KTH Computer Science and Communication

DD2446 Complexity Theory: Problem Set 1

Due: September 17, 2013, at 10:00 am. Submit your solutions as a PDF file by e-mail to `jakobn at kth dot se` with the subject line `Problem set 1:` ⟨`your name`⟩. Name the PDF file `PS1_`⟨`YourName`⟩`.pdf` (with your name coded in ASCII without national characters). Solutions should be written in LaTeX or some other math-aware typesetting system. 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 write down your own solution individually and understand all aspects of it fully. You should also acknowledge any collaboration. State at the beginning of the problem set if you have been collaborating with someone and if so with whom.

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 on in the lecture notes, or which can be found in chapters of Arora-Barak covered in the course, should be fair game, though, unless you are specifically asked to show something that we claimed without proof in class. It is hard to pin down 100% formal rules on what all this means—when in doubt, ask the lecturer.

About the problems: Some of the problems on this problem set are meant to be quite challenging and you are not necessarily expected to solve all of them. A total score of around 50 points should be enough for grade E, 80 points for grade D, 110 points for grade C, 140 points for grade B, and 170 points for grade A on this problem set. Any corrections or clarifications will be posted on the course webpage `www.csc.kth.se/utbildning/kth/kurser/DD2446/kplx13/`.

1 (10 p) Prove that either $\mathsf{P} \neq \mathsf{NP}$ or $\mathsf{NP} \neq \mathsf{EXP}$ must hold. (Note that it might well be that both of these inequalities are true, but we cannot prove either of them.)

2 (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.

3 (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.

4 (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).

5 (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.

5a (10 p) What is the complexity of 2-Colouring?

5b (10 p) What is the complexity of 4-Colouring?

For full credit on each of these subproblems, 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.

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

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

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 (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 $\widetilde{F}$.

7a (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.)

7b (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.

7c (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 on this problem set.

8 (50 p) The purpose of this problem is to fill in the gaps in the proof of Ladner's theorem that we did in class. Some of the questions below cover arguments already sketched during the lecture. For such questions you do not have to invent a new argument—just carefully filling in the missing details to give a complete proof is perfectly fine. Recall that

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

is the language of satisfiable CNF formulas padded by a suitable number of ones at the end as determined by the function H.

8a (10 p) Prove that if we want $\textsc{Sat}_H$ to have a chance to lie strictly between P-languages and NP-complete languages, then we must have $\omega(1) = H(n) = o(n/\log n)$, i.e., H has to be unbounded but grow asymptotically more slowly than $n/\log n$. (In this subproblem, you can assume for simplicity that we only consider monotonic functions H).

8b (10 p) Let H be defined by the algorithm presented during the lecture (and available in the handwritten lecture notes). Prove Claim 1 in the lecture notes, i.e., that H is well-defined in that the algorithm terminates and computes some specific function.

8c (10 p) For H as defined during the lecture, prove Claim 2 in the notes 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.)

8d (20 p) Fill in the missing details in the clinching argument of the proof of Ladner's theorem that if $\textsc{Sat}_H$ is NP-complete then it can be decided in polynomial time. To do so, you can freely use Claims 1–4 in the lecture notes as well as the fact that $\textsc{Sat}_H \notin \mathsf{P}$ has already been proven.

9 (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.

9a (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.

9b (15 p) Given a 3-CNF formula F with m clauses over n variables, run the same reduction as in problem 9a 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.

9c (20 p) Consider the language HACKEDSAT consisting of 3-CNF formulas that map to satisfiable SUBSETSUM instances under the reduction in problem 9b. 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.