Exercise II, Computational Complexity 2025

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 the following basic properties of polynomial-time reductions.

(a) $A \leq_p A$. (*reflexivity*)

(b) If $A \leq_p B$ and $B \leq_p C$, then $A \leq_p C$. (*transitivity*)

(c) If $A \leq_p B$ then $\bar{A} \leq_p \bar{B}$.

(d) If $A \leq_p B$ and $B \in \mathsf{NP}$, then $A \in \mathsf{NP}$.

(e) If $A \in \mathsf{P}$ then $A \leq_p \{1\}$.

Solution: Recall that $A \leq_p B$ if there exists a function $f : \{0,1\}^* \to \{0,1\}^*$ computable in polynomial time such that for any $x \in \{0,1\}^*$, $x \in A \iff f(x) \in B$.

(a) Taking f to be the identity function witnesses $A \leq_p A$. Indeed, the identity function is polynomial time computable.

(b) Let f_1 and f_2 be the witnesses to $A \leq_p B$ and $B \leq_p C$. Note that $f := f_2 \circ f_1$ is such that for any $x \in \{0,1\}^*$:

$$x \in A \iff f_1(x) \in B \iff f_2(f_1) \in C \iff f(x) \in C$$

Observe that f is also efficiently computable since the size of $f_1(x)$ is polynomial in $|x|$ so that the size of $f(x)$ is also polynomial in $|x|$. This shows that $\leq_p$ is transitive.

(c) Let f be the witness to $A \leq_p B$ which is poly-time computable by assumption. Observe that for any $x \in \{0,1\}^*$:

$$x \in \overline{A} \iff x \notin A \iff f(x) \notin B \iff f(x) \in \overline{B}$$

(d) Let f be a witness for $A \leq_p B$ and $\mathcal{V}$ be a verifier for B that has certificates bounded by some polynomial $p : \mathbb{N} \to \mathbb{N}$ (those two objects exist because $B \in \mathsf{NP}$). Now, a verifier for A is simply $\mathcal{V}'$ that on input (x, c) runs $\mathcal{V}(f(x), c)$. This verifier is correct because for any $x \in \{0,1\}^*$:

$$\begin{aligned} x \in A &\iff f(x) \in B \\ &\iff \exists c \in \{0,1\}^{p(|f(x)|)} : \mathcal{V}(f(x), c) \\ &\iff \exists c \in \{0,1\}^{p(|f(x)|)} : \mathcal{V}'(x, c) \end{aligned}$$

Note that certificates have again poly-size since f is poly-time computable so that $A \in \mathsf{NP}$ too.

(e) Since $A \in \mathsf{P}$, A has some efficient decider $\mathcal{D}$. Now, let $f : \{0,1\}^* \to \{0,1\}^*$ be the function defined by $f(x) = 1$ if $x \in A$ and $f(x) = 0$ else. This function can be computed in poly-time using $\mathcal{D}$ and witnesses that $A \leq_p \{1\}$.

2 What is wrong with the following "proof" that 2-SAT is NP-complete? Since 3-SAT is in NP, so is 2-SAT. We then prove that 3-SAT $\leq_p$ 2-SAT by giving a polynomial time computable reduction. We define a function $\varphi \mapsto f(\varphi)$ which maps boolean 2-CNF fomulas to 3-CNF formulas. For a formula $\varphi = (a_1 \vee b_1) \wedge (a_2 \vee b_2) \wedge \cdots \wedge (a_n \vee b_n)$, where a_i, b_i are literals, we define $f(\varphi) = (a_1 \vee b_1 \vee b_1) \wedge (a_2 \vee b_2 \vee b_2) \wedge \cdots \wedge (a_n \vee b_n \vee b_n)$. Note that

φ is satisfiable $\Leftrightarrow$ $f(\varphi)$ is satisfiable.

Clearly, f is polynomial time computable given φ. Hence 3-SAT $\leq_p$ 2-SAT, and in particular it follows that 2-SAT is NP-complete.

Solution: The issue is that φ is actually reducing from 2-SAT to 3-SAT, effectively proving that 2-SAT $\leq_p$ 3-SAT. This is indeed true since 3-SAT $\in$ NP. To prove that 2-SAT is NP-complete, we actually need the opposite statement 3-SAT $\leq_p$ 2-SAT.

3 Show that if any NP-complete problem lies in coNP, then NP = coNP.

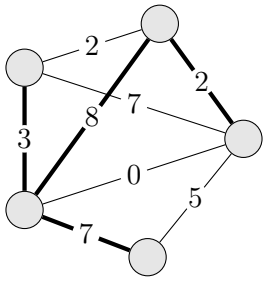
Solution: Suppose that $L \in$ coNP is NP-complete. We first show how to use that assumption to prove that any $L' \in$ NP is also in coNP - so that effectively NP $\subseteq$ coNP. Since L is NP-complete, we have $L' \leq_p L$ and thus $\overline{L'} \leq_p \overline{L}$. But since $\overline{L} \in$ NP, this means that $\overline{L'} \in$ NP too and hence $L' \in$ coNP. The other containment coNP $\subseteq$ NP can be proved in a similar fashion.

4 Prove that the following problem is NP-complete: Given a set S, a collection $\mathcal{C}$ of subsets of S and a number k, is there a subset $T \subseteq S$ of size k such that $T \cap C_i \neq \emptyset$ for all $C_i \in \mathcal{C}$? *(Hint: reduce from* VertexCover*.)*

Solution: This problem is called HittingSet. Note that HittingSet $\in$ NP is witnessed by the certificate T and the verifier which simply checks that $T \subseteq S$, $|T| = k$ and $C \cap T \neq \emptyset$ for each $C \in \mathcal{C}$. We further argue that HittingSet is NP-complete by showing that VertexCover $\leq_p$ HittingSet.

