Exercise I, Computational Complexity 2023

These exercises are for your own benefit. Feel free to collaborate and share your answers with other students. Solve as many problems as you can and ask for help if you get stuck for too long. Problems marked * are more difficult but also more fun :).

1 Prove that the following problems are in NP. Which of them are also in P?

(a) A Boolean formula is said to be in Disjunctive Normal Form (DNF) if it is an OR ($\vee$) of a number of terms where each term is an AND ($\wedge$) of the literals present in it. For instance, the following is a DNF formula:

$$(x \wedge y \wedge \bar{z}) \vee (\bar{y} \wedge z) \vee (\bar{x} \wedge \bar{y})$$

The DNF-SAT problem asks, given a DNF formula, if it is satisfiable.

(b) Given n positive integers $a_1, a_2, \ldots, a_n$ in their binary representation, is there a set $S \subseteq \{1, 2, \ldots, n\}$ such that

$$\sum_{i \in S} a_i = \sum_{i \in \bar{S}} a_i \,.$$

Here, $\bar{S}$ denotes the complement of the set S. That is $\bar{S} = \{1, 2, \ldots, n\} \setminus S$.

(c) Given a pair of integers (n, m) in binary, does there exist $2 \leq k \leq m$ such that k divides n?

(d) A k-CNF formula is a CNF formula where each clause has at most k literals. The 2-SAT problem asks if a 2-CNF formula given as input is satisfiable.

Solution:

Showing that a decision problem belongs to NP amounts to coming up with an efficient certificate system i.e. a polynomial-time Turing machine $\mathcal{V}$ together with a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for all $x \in \{0, 1\}^*$:

$$x \in L \iff \exists c \in \{0, 1\}^{p(|x|)} : \mathcal{V}(x, c) \text{ accepts}$$

(a) A certificate for a DNF-SAT instance φ is simply an assignment c to the variables such that $\phi(c) = 1$. Note that evaluating $\varphi(c)$ takes time $O(\text{poly}(|\varphi|))$. DNF-SAT belongs to P because of the simple Turing machine that iterates over all terms and accepts if a term with no contradiction (containing both x and $\bar{x}$ for some variable x) is seen.

(b) The certificate here would be a valid balanced partition S. The verifier thus only needs to compute two sums and compare the results. This problem is not known to be in P as it is NP-hard.

(c) One can simply take k as a certificate. The verifier then simply checks the divisibility of n by k - this boils down to computing a modulo and hence can be done in polynomial time. This problem is not known to be in P and not believed to be NP-hard either. Note that the related problem of checking primality is however in P.

(d) Similarly to DNF-SAT, a certificate for 2-CNF is a satisfying assignment. Perhaps surprisingly, there exists a polynomial-time algorithm for deciding whether such formula are satisfiable. The crux is to transform the instance into a directed graph and analyze its transitive closure; see, e.g., `https://en.wikipedia.org/wiki/2-satisfiability`.

2 Is NP is closed under **union**? That is, if $L, L' \in \mathsf{NP}$ then do we always have $L \cup L' \in \mathsf{NP}$? What about **intersection** $L \cap L'$, **complement** $\bar{L} = \Sigma^* \setminus L$, and **Kleene closure**, which is defined for a language $L \subseteq \Sigma^*$ by (here uv is the concatenation of strings u and v)

$$L^* = \{u_1 u_2 \dots u_k : u_1, u_2, \dots, u_k \in L \text{ and } k \geq 0\}.$$

(Hint: For one of the four operations mentioned above, it is still **unknown** *whether* NP *is closed under that operation—do not spend too much time trying to prove closure for all four!)*

Solution: Let $\mathcal{V}$ and $\mathcal{V}'$ be the poly-time verifiers witnessing $L, L' \in \mathsf{NP}$. We further assume that the certificates lengths are bounded by the same polyonomial $p : \mathbb{N} \to \mathbb{N}$ for both verifiers - this is without loss of generality as we can take the maximum of both.

$L \cup L' \in \mathsf{NP}$ is witnessed by the verifier $\mathcal{V}''$ that takes two certificates c, c' and on input (x, c, c') accepts if one of $\mathcal{V}(x, c)$ and $\mathcal{V}'(x, c')$ accept. Note that the certificate size is bounded by $2p$ and thus still polynomial. If we instead ask that $\mathcal{V}''$ accepts (x, c, c') if both $\mathcal{V}(x, c)$ and $\mathcal{V}'(x, c')$ accept, then $\mathcal{V}''$ witnesses that $L \cap L' \in \mathsf{NP}$.

The certificate for the Kleene closure would be the breakout of the input in k parts as well as k corresponding certificates. The verifier then verifies that the breakout of the input is indeed legal and that each certificate is valid using $\mathcal{V}$.

Finally, let us discuss the complement. It would be tempting to use a verifier $\widetilde{\mathcal{V}}$ that simply runs $\mathcal{V}$ and reverse its output. This would however be incorrect: it is possible that $x \in L$ has a certificate $c \in \{0, 1\}^*$ such that $\mathcal{V}(x, c)$ rejects. This would imply that $\widetilde{\mathcal{V}}(x, c)$ accepts and thus $x \in \overline{L}$: a contradiction. This shows that if $\mathsf{NP} = \mathsf{coNP}$, then a smarter argument needs to be used and indeed one hasn't been able to come up with one yet!

3 Define coNP as the class of languages whose complements are in NP, that is, $\mathsf{coNP} = \{L : \bar{L} \in \mathsf{NP}\}$. Show that if $\mathsf{P} = \mathsf{NP}$ then $\mathsf{NP} = \mathsf{coNP}$. Is the converse necessarily true?

Solution: Note that P is closed under complement. Indeed, if $L \in \mathsf{P}$, there exists a polytime decider $\mathcal{M}$ for L and switching the accept and reject state of $\mathcal{M}$ creates a decider for $\overline{L}$. Hence, if $\mathsf{P} = \mathsf{NP}$ then $\mathsf{NP} = \mathsf{P} = \mathsf{coP} = \mathsf{coNP}$. On the other hand, it is still possible that $\mathsf{NP} = \mathsf{coNP}$ but $\mathsf{P} \neq \mathsf{NP}$.

4 Suppose $\mathsf{P} = \mathsf{NP}$ and let A be a polynomial-time algorithm for SAT. Show that, by using A as a subroutine, we can find in polynomial time a satisfying assignment to a CNF formula if one exists. *(Hint: Call A repeatedly to build up the satisfying assignment one variable at a time.)*

Solution: Let φ be a formula to satisfy with variables $\{x_i\}_{i \in [k]}$. First, run A on φ to check that φ is indeed satisfiable. If it is not then the job is done. If φ is satisfiable, it has to be satisfiable for $x_1 = 0$ or $x_1 = 1$. Add a clause "x_1" to φ (this is still a 2-CNF) and if this new formula is satisfiable (this can again be checked using A), then substitute 1 to x_1 and continue with a smaller formula not containing x_1. Else do the same but substitute 0 to x_1. This property of NP languages is called *self-reducibility*.