Homework 1, Computational Complexity 2024

The deadline is 23:59 on Wednesday 9 October. Please submit your solutions on Moodle. Typing your solutions using LaTeX is strongly encouraged. The problems are meant to be worked on in groups of 2–3 students. Please submit only one writeup per team. You are strongly encouraged to solve these problems by yourself. If you must, you may use books or online resources to help solve homework problems, but you must credit all such sources in your writeup and you must never copy material verbatim.

1 A *decisive* nondeterministic Turing machine is one where each nondeterministic computation path outputs either *yes*, *no*, or *maybe*. We say such a machine decides a language L if the following holds: If $x \in L$, then all computations end up with *yes* or *maybe*, and at least one *yes*. If $x \notin L$, then all computations end up with *no* or *maybe*, and at least one *no*. Show that L is decided by a decisive polytime NTM if and only if $L \in \mathsf{NP} \cap \mathsf{coNP}$.

Solution: Let N be a polytime decisive NTM that decides L and N' a copy of N with *maybe* branches changed to *no* branches. N is a polytime NTM and we show that it decides L. To do so, fix some $x \in \{0, 1\}^*$. If $x \in L$, then N has some branch that outputs *yes* and therefore N' accepts x. On the other hand, if $x \notin L$, then all branches of N output *no* or *maybe*, therefore all the branches of N' output *no* and N' rightly rejects. One can show in a similar manner that $L \in \mathsf{coNP}$ by changing *maybe* branches into *yes* branches.

Regarding the other direction, fix some $L \in \mathsf{NP} \cap \mathsf{coNP}$. Let M be a non-deterministic poly-time decider for L and N a co-non-deterministic poly-time decider for L. Consider the machine D that first runs M on x, get some non-deterministic *yes*/*no* guess, then runs N on x to get yet another non-deterministic *yes*/*no* guess and then finally produce its own guess according to the following table:

M's guess	N's guess	D's guess
yes	yes	yes
yes	no	maybe
no	yes	maybe
no	no	no

This machine runs in polynomial non-deterministic time and we now argue it indeed decides L. To do so, fix some $x \in \{0, 1\}^*$. If $x \in L$, then all branches of N accept x and at least one branch of M accept. Hence, branches of D either output *yes* or *maybe* but at least one accepts. Correspondingly, if $x \notin L$, all branches of M reject and at least one branch of N rejects as well so that D outputs *no* or *maybe* but has at least one *no* path.

2 Show that the following *distinct-3SAT* problem is NP-complete:

$$\text{D3SAT} = \{\langle\varphi\rangle : \varphi \in \text{3SAT and each clause of } \varphi \text{ involves three distinct variables}\}.$$

(For example, $(x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee y \vee \overline{w}) \in$ D3SAT, whereas $(x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee y \vee \overline{y}) \notin$ D3SAT since the clause $(\overline{x} \vee y \vee \overline{y})$ involves only two distinct variables.)

Solution: D3SAT is in NP: the certificate will be an assignment to a formula, and it is easy to check in polynomial time 1) that the assignment is a satisfying one; 2) that each clause of φ has three distinct variables.

D3SAT is NP-hard: we can build a reduction from 3SAT to D3SAT in the following way. Suppose a formula ψ is such that its every clause contains no more than 3 literals. It is enough to generate a formula $f(\psi) = \varphi$ such that its every clause contains three literals with distinct variables and such that φ is satisfiable iff ψ is satisfiable.

First, while ψ contains clauses with one literal, we set these literals to 1, since any satisfying assignment to ψ should satisfy such clauses. Also, while ψ contains clauses with fewer variables than literals (they contain both $\neg x$ and x for some x), we delete such clauses, since they are equivalent to 1 and do not affect the satisfiability of ψ.

Then, for each clause $C \in \psi$, we consider two cases:

- C contains three distinct variables. Then we add C to φ;
- C contains two distinct variables. Then we add to φ clauses $C \vee x$ and $C \vee \neg x$ for a fresh variable x.

It's easy to check that any extension of a satisfying assignment for ψ satisfies φ, and any restriction of a satisfying assignment for φ on variables of ψ satisfies ψ.

3 Let $G = (V, E)$ be an undirected graph. We say that a vertex set $K \subseteq V$ is a *kernel* iff (i) for any two $v, u \in K$ we have $\{v, u\} \notin E$, and (ii) for every $u \in V \setminus K$ there is $v \in K$ such that $\{u, v\} \in E$. In other words, a kernel is a set that is both (i) independent and (ii) dominating*. Show that the following problem is NP-complete:

$$\textsc{Kernel} = \{\langle G, k\rangle : G \text{ has a kernel of size at most } k\}.$$

(You can use any NP-complete problem discussed in class/exercises in your reductions.)

