Corollary. $\mathrm{P} \neq \mathrm{EXP}$.

Exercises 1.2. *Compare Theorem 1.1 (proven in the class) with time hierarchy theorems in [1, 2, 3, 4], with a weaker constraint on $f(n)$ and $g(n)$. Read the proof of the time hierarchy theorem in (at least) one of these sources. Beware of the definition of a time constructible function!*

2 Space complexity

Definition. Denote by $\mathrm{DSPACE}[f(n)]$ the class of all languages that can be recognized by a deterministic Turing machine that uses $O(f(n))$ cells on the working space for all input size n. We also use the notation

$$\mathrm{PSPACE} := \bigcup_{k=1}^{\infty} \mathrm{DSPACE}[n^k].$$

Similarly, we denote by $\mathrm{NSPACE}(f(n))$ the class of all languages that can be recognized by a nondeterministic Turing machine that uses $O(f(n))$ cells on the working space for all input size n (for all computation paths) and

$$\mathrm{NPSPACE} := \bigcup_{k=1}^{\infty} \mathrm{NSPACE}[n^k].$$

(the machine must have accepting computations for words in the language).

Definition. A function $f : \mathbb{N} \to \mathbb{N}$ is called ***space constructible***, if it is computable and, moreover, the mapping

$$\underbrace{11\ldots1}_{n} \longmapsto \underbrace{11\ldots1}_{f(n)}$$

can be computed using $O(f(n))$ cells on the working tape.

[1]The students following this course can address with questions to Andrei Romashchenko, `andrei.romashchenko@lirmm.fr`

Exercises 2.1. *Show that the functions* n, $n \log n$, n^{10}, 2^n, 2^{2^n} *are space-constructible.*

Theorem 2.1. *Let* $f : \mathbb{N} \to \mathbb{N}$ *be a space constructible function such that* $f(n) \geq n$ *(for all* n*), and* $g : \mathbb{N} \to \mathbb{N}$ *be a function such that* $f = o(g)$*. Then there exists a language* $L \subset \{0,1\}^*$ *that belongs to* DTIME$[g(n)]$ *but not to* DTIME$[f(n)]$*.*

(We did not prove this theorem in the class.)

Exercises 2.2. *Prove Theorem 2.1. (You can find a sketch of the proof of this theorem in [1, 3]. A more detailed proof of the space hierarchy theorem is given in [2] and [4]. Beware of subtle differences in the definition of a space constructible function!)*

Theorem 2.2. *[Savitch's theorem] For every space constructible function* $f : \mathbb{N} \to \mathbb{N}$ *such that* $f(n) \geq n$

$$\mathrm{NSPACE}[f(n)] \subset \mathrm{DSPACE}[(f(n))^2].$$

(Proven in the class. You can find a proof of this theorem in [1, 2, 3].)

Corollary. NPSPACE = PSPACE.

Exercises 2.3. *(a) Explain where we used the condition of* **space constructibility** *in the proof of Theorem 2.2. (b)* Prove that the space hierarchy theorem remains true even without the condition of space-constructibility of* $f(n)$*.*

Theorem 2.3. *The language* BFQ *(**true quantified Boolean formulas**) is* PSPACE*-complete.*

(Proven in the class. The proof of this theorem can be found in [1, 2, 3].)

3 Computations with an oracle

In the class we defined a Turing machine with an oracle (see [1, 2, 3]).

Definition 3.1. *A language* A *is **Turing-reducible** to a language* B *(denoted* $A \leq_T B$*), if there is an oracle Turing machine that machine that recognizes* A *when given* B *as an oracle. Such a machine is said to reduce* A *to* B

A language A *is* **polynomial-time reducible** *to a language* B *(denoted* $A \leq_T^p B$*), if there is an oracle Turing machine that reduces* A *to* B *and runs in polynomial time*

Some basic properties of the Turing reduction:

- $A \leq_T^p A$ for every A,
- $A \leq_T^p (\{0,1\}^* \setminus A)$ for every A,
- if $A \leq_T^p B$ and $B \leq_T^p C$, then $A \leq_T^p C$,
- if $A \leq_T^p B$ and $B \in \mathrm{P}$, then $A \in \mathrm{P}$,
- if $A \in \mathrm{P}$, then $A \leq_T^p B$ for all B,
- if $A \leq_m B$, then $A \leq_T^p B$.

