KTH Computer Science and Communication

DD2445 Complexity Theory: Problem Set 1

Due: Friday October 6, 2017, at 23:59 AoE. Submit your solutions as a PDF file by e-mail to `jakobn at kth dot se` with the subject line `Problem set 1: ⟨your full name⟩`. Name the PDF file `PS1_⟨YourFullName⟩.pdf` with your name written in CamelCase without blanks and in ASCII without national characters. State your name and e-mail address at the very top of the first page. Solutions should be written in LaTeX or some other math-aware typesetting system with reasonable margins on all sides (at least 2.5 cm). Please try to be precise and to the point in your solutions and refrain from vague statements. *Write so that a fellow student of yours can read, understand, and verify your solutions.* In addition to what is stated below, the general rules stated on the course webpage always apply.

Collaboration: Discussions of ideas in groups of two people are allowed—and indeed, encouraged—but you should always produce your solutions completely on your own, from start to finish, and you should understand all aspects of them fully. It is not allowed to write down draft solutions together and then just add the finishing touches individually. You should also clearly acknowledge any collaboration. State at the very top of the first page of your problem set solutions if you have been collaborating with someone and if so with whom. *Note that collaboration is on a per problem set basis, so you should not discuss different problems on the same problem set with different people.*

Reference material: Some of the problems are "classic" and hence it might be easy to find solutions on the Internet, in textbooks or in research papers. It is not allowed to use such material in any way unless explicitly stated otherwise. Anything said during the lectures or in the lecture notes should be fair game, though, unless you are specifically asked to show something that we claimed without proof in class. All definitions should be as given in class or in Arora-Barak and cannot be substituted by versions from other sources. It is hard to pin down 100% watertight formal rules on what all of this means—when in doubt, ask the main instructor.

About the problems: Some of the problems are meant to be quite challenging and you are not necessarily expected to solve all of them. A total score of around 100 points should be enough for grade E, 130 points for grade D, 160 points for grade C, 190 points for grade B, and 220 points for grade A on this problem set. Any corrections or clarifications will be given at `piazza.com/kth.se/fall2017/dd2445/` and any revised versions will be posted on the course webpage `www.csc.kth.se/DD2445/kplx17/`.

1 (10 p) In class, we defined NP to be the set of languages L with the following property: There is a polynomial-time (deterministic) Turing machine M and a polynomial p such that $x \in L$ holds if and only if there is a witness y of length *exactly* $p(|x|)$ for which $M(x, y) = 1$.

Show that we can relax this so that the witness y is of length *at most* $p(|x|)$, but might be shorter for some x. That is, prove formally that with this new definition we get exactly the same set of languages in NP. (This is not hard, but please be careful so that you do not run into problems with any annoying details.)

2 (20 p) A *legal k-colouring* of a graph $G = (V, E)$ is an assignment of colours $\{1, 2, \ldots, k\}$ to the vertices in V such that if $(u, v) \in E$ is an edge, then the colours of u and v are distinct. Let k-Colouring be the language consisting of graphs that have a legal k-colouring. Recall that we proved in class that 3-Colouring is NP-complete.

2a What is the complexity of 2-Colouring?

2b What is the complexity of 4-Colouring?

For full credit on each of these subproblems, provide either an explicit algorithm (for an upper bound) or a reduction from some problem proven NP-complete in chapter 2 in Arora-Barak or during the lectures.

3 (20 p) Consider the language

$$\textsc{SpaceBoundedTM} = \left\{ \langle M, x, 1^n \rangle \,\middle|\, M \text{ accepts } x \text{ in space } n \right\}$$

where M is a deterministic Turing machine and 1^n denotes a string of ones of length n (as usual). Prove that SpaceBoundedTM is PSPACE-complete from first principles (i.e., prove that SpaceBoundedTM is in PSPACE and that any other language in PSPACE reduces to it).