Solution: We first show that $\textsc{Kernel} \in \mathsf{NP}$. A certificate for $\langle G, k\rangle \in \textsc{Kernel}$ is a set of vertices K and it can be checked in polynomial time that:

1. K is an independent set,
2. K is a dominating set,
3. K has size at most k.

We then show that Kernel is NP-hard by reducing 3SAT to Kernel. Given a 3SAT instance φ over variables $\{x_i\}_{i\in[n]}$ and clauses $\{C_j\}_{j\in[m]}$, we construct a graph $G = (V, E)$ as follows. V has one vertex u_j for each clause C_j and for each $i \in [n]$ it contains the vertices v_i, v_i^+ and v_i^-. The edge set makes sure that each group $\{v_i, v_i^+, v_i^-\}$ forms a triangle – i.e. E contains edges $\{v_i, v_i^+\}$, $\{v_i^+, v_i^-\}$ and $\{v_i^-, v_i\}$ to E. E also models the clauses as follows. For each clause $C_j = \ell_1 \vee \ell_2 \vee \ell_3$, we add an edge $\{v_i^+, u_j\}$ if $\ell_k = x_i$ and an edge $\{v_i^-, u_j\}$ if $\ell_k = \overline{x_i}$ for $k \in [3]$. See Figure 1 for a visual representation of the reduction. Finally, we define the Kernel instance with $\langle G, n\rangle$. As this reduction can be performed in polynomial time it remains to argue that

*https://en.wikipedia.org/wiki/Dominating_set

$\varphi \in 3\text{SAT} \iff \langle G, n\rangle \in \textsc{Kernel}$. Suppose first that $\varphi \in 3\text{SAT}$ and let $x \in \{0, 1\}^n$ be a satisfying assignment and define $K \subseteq V$ with:

$$K = \{v_i^+ : x_i = 1\}_{i\in[n]} \cup \{v_i^- : x_i = 0\}_{i\in[n]}$$

Note first that K has size n. K is furthermore an independent set as it contains no clause-vertex K and has only one vertex per group $\{v_i^+, v_i^-, v_i\}$. Finally, as φ is satisfied by x it must be that K dominates G as well and so $\langle G, n\rangle \in \textsc{Kernel}$.

For the other direction, fix a kernel K of size n in G. Observe that for each $i \in [n]$, $|K \cap \{v_i, v_i^+, v_i^-\}| = 1$. Indeed, if the intersection size is 0, then v_i is not dominated by K and if the intersection size is 2 or 3, then K is not an independent set. This further implies that K does not contain any u_j as $|K| = n$. We define a satisfying assignment $x \in \{0, 1\}^n$ for φ with:

$$x_i = \begin{cases} 1 & \text{if } v_i^+ \in K \text{ or } v_i \in K \\ 0 & \text{if } v_i^- \in K \end{cases}$$

As $\{u_j\}_{j\in[m]}$ is dominated by K, it must be that x satisfies each clause C_j and thus $\varphi \in 3\text{SAT}$.

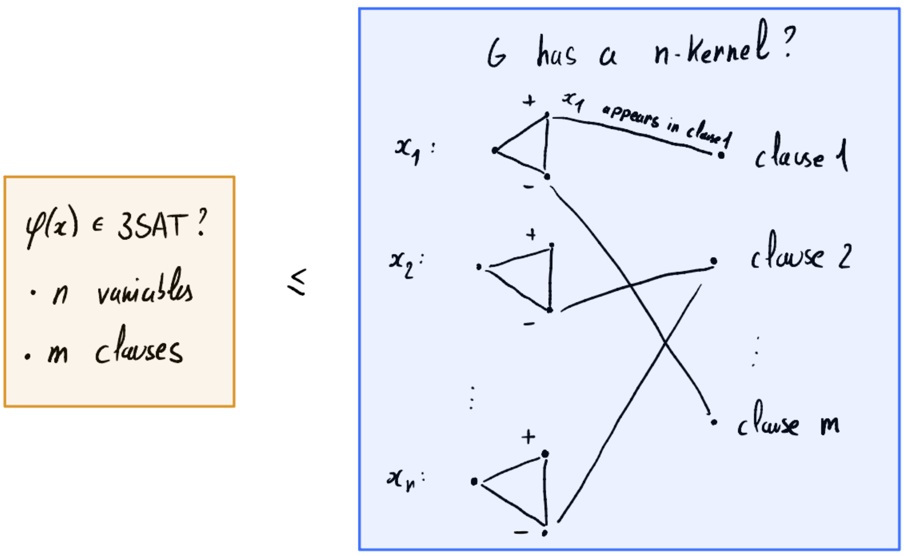

Figure 1. A visual representation of the reduction in Problem 3. The crux is that if K is an n-kernel then there is exactly one vertex per triangle in K. Selecting these forms a satisfying assignment for φ.