KTH Computer Science and Communication

Computational Complexity: Problem Set 1

Due: Tuesday October 6, 2015. 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 coded in ASCII without national characters), and also state your name and e-mail address at the top of the first page. 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. (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, 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 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, 75 points for grade D, 100 points for grade C, 125 points for grade B, and 150 points for grade A on this problem set. Any corrections or clarifications will be given at `piazza.com/kth.se/fall2015/dd2445/` and any revised versions will be posted on the course webpage `www.csc.kth.se/DD2445/kplx15/`.

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 (10 p) When we reduced CircuitSat to 3-Sat, for every gate v_i in the circuit we introduced a Boolean variable z_i encoding the value computed at that gate. Assuming that v_i was an AND-gate with incoming wires from v_j and v_k, we then demonstrated how to write down a 3-CNF formula which was satisfied if and only if $z_i = z_j \wedge z_k$ holds.

Show how to write analogous 3-CNF formulas if v_i is an OR-gate with inputs from v_j and v_k or a NOT-gate with input from v_j.

4 (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.

4a What is the complexity of 2-Colouring?

4b 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.

5 (40 p) In our first lecture on Boolean circuits, we defined $\mathsf{DTIME}\big(T(n)\big)/a(n)$ as the class of languages decided by Turing machines M running in time $\mathrm{O}\big(T(n)\big)$ that also get a specific advice string $\alpha_n \in \{0,1\}^{a(n)}$ for inputs of size n. We then proved (or at least outlined a proof) that $\mathsf{P/poly} = \bigcup_{c,d \in \mathbb{N}^+} \mathsf{DTIME}\big(n^c\big)/n^d$.

Is it possible to change the definition so that not only the advice string α_n depends on the size of the input, but so that we can also pick different Turing machines M_n for different input sizes (while still maintaining that all running times be bounded by a common polynomial $p(n)$), and prove that P/poly is equal to the set of languages decided by such sequences of Turing machines $\{M_n\}_{n \in \mathbb{N}^+}$ with advice strings $\{\alpha_n\}_{n \in \mathbb{N}^+}$? Work out the details to show that this alternative definition is just as fine, or give a clear mathematical argument why it seems problematic.

6 (30 p) We say that a language L is in the complexity class Σ_i^p if there exists a polynomial-time Turing machine M and a polynomial $q(n)$ such that $x \in L$ if and only if

$$\exists u_1 \in \{0,1\}^{q(n)}\, \forall u_2 \in \{0,1\}^{q(n)}\, \exists u_3 \in \{0,1\}^{q(n)}\, \cdots Q_i u_i \in \{0,1\}^{q(n)}\, M(x, u_1, u_2, u_3, \ldots, u_i) = 1$$

(in words, there is a "witness" u_1 such that for all "challenges" u_2 there is a "witness" u_3 such that ... the Turing machine M accepts x given this extra information if and only if x is in the language).

Let Σ_iSat be the set of true formulas ψ on the form

$$\psi = \exists u_1 \in \{0,1\}^{q(n)} \forall u_2 \in \{0,1\}^{q(n)} \exists u_3 \in \{0,1\}^{q(n)} \cdots Q_i u_i \in \{0,1\}^{q(n)} \phi(u_1, u_2, u_3, \ldots, u_i)\ ,$$

where all u_i:s denote sets of variables and ϕ is a formula in propositional logic.[1] Show that Σ_iSat is a complete problem for the class Σ_i^p.

[1] Note that since all variables in ϕ are bound by quantifiers the formula ψ has a fixed truth value which is true or false. Hence, the "SAT" in Σ_iSat might seem like a bit of a misnomer. But on the other hand the standard CnfSat problem of deciding whether $\phi(u)$ is satisfiable is the same problem as whether the formula $\exists u \phi(u)$ is true, and from this point of view is is natural to talk about Σ_iSat problems higher up in the polynomial hierarchy.

7 (40 p) Recall that we write $\Pi_i^p = \mathsf{co}\Sigma_i^p$ to denote the complement of the complexity class Σ_i^p and that the union of all such classes form the polynomial hierarchy $\mathsf{PH} = \bigcup_{i \in \mathbb{N}^+} \Sigma_i^p = \bigcup_{i \in \mathbb{N}^+} \Pi_i^p$. If it would hold that $\mathsf{PH} = \Sigma_i^p$ one says that "the polynomial hierarchy collapses to the ith level" (which, as we said in class, is generally not believed to be the case).

Prove that if $\Sigma_i^p = \Pi_i^p$, then the polynomial hierarchy collapses to the ith level.

8 (60 p) ***For this problem, and this problem only, please look at any textbooks, search in the research literature, or roam the internet to find helpful information.***

The CnfSat problem is NP-complete, and so conventional wisdom is that it should require exponential time to solve in the worst case. However, as discussed in class there are so-called *SAT solvers* that successfully solve real-world instances of CnfSat with millions of variables, and even do so in linear time. The purpose of this problem is to investigate this seeming paradox.

8a Can current state-of-the-art SAT solvers decide any CNF formulas thrown at them, or is it the case that there are formulas, perhaps even not too large ones, that are infeasible in practice? Can you find any examples of constructions of such formulas together with empirical data showing that these formulas are indeed impossibly hard to solve, say, with hardness scaling exponentially? Are there any theoretical results explaining hardness results observed in practice?

As your solution to this subproblem, provide a brief but detailed discussion of your findings together with solid references where one can look up any definitions and/or proofs (i.e., not just a webpage but rather a research paper or possibly a book). 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.

8b One SAT solver that has been, and still is, hugely influential is MiniSat (`www.minisat.se`), and it is still quite good although it is no longer the best solver available (and so, in particular, hardness for MiniSat alone is not a sufficient indicator of hardness in problem 8a). This solver is available in the CSC Ubuntu environment by running the command `minisat` in a terminal window, and in this subproblem we want to play around with it a bit to get a sense of what SAT solvers can do.

Your task is to find a concrete, small CnfSat instance that is hard for MiniSat (possibly using insights from problem 8a). What is the smallest size of a CNF formula you can construct, measured as the total number of literals in the formula, that requires more than 10 minutes to be solved by MiniSat? (To make this question more precise, let us agree to run on `u-shell.csc.kth.se`, measuring user time as reported by `time`.) In your problem set solutions, provide a description of how your formula is constructed and exactly what size it has, and provide the address to an AFS directory where it can be accessed in the standard DIMACS format used by SAT solvers (and where the size is stated on a comment line). Make the directory readable by issuing the command `fs sa . system:anyuser rl` in the directory and `fs sa . system:anyuser l` in the directories in the path leading there, so that your formula can be checked during peer evaluation.

It should be possible to find a formula with 500-600 literals or less. The number of credits earned on this subproblem will increase as the formula size decreases. *In addition, the smallest formula submitted by any student that requires more than* 10 *minutes of CPU user time on* `u-shell.csc.kth.se` *will be awarded a 30-point bonus.*

Some information about how to use MiniSat and the standardized DIMACS format for CNF formulas can be found at `www.csc.kth.se/DD2445/kplx15/minisat.php`.