4 (20 p) We proved in class that there are oracles relative to which P and NP are equal by defining the language $\textsc{ExpCom} = \left\{ \langle M, x, 1^n \rangle \,\middle|\, M \text{ accepts } x \text{ within } 2^n \text{ steps} \right\}$ and showing that $\mathsf{P}^{\textsc{ExpCom}} = \mathsf{NP}^{\textsc{ExpCom}} = \mathsf{EXP}$. In this problem we want to understand how important (or unimportant) the exact details in the definition of ExpCom is for this result to hold.

4a Let $\textsc{ExpCom}' = \left\{ \langle M, x, 1^n \rangle \,\middle|\, M \text{ accepts } x \text{ within } n \text{ steps} \right\}$. Does it hold that $\mathsf{P}^{\textsc{ExpCom}'} = \mathsf{NP}^{\textsc{ExpCom}'} = \mathsf{EXP}$ hold? Modify the argument we gave in class to establish these equalities or explain where the proof fails.

4b Let $\textsc{ExpCom}'' = \left\{ \langle M, x, n \rangle \,\middle|\, M \text{ accepts } x \text{ within } 2^n \text{ steps} \right\}$ (where n in the input is a number given in binary). Does it hold that $\mathsf{P}^{\textsc{ExpCom}''} = \mathsf{NP}^{\textsc{ExpCom}''} = \mathsf{EXP}$? Adapt the proof given in class or explain where it fails.

5 (20 p) Let us say that a function $f : \{0,1\}^* \to \{0,1\}^*$ is *write-once logspace computable* if f can be computed by a Turing machine M that uses $O(\log n)$ space on its work tapes and whose output tape is *write-once*. By a write-once tape we mean a tape where at every time step M can either keep its head at the same position on the tape or write a symbol to it and move one location to the right, but M can never read from the tape or move left. The used cells on the write-once tape are not counted towards the space bound on M.

Prove that f is write-once logspace computable if and only if it is *implicitly logspace computable* as defined in class.

6 (30 p) A *vertex cover* of a graph $G = (V, E)$ is a subset $S \subseteq V$ of vertices such that for each edge $(u, v) \in E$ it holds that either $u \in S$ or $v \in S$. The language

$$\textsc{VertexCover} = \big\{\langle G, k\rangle \,\big|\, G \text{ has a vertex cover of size } k\big\}$$

is known to be NP-complete (and this fact can be assumed without proof).

Suppose that you are given a graph G and a parameter k and are told that the smallest vertex cover of G is either (i) of size at most k or (ii) of size at least $3k$. Show that there is a polynomial-time algorithm that can distinguish between the cases (i) and (ii). Can you do the same for a smaller constant than 3? If so, how small? Since VertexCover is NP-complete, why does this not show that P = NP?

7 (30 p) We proved in class that the language $\textsc{Path} = \big\{\langle G, s, t\rangle \,\big|\, \exists \text{path from } s \text{ to } t \text{ in } G\big\}$ is NL-complete. We also proved that NL = coNL, and, in particular, that it holds for the complement language $\overline{\textsc{Path}} = \big\{\langle G', s', t'\rangle \,\big|\, \neg\exists \text{path from } s' \text{ to } t' \text{ in } G'\big\}$ that $\overline{\textsc{Path}} \in \mathsf{NL}$.

But this means that there must exist an implicitly logspace computable function that takes a directed graph G' and two vertices $s', t' \in V(G')$ and outputs a directed graph G and two vertices $s, t \in V(G)$ such that there is *some path* from s to t in G if and only if there is *no path* from s' to t' in G'. Describe such a function and how to compute it.

You do not need to decribe every nut and bolt in the construction of G from G', but your description should contain enough details so that you could code it up in principle in your favourite high-level programming language (using well-defined subroutines that we also know can be coded up in principle).

8 (40 p) For a CNF formula F, let $\widetilde{F}$ denote the "canonical 3-CNF version" of F constructed as follows:

- Every clause $C \in F$ with at most 3 literals appears also in $\widetilde{F}$.
- For every clause $C \in F$ with more than 3 literals, say, $C = a_1 \vee a_2 \vee \cdots \vee a_k$, we add to $\widetilde{F}$ the set of clauses

  $$\{y_0,\ \overline{y}_0 \vee a_1 \vee y_1,\ \overline{y}_1 \vee a_2 \vee y_2,\ \ldots,\ \overline{y}_{k-1} \vee a_k \vee y_k,\ \overline{y}_k\}\ ,$$

  where $y_0, \ldots, y_k$ are new variables that appear only in this subset of clauses in $\widetilde{F}$.