Exercises 3.1. *Most theorem of the computability theory* relativize*, i.e., they remain true for Turing machines with any oracle. Prove the following properties:*

(a) For every oracle A *there exists a function* $f : \{0,1\}^* \to \{0,1\}^*$ *that is* not computable *with this oracle.*

(b) For all sets A and S, if a set S is enumerable with the oracle A and co-enumerable with the oracle A, then this set is decidable with the oracle A.

(c) For every oracle A we have $\mathrm{P}^A \subset \mathrm{NP}^A \subset \mathrm{PSPACE}^A \subset \mathrm{EXP}^A$.

(d) For every set A we have $\mathrm{PSPACE}^A = \mathrm{NPSPACE}^A$.

Hint: Verify that the standard proofs of these properties for computations without oracles can be adapted to the oracle machines.

Theorem 3.1. *There exists an oracle A such that $\mathrm{P}^A = \mathrm{NP}^A = \mathrm{PSPACE}^A$. In particular, these equalities hold for the oracle BFQ.*

(Proven in the class. A proof can be found in [1, 2, 3].)

December 14.

Theorem 3.2. *There exists an oracle B such that $\mathrm{P}^B \neq \mathrm{NP}^B$.*

(Proven in the class. A proof can be found in [1, 2, 3].)

Definition 3.2. *The class* coNP *consists of all languages L such that the complement of the language (i.e., the set $\{0,1\}^* \setminus L$) belongs to* NP*.*

Exercises 3.2. *Prove that* $\mathrm{coNP} \subset \mathrm{P}^{\mathrm{NP}}$ (P^{NP} *is defined as the union* $\bigcup\limits_{A \in \mathrm{NP}} \mathrm{NP}^A$).

Definition 3.3 (polynomial hierarchy)**.** *The classes* NP *and* coNP *are also denoted Σ_1^p and Π_1^p respectively. Further, for $n = 2, 3, \ldots$ we define by induction the classes Σ_n^p and Π_n^p as follows:*

$$\Sigma_n^p := \mathrm{NP}^{\Sigma_n^p} = \bigcup_{A \in \Sigma_n^p} \mathrm{NP}^A,$$

$$\Pi_n^p := \mathrm{coNP}^{\Pi_n^p} = \bigcup_{A \in \Pi_n^p} \mathrm{coNP}^A.$$

Exercises 3.3. *(a) Prove that for every n*

$$\Sigma_n^p \cup \Pi_n^p \subset \Sigma_{n+1}^p \cap \Pi_{n+1}^p.$$

(b) Prove that for every n

$$\Sigma_n^p \subset \mathrm{PSPACE} \quad \textit{and} \quad \Pi_n^p \subset \mathrm{PSPACE}.$$

Exercises 3.4 (optional)**.** *It is believed that $\Sigma_2^p \subsetneq \Sigma_3^p$, i.e., the difference $\Sigma_3^p \setminus \Sigma_2^p$ is not empty (though this conjecture remains unproven). Suggest a language A that could belongs to $\Sigma_3^p \setminus \Sigma_2^p$.*

Exercises 3.5. *(a) Prove that* $\mathrm{BPP} \subset \mathrm{EXP}$*. (b) Prove a stronger statement:* $\mathrm{BPP} \subset \mathrm{PSPACE}$*.*

Remark: It is known that $\mathrm{BPP} \subset \Sigma_2^p \cap \Pi_2^p$, but we do not prove this fact in the class.