We simply transform the instance $(G = (V, E), k)$ of VertexCover into the instance (V, E, k) for HittingSet. Let us show that this transformation is correct. Suppose first that $(G = (V, E), k) \in$ VertexCover. This means that there exists a way to choose $T \subseteq V$ with $|T| = k$ such that each $\{u, v\} = e \in E$ is covered by T: $u \in T$ or $v \in T$, i.e. $e \cap T \neq \emptyset$. Hence, $(V, E, k) \in$ HittingSet. For the other direction, if $(V, E, k) \in$ HittingSet, then there exists some $T \subseteq V$ of size k with $T \cap e \neq \emptyset$ for each $e \in E$ and thus T is a valid vertex cover of size k for G. Therefore, $(G = (V, E), k) \in$ VertexCover.

5 Let $G = (V, E)$ be a graph and $w \colon E \to \mathbb{N}$ an assignment of non-negative integer weights to the edges. A subset of edges $E' \subseteq E$ is a *spanning tree* if the subgraph (V, E') is a tree (no cycles) that connects all vertices. The weight of the spanning tree is $\sum_{e \in E'} w(e)$. For example, the bold edges below form a spanning tree of weight $3 + 7 + 8 + 2 = 20$.



In the Exact Spanning Tree problem (ESP for short), the input consists of a graph $G = (V, E)$, edge weights $w\colon E \to \mathbb{N}$, and a target $k \in \mathbb{N}$. The goal is to decide whether G contains a spanning tree of weight exactly k. That is,

$$\text{ESP} \;=\; \{\langle G, w, k\rangle : G \text{ contains a spanning tree of weight exactly } k\}.$$

Prove that ESP is NP-complete by finding a reduction Subset-Sum $\leq_p$ ESP.

Solution: In an instance $\langle S, t\rangle$ of Subset-Sum, where S is a set of non-negative integers and t is a non-negative integer, one has to decide whether $\exists$ a subset T of S that sums to t. We will convert an instance of Subset-sum into ESP as follows.

1. For each integer i in S, add two separate vertices v and v' to G. Add an edge of weight i between these two new vertices.
2. For every pair of vertices in G not connected by an edge, add an edge of weight 0.
3. Set $k = t$.

The reduction clearly takes polynomial time. Now we prove that $\langle S, t\rangle \in$ Subset-Sum iff $\langle G, k\rangle \in$ ESP.

1. If $\langle S, t\rangle \in$ Subset-Sum, $\exists$ a $T \subseteq S$ that sums to t. Hence, a spanning tree in G can be formed by taking edges corresponding to integers in T and then connecting the isolated vertices by zero edges.
2. If $\langle G, k\rangle \in$ ESP, the sum of edges selected in the spanning tree sum to k. By taking the integers corresponding to the selected non-zero edges in S, one gets T that sums to t. Hence, $\langle S, t\rangle \in$ Subset-Sum.

6 (*) A multi-variate polynomial equation with integer coefficients is called a *Diophantine equation*; for example, $x^2y - y^2 + 6 = 0$. Let DEQ be the language consisting of all (binary encodings of) Diophantine equations that admit an integer solution; for example, $x = 1$, $y = -2$ for the above equation. Find a reduction 3-SAT $\leq_p$ DEQ. Do we have DEQ $\in$ NP?

Solution: Let $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_k$ with $C_i = \ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}$ be a 3-SAT instance over boolean variables $\{x_j\}_{j\in[m]}$. We describe how to transform φ into a Diophantine equation. For each boolean variable x_i we introduce a corresponding integer variable y_j. Let A be the operator that transforms a clause C_i with negative literals $N_i \subseteq [m]$ and positive literals $P_i \subseteq [m]$ into the polynomial $A(C_i) := \prod_{j\in N_i} y_j \cdot \prod_{j\in P_i}(1 - y_j)$. Now, associate to φ the polynomial p_φ over variables $\{y_j\}_{j\in[m]}$:

$$p_\varphi(y) := \sum_{i\in[k]} (A(C_i)[y])^2$$

So that formally our tranformation is $f(\varphi) \mapsto p_\varphi = 0$. Note that this function f is poly-time computable and indeed yields diophantine equations. For instance, the 3-SAT instance $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_4 \vee x_5) \wedge (\overline{x_3} \vee \overline{x_4} \vee \overline{x_2})$ is mapped to the following diophantine equation:

$$((1 - x_1) \cdot (1 - x_2) \cdot x_3)^2 + (x_1 \cdot (1 - x_4) \cdot (1 - x_5))^2 + (x_3 \cdot x_4 \cdot x_2)^2 = 0$$

We prove that the tranformation is correct. Suppose first that $\varphi \in$ 3-SAT. Then, it implies that there is a satisfying assignment $\{c_i\}_{i\in[m]}$. Note that $p_\varphi(c) = 0$ since each $A(C_i)(c) = 0$. On

the other hand, suppose that p_φ has a solution c. Note that this means $A(C_i)(c) = 0$ for each $i \in [k]$. Thus, at least one product in each clause is zero, so that $\varphi \in$ 3-SAT.

It would be tempting to say that DEQ $\in \mathsf{NP}$ by just using the solution of the equation as the certificate, but this is not good enough: there exists Diophantine equations that only have exponential solutions. This argument is not strong enough to rule out DEQ $\in \mathsf{NP}$ but it turns out that DEQ is undecidable (hence DEQ $\notin \mathsf{NP}$). This is the result of a long line of work that solved Hilbert's tenth problem.