8a (10 p) Prove that $\widetilde{F}$ is unsatisfiable if and only if F is unsatisfiable. (Please make sure to prove this claim in both directions, and to be careful with what you are assuming and what you are proving.)

8b (10 p) A CNF formula F is said to be *minimally unsatisfiable* if F is unsatisfiable but any formula $F' = F \setminus \{C\}$ obtained by removing an arbitrary clause C from F is always satisfiable. Prove that $\widetilde{F}$ is minimally unsatisfiable if and only if F is minimally unsatisfiable.

8c (20 p) Consider the language

$$\textsc{MinUnsat} = \{F \mid F \text{ is a minimally unsatisfiable CNF formula}\} \ .$$

What can you say about the computational complexity of deciding this language?

For this subproblem, and for this subproblem only, please look at textbooks, search in the research literature, or roam the internet to find an answer. As your solution to this subproblem, provide a brief but detailed discussion of your findings regarding MinUnsat together with solid references where one can look up any definitions and/or proofs (i.e., not a webpage but rather a research paper or possibly textbook). Note that you should still follow the problem set rules in that you are not allowed to collaborate or interact with anyone other than your partner on this problem set.

9 (50 p) Show that $\mathsf{P} \neq \mathsf{SPACE}(n^k)$ for any fixed $k \in \mathbb{N}^+$.
Hint: Use padding.

10 (60 p) Your task in this problem is to produce a complete, self-contained proof of (the vanilla version of) Ladner's theorem that we sketched in class. The goal is (at least) twofold:

- To have you work out the proof in detail and make sure you understand it.
- To train your skills in mathematical writing.

When you write the proof, you can freely consult the lecture notes as well as the relevant material in Arora-Barak, but you need to fill in all missing details. Also, the resulting write-up should stand on its own without referring to the lecture notes, Arora-Barak, or any other source.

Your write-up should be accessible to a student who has studied and fully understood the material at the level of *DD1352 Algorithms, Data Structures, and Complexity* but has not seen any more computational complexity than that (i.e., not more than the first three lectures of the current course, but you do not need to explain again the material in these lectures).

You are free to structure your proof as you like, except that all of the ingredients listed below should be explicitly addressed somewhere in your proof. (You can take care of them in whatever order you find appropriate, however. Please do not refer to the labelled subproblems in your write-up, since it should be a stand-alone text, but make sure your peer reviewer can find without problems where in your solution the different items are dealt with.)

10a Define

$$\textsc{Sat}_P = \left\{ \psi 01^{n^{P(n)}} \;\middle|\; \psi \in \textsc{CnfSat} \text{ and } n = |\psi| \right\}$$

as the language of satisfiable CNF formulas padded by a suitable number of ones at the end as determined by the function P, which we assume to be polynomial-time computable.

10b Prove that if $P(n) = \mathrm{O}(1)$, then $\textsc{Sat}_P$ is NP-complete.

10c Prove that if $P(n) = \Omega(n/\log n)$, then $\textsc{Sat}_P \in \mathsf{P}$.

10d Give a complete description of the algorithm computing $H(n)$ (as in the lecture notes) and prove that H is well-defined in that the algorithm terminates and computes some specific function.

10e Prove that not only does the algorithm terminate, but it can be made to run in time polynomial in n. (Note that there are a number of issues needing clarification here, such as, for instance, how to solve instances of CnfSat efficiently enough.)

10f Prove that $\textsc{Sat}_H \in \mathsf{P}$ if and only if $H(n) = \mathrm{O}(1)$.

10g Prove that if $\textsc{Sat}_H \notin \mathsf{P}$, then $H(n) \to \infty$ as $n \to \infty$.

10h Assuming that $\mathsf{P} \neq \mathsf{NP}$, prove that $\textsc{Sat}_H$ does not lie in P but also cannot be NP-complete.