KØBENHAVNS UNIVERSITET

Computability and Complexity: Problem Set 1

Due: Tuesday February 21 at 23:59 AoE .
Submission: Please submit your solutions via *Absalon* as a PDF file. State your name and e-mail address close to the 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. Make sure to explain your reasoning. *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 for problem sets stated on the course webpage always apply.
Collaboration: Discussions of ideas in groups of two to three people are allowed—and indeed, encouraged—but you should always write up your solutions completely on your own, from start to finish, and you should understand all aspects of them fully. It is not allowed to compose draft solutions together and then continue editing individually, or to share any text, formulas, or pseudocode. Also, no such material may be downloaded from or generated via the internet to be used in draft or final solutions. Submitted solutions will be checked for plagiarism. You should also clearly acknowledge any collaboration. State close to the 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 possible 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.
Grading: A score of 120 points is guaranteed to be enough to pass this problem set.
Questions: Please do not hesitate to ask the instructor if any problem statement is unclear, but please make sure to send private messages sometimes specific enough questions could give away the solution to your fellow students, and we want all of you to benefit from working on, and learning from, the problems. Good luck!

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 (10 p) Consider the reduction from 3-SAT to INDEPENDENTSET in the proof of Theorem 2.15 in Arora-Barak establishing that the latter problem is NP-hard. Suppose that we modify the reduction in the obvious way to be from general CNFSAT instead of 3-SAT. Would this work just as fine to establish NP-hardness, or would there be problems? For full credit, give a complete proof of the correctness of this new reduction or point out where it fails.

3 (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.

3a What is the complexity of 2-COLOURING?

3b 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.

4 (20 p) We proved in class that there are oracles relative to which P and NP are equal by defining the language $\text{EXPCOM} = \{\langle M, x, 1^n\rangle \,|\, M \text{ accepts } x \text{ within } 2^n \text{ steps}\}$ and showing that $\mathsf{P}^{\text{EXPCOM}} = \mathsf{NP}^{\text{EXPCOM}} = \mathsf{EXP}$. 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 $\text{EXPCOM}' = \{\langle M, x, 1^n\rangle \,|\, M \text{ accepts } x \text{ within } n \text{ steps}\}$. Do the equalities $\mathsf{P}^{\text{EXPCOM}'} = \mathsf{NP}^{\text{EXPCOM}'} = \mathsf{EXP}$ hold? Modify the argument we gave in class to establish these equalities or explain where the proof fails.

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

5 (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

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

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 $\mathsf{P} = \mathsf{NP}$?

6 (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 $\widetilde{F}$.

6a (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.)

6b (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.

6c (20 p) Consider the language

$$\textsc{MinUnsat} = \left\{F \,\middle|\, F \text{ is a minimally unsatisfiable CNF formula}\right\}.$$

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(s) on this problem set.

7 (50 p) Given a (multi)set $A = \{a_1, a_2, \ldots, a_m\}$ of integer terms and a target sum B, does there exist a subset $S \subseteq [m]$ such that $\sum_{i \in S} a_i = B$? This problem is known as SUBSETSUM and is NP-complete. We want to analyse a purported proof of NP-hardness and study what happens when one tinkers with the reduction.

7a (15 p) Consider the following suggested reduction of 3-SAT to SUBSETSUM. We are given a 3-CNF formula F with m clauses $C_1, \ldots, C_m$ over n variables $x_1, \ldots, x_n$. We build from this F a SUBSETSUM instance with $2(n + m)$ integer terms and target sum as follows, where all numbers below have $n + m$ decimal digits each:

- For each variable x_i, construct numbers t_i and f_i such that:
  - the ith digit of t_i and f_i is equal to 1;
  - for $n+1 \leq j \leq n+m$, the jth digit of t_i is equal to 1 if the clause C_{j-n} contains the literal x_i;
  - for $n+1 \leq j \leq n+m$, the jth digit of f_i is equal to 1 if C_{j-n} contains $\overline{x}_i$, and
  - all other digits of t_i and f_i are 0.
- For each clause C_j , construct numbers u_j and v_j such that
  - the $(n + j)$th digit of u_j and v_j is equal to 1 and
  - all other digits of u_j and v_j are 0.
- The target sum B has
  - jth digit equal to 1 for $1 \leq j \leq n$ and
  - jth digit equal to 3 for $n + 1 \leq j \leq n + m$.

Is the the above a correct reduction from 3-SAT to SUBSETSUM that proves the NP-hardness of the latter problem? If your answer is yes, then give a full proof of correctness showing that the reduction (i) is polynomial-time computable, (ii) maps yes-instances to yes-instances, and (iii) maps no-instances to no-instances. If your answer is no, then show how at least one of these conditions fails to hold.

7b (15 p) Given a 3-CNF formula F with m clauses over n variables, run the same reduction as in problem 7a except that the numbers u_j and v_j are omitted and the target sum B has all digits equal to 1. Formulas that map into satisfiable instances of SUBSETSUM under this modified reduction have a very specific form of satisfying assignments. Describe what such assignments look like.

7c (20 p) Consider the language HACKEDSAT consisting of 3-CNF formulas that map to satisfiable SUBSETSUM instances under the reduction in problem 7b. What is the complexity of deciding this language? Is it in NP? In P? Or NP-complete? For full credit, provide either a polynomial-time algorithm or a reduction from some problem proven NP-complete in chapter 2 in Arora-Barak or during the lectures.

8 (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 during the first two weeks of lectures of this course but does not necessarily know more than that. (However, you do not need to explain again the material in our first lectures, but can assume that they have been fully digested.)

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 that it is easy to find where in your solution the different items are dealt with.)

8a 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.

8b Prove that if $P(n) = \mathrm{O}(1)$, then $\textsc{Sat}_P$ is NP-complete.

8c Prove that if $P(n) = \Omega(n/\log n)$, then $\textsc{Sat}_P \in \mathsf{P}$.

8d 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.

8e 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 $\textsc{CnfSat}$ efficiently enough.)

8f Prove that $\textsc{Sat}_H \in \mathsf{P}$ if and only if $H(n) = \mathrm{O}(1)$.

8g Prove that if $\textsc{Sat}_H \notin \mathsf{P}$, then $H(n) \to \infty$ as $n \to \infty$.

8h Assuming that $\mathsf{P} \neq \mathsf{NP}$, prove that $\textsc{Sat}_H$ does not lie in P but also cannot be NP-complete.