4 Interactive proofs

Definition 4.1. *We say that a language L belongs to the class* IP *(L has* an interactive proof system*), if there exists a poly-time randomized Turing machine V* (Verifier) *and an function P* (Prover) *such that*

- *for each $x \in L$* $\text{Prob}[$*result of communication of V and P on input x* $= 1] > 2/3$, *and*
- *for each $x \notin L$, for every prover P'* $\text{Prob}[$*result of communication of V and P' on input x* $= 1] < 1/3$.

Remark: The constants 2/3 and 1/3 in the definition above can be changed to 0.99 and 0.01 respectively, or even to any reals $1 - \varepsilon$ and ε (for $\varepsilon < 1/2$). These modifications will not affect the defined class IP (all variants of the definition are equivalent to each other).

Some simple properties:

- BPP $\subset$ IP (the class BPP corresponds to the «interactive protocols» where a poly-time randomized Verifier does not ask any question to the Prover and performs all the computations without assistance).
- NP $\subset$ IP (the class NP corresponds to the «interactive protocols» where a poly-time Verifier does not use randomness).

Proposition 4.1. *The language*

$$nonIso := \{(G_1, G_2) \ : \ \textit{graphs } G_1 \textit{ and } G_2 \textit{ are } \textbf{\textit{not}} \textit{ isomorphic}\}$$

belongs to IP*.*

(Proven in the class. A proof can be found in [1, 2, 3].)

Exercises 4.1. *Prove that the language of quadratic non-residues*

$$NQR = \{(k, p) \mid p \textit{ is prime, and there is no } m \textit{ such that } m^2 = k \textit{ mod } p\}$$

belongs to IP*.*

Theorem 4.1. *(a)* IP $\subset$ EXP*. (b)* IP $\subset$ PSPACE*.*

(A sketch of the proof was discussed in the class. A proof can be found in [1, 2, 3].)

Theorem 4.2. PSPACE $\subset$ IP*.*

We did not prove this theorem in the class. See a proof in the [5] (very short!) or in [1, 2, 3].
Remark: The equality IP = PSPACE is not «relativizable», i.e., there exists an oracle A such that $\text{IP}^A \neq \text{PSPACE}^A$. We did not prove this fact in the class; the interested students can find a proof in [1].

Zero knowledge proof: In the class we discussed a protocol of a *zero-knowledge* interactive proof for the problem 3-*coloring of a graph* with physical gadgets (the assigned colors were hidden by cups, like in the *shell game*). We briefly discussed an «electronic» version if this protocol — without special physical gadgets, with a digital encryption of colors assigned to the vertices of the graph.

References

[1] Sylvain Perifel. *Complexité algorithmique.* Ellipses, 2014.

[2] Michael Sipser. *Introduction to the Theory of Computation.* Cengage Learning, 2012.

[3] Sanjeev Arora and Boaz Barak. *Computational complexity. A modern approach.* Cambridge University Press, 2009.

[4] Luca Trevisan. *Notes on Hierarchy Theorems.* https://people.eecs.berkeley.edu/∼luca/cs172/noteh.pdf

[5] Alexander Shen, *IP= SPACE: simplified proof.* Journal of the ACM (JACM) 39, no. 4 (1992): 878-880.

Further reading

[6] Lance Fortnow and Steve Homer. *A Short History of Computational Complexity.* Bulletin of the EATCS, 80, 2003, pp. 95-133.

[7] Lance Fortnow. *The status of the P versus NP problem.* Communications of the ACM, 52(9), 2009, pp. 78-86.

[8] Russell Impagliazzo. *A Personal View of Average Case Complexity.* Proceedings of Tenth Annual IEEE Conference Structure in Complexity Theory, 1995, pp. 134-147. (5 possible worlds of complexity)

[9] Scott Aaronson's Shtetl Optimized blog: *Reasons to believe.* (10 justifications for the belief that P $\neq$ NP)