KTH Computer Science and Communication

DD2445 Complexity Theory: Problem Set 2

Due: Friday November 10, 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 2: ⟨your full name⟩`. Name the PDF file `PS2_⟨YourFullName⟩.pdf` with your name written in CamelCase without blanks and in ASCII without national characters. 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. *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 continue editing individually. 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 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) Under the assumption $\mathsf{NP} \subseteq \mathsf{P/poly}$, describe how to construct a polynomial-size family of circuits $\{C_{m,n}\}_{m,n\in\mathbb{N}^+}$ that take any CNF formula $\phi(x,y) = \phi(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n)$ of size m over $2n$ variables and any assignment $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n) \in \{0,1\}^n$ as inputs, and output an assignment $\beta = (\beta_1, \beta_2, \ldots, \beta_n) \in \{0,1\}^n$ such that it holds that $\phi(\alpha, C_{m,n}(\phi, \alpha)) = \phi(\alpha_1, \alpha_2, \ldots, \alpha_n, \beta_1, \beta_2, \ldots, \beta_n) = 1$ if such a β exists. That is, fill in the details in the construction that we claimed without proof when showing the Karp-Lipton theorem.

Remark: You do not need to provide an exact gate-by-gate specification of the circuits (especially since we do not believe $\mathsf{NP} \subseteq \mathsf{P/poly}$), but you should describe in reasonable detail what subcircuits you use and how they are glued together. Also, make sure to argue why the size is polynomial.

2 (10 p) We say that a language $L \subseteq \{0,1\}^*$ is *sparse* if there is a polynomial p such that it holds for every $n \in \mathbb{N}^+$ that $\left|L \cap \{0,1\}^n\right| \leq p(n)$. Show that if L is sparse, then $L \in \mathsf{P/poly}$.

3 (20 p) Show that $\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP}$.

4 (30 p) During the second half of the course we will be interested in proving size lower bounds for AC^0 circuits. When doing so, it can be convenient to use that any AC^0 circuit C can be modified to obtain an equivalent circuit C' with the following properties:

1. All gates in C' have fan-out 1 (i.e., it is what is known as a *formula*, with a DAG structure that is a tree).
2. All NOT ($\neg$) gates are at the input level of C' (i.e., they only apply to variables).
3. The AND ($\wedge$) and OR ($\vee$) gates alternate, so that at each level of C' all gates are either AND or OR.
4. The bottom level has AND gates of some small, bounded fan-in (for the purposes of this problem, let us say some global constant K).

Show how these modifications can be done without increasing the circuit depth by more than a constant and the size more than polynomially (so that C' is also a bounded-depth polynomial-size circuit computing the same function as C). If C is a circuit of size S and depth d, what size and depth do you get for C'?

5 (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 $\Sigma_i\textsc{Sat}$ 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 $\Sigma_i\textsc{Sat}$ is a complete problem for the class Σ_i^p.

6 (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.

[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 $\Sigma_i\textsc{Sat}$ might seem like a bit of a misnomer. But on the other hand the standard $\textsc{CnfSat}$ 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 $\Sigma_i\textsc{Sat}$ problems higher up in the polynomial hierarchy.

7 (40 p) In our lectures 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 $\mathsf{P/poly}$ 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.

8 (30 p) Recall that the language

$$\textsc{CircuitEval} = \big\{ \langle C, \alpha \rangle \,\big|\, C \text{ is a circuit that evaluates to 1 on } \alpha \big\}$$

is P-complete (we did not prove this in class, but this fact can be used freely in this problem). We want to understand the complexity of the closely related language

$$\textsc{CnfEval} = \big\{ \langle F, \alpha \rangle \,\big|\, F \text{ is a CNF formula that evaluates to 1 on } \alpha \big\}$$

(in both of the descriptions above we are assuming that the domain of α matches the number of variables in C or F, or else we have a no-instance due to syntax error).

Your task is either to prove that $\textsc{CnfEval}$ is also P-complete, just as $\textsc{CircuitEval}$ is, or else to explain why this seems unlikely and what problems you run into when trying to prove P-completeness.

9 (40 p) A *decision tree* T is a binary tree with edges directed from the root to the leaves and with leaves labelled 0/1, non-leaves labelled by variables x_i, and the two edges out of every non-leaf labelled 0 and 1, respectively. We say that T *represents* the Boolean function $f : \{0,1\}^n \to \{0,1\}$ if for every assignment α it holds that when starting in the root of T and following the edge labelled by $\alpha(x_i)$ out of every non-leaf labelled by x_i we end up in a leaf labelled by the value $f(\alpha)$. The *depth* of a decision tree T is the length of a longest root-to-leaf path in T.

In this problem we want to study some connections between decision trees, CNF formulas, and DNF formulas.

9a Suppose that a Boolean function f can be represented as a decision tree of depth d. Show that f can also be represented as a d-CNF formula and as a d-DNF formula.

9b Suppose that a Boolean function f can be written both as a k-CNF formula and as an ℓ-DNF formula. Show that this implies that f also can be represented as a decision tree of depth at most $k\ell$.

10 (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.

10a (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.

10b (15 p) Given a 3-CNF formula F with m clauses over n variables, run the same reduction as in problem 10a 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.

10c (20 p) Consider the language HACKEDSAT consisting of 3-CNF formulas that map to satisfiable SUBSETSUM instances under the reduction in problem 10b. 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.