Computational Complexity

Homework Sheet 3

Hand in via Canvas before May 14, 2024, at 23:59

`https://canvas.uva.nl/courses/42595/assignments/496117`

For this homework assignment, solve Exercises 1 and 2. In addition, choose exactly one of Exercises 3 and 4, and solve it. This adds up to a total of 10 points that you can (maximally) obtain.

Exercise 1 (3pt)**.** Define:

$$\mathsf{P/log} = \bigcup_{c,d \in \mathbb{N}} \mathrm{DTIME}(n^c)/(d \log n).$$

That is, $\mathsf{P/log}$ is the class of all languages that can be decided in polynomial time with $O(\log n)$ bits of advice. Prove that $\mathsf{SAT} \notin \mathsf{P/log}$, unless $\mathsf{P} = \mathsf{NP}$.

- *Hint:* iterate over all possible advice strings of length $O(\log n)$.
- *Hint:* you may assume that for any string x that represents a propositional formula φ and any truth assignment α to (some of) the variables of φ, one can in polynomial-time encode the formula $\varphi[\alpha]$ as a string x' that is of the same length as x—where $\varphi[\alpha]$ is obtained from φ by instantiating each variable z in the domain of α by $\alpha(z)$.

Exercise 2 (3pt)**.**

(a) Show that $\mathsf{RP} \subseteq \mathsf{NP}$.

(b) Show that $\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP}$.

- *Hint:* use Markov's inequality for showing that $\mathsf{ZPP} \subseteq \mathsf{RP} \cap \mathsf{coRP}$. If X is a non-negative random variable and $a > 0$, then:

$$\mathbb{P}(X \geq a) \leq \frac{\mathbb{E}(X)}{a}.$$

Definition 1. Consider the following two complexity classes $\mathsf{P}^{\mathsf{NP}[\log]}$ and $\mathsf{P}^{\mathsf{NP}}_{||}$:

- $\mathsf{P}^{\mathsf{NP}[\log]}$ is the class of all decision problems $L \subseteq \{0,1\}^*$ for which there exists a polynomial-time deterministic oracle TM $\mathbb{M}$ and an oracle language $O \in \mathsf{NP}$ such that $\mathbb{M}^O$ decides L, and a function $f(n) : \mathbb{N} \to \mathbb{N}$ that is $O(\log n)$ such that for each input $x \in \{0,1\}^*$, $\mathbb{M}^O(x)$ makes at most $f(|x|)$ queries to the oracle O.
- $\mathsf{P}^{\mathsf{NP}}_{||}$ is the class of all decision problems $L \subseteq \{0,1\}^*$ for which there exists a polynomial-time deterministic oracle TM $\mathbb{M}$ and an oracle language $O \in \mathsf{NP}$ such that $\mathbb{M}^O$ decides L, and for each input, $\mathbb{M}^O$ makes their queries to O *in parallel*.

  Making parallel oracle queries works as follows. The machine $\mathbb{M}$ may write an arbitrary number of oracle queries $q_1, \ldots, q_m \in \{0,1\}^*$ on the oracle tape, separated by a designated symbol `#` (i.e., $q_1\#q_2\#\cdots\#q_m$). Then, when the machine $\mathbb{M}$ enters the designated query state $q_{\text{query}} \in Q$, instead of transitioning into q_{yes} or q_{no} depending on the answer of the oracle query, the machine transitions into a designated state q_{done}, and the contents of the oracle tape are replaced by the string $b_1\#b_2\#\cdots\#b_m$ that represents the answers $b_1, \ldots, b_m \in \{0,1\}$ to the oracle queries (e.g., `0#1#1#0`), where $b_i = 1$ if and only if q_i is in the oracle language O. The machine $\mathbb{M}$ is only allowed

to make a single such combined query for each input $x \in \{0,1\}^*$. In other words, it must compute (and write down) all the oracle queries that it wants to make before getting an answer for any of them.

Exercise 3 (4pt)**.** Prove that $\mathsf{P}^{\mathsf{NP}[\log]} = \mathsf{P}_{||}^{\mathsf{NP}}$.

- *Hint:* for showing that $\mathsf{P}_{||}^{\mathsf{NP}} \subseteq \mathsf{P}^{\mathsf{NP}[\log]}$, choose an NP-complete oracle O, allowing you to solve different NP problems by querying this single oracle—using the fact that for each $L \in \mathsf{NP}$, there exists a polynomial-time reduction from L to O.

Exercise 4 (4pt)**.** Consider the following problem Overlapping-Sets. Inputs for this problem consist of a pair $(U, \mathcal{S})$, where U is a finite set (the *universe*) and $\mathcal{S} = \{S_1, \ldots, S_m\}$ is a set of subsets of U, i.e., $S_i \subseteq U$ for all $1 \leq i \leq m$. A subset $\mathcal{C} \subseteq \mathcal{S}$ is a solution if for **each** $S_i, S_j \in \mathcal{C}$ it holds that $S_i \cap S_j \neq \emptyset$.

Let $\rho < 1$. An algorithm A is called a *ρ-approximation algorithm for* Overlapping-Sets if for every input $(U, \mathcal{S})$, the algorithm outputs a solution $\mathcal{C}$ for $(U, \mathcal{S})$ of size at least $\rho \cdot \mu_{(U,\mathcal{S})}$, where $\mu_{(U,\mathcal{S})}$ is the size of the largest solution for $(U, \mathcal{S})$. A probabilistic algorithm A is called a *probabilistic ρ-approximation algorithm for* Overlapping-Sets if for every input $(U, \mathcal{S})$, the probability that the algorithm outputs a solution $\mathcal{C}$ for $(U, \mathcal{S})$ of size at least $\rho \cdot \mu_{(U,\mathcal{S})}$ is at least $2/3$.

(a) Show that the decision variant of Overlapping-Sets is NP-complete—that is, the problem where the inputs consist of $(U, \mathcal{S})$ and $k \in \mathbb{N}$, and the question is to decide if there exists a solution $\mathcal{C}$ for $(U, \mathcal{S})$ of size k.

(b) Show that if there exist a polynomial-time ρ-approximation algorithm for Overlapping-Sets for some $\rho < 1$, then $\mathsf{P} = \mathsf{NP}$.

- *Hint:* have a look at Section 11.4 of [1].

(c) Show that if there exist a polynomial-time probabilistic ρ-approximation algorithm[1] for Overlapping-Sets for some $\rho < 1$, then $\mathsf{RP} = \mathsf{NP}$.

- *Hint:* show that for any $\rho < 1$, it is NP-hard to decide if $\mu_{(U,\mathcal{S})} = k$, when given $(U, \mathcal{S})$ and k as input, even when one is promised that either $\mu_{(U,\mathcal{S})} = k$ or $\mu_{(U,\mathcal{S})} < \rho \cdot k$.

Remark 1. Answers will be graded on two criteria: they should (1) be correct and intelligent, and also (2) concise and to the point.

Remark 2. If you find a solution to one of the exercises in a paper or book, you can use this to inform your solution. Make sure that you write down the solution in your own words, conveying that you understand what is going on.

References

[1] Sanjeev Arora and Boaz Barak. *Computational Complexity – A Modern Approach.* Cambridge University Press, 2009.

[1] That is, a probabilistic algorithm A for which there exists a polynomial $p(n)$ such that the running time of A is bounded by $p(n)$, regardless of the random choices that it makes.