Problem Set 1

- Due date: **12 Feb, 2023** (released on 28 Jan, 2023)
- The points for each problem is indicated on the side. The total for this set is **100** points.
- The problem set has a fair number of questions so please do not wait until close to the deadline to start on them. Try and do one question every couple of days.
- Turn in your problem sets electronically (PDF; either LaTeXed or scanned etc.) via email.
- Collaboration with other students taking this course is encouraged, but collaboration with others is not allowed. Irrespective of this, all writeups must be done individually and must include names of all collaborators (if any).
- Referring to sources other than the text book and class notes is STRONGLY DISCOURAGED. But if you do use an external source (eg.,other text books, lecture notes, or any material available online), ACKNOWLEDGE all your sources (including collaborators) in your writeup. This will not affect your grades. However, not acknowledging will be treated as a serious case of academic dishonesty.
- Be clear in your writing.

---

1. **[Classification of problems]** (3 + 3 + 4)

   For each of these problems, mention (with justification) if they are in P, or not (known to be) in P, or in NP, or in coNP, or is NP-hard, or is coNP-hard etc.

   (a) Factoring $= \left\{(1^n, 1^k) \ : \ n \text{ has a prime factor less than } k\right\}$.

   (b) Contradiction $= \{\langle\varphi\rangle \ : \ \langle\varphi\rangle$ encodes a formula that is false for every assignment.$\}$ (For example, $\langle x_1 \wedge \neg x_1\rangle$ is in the language.)

   (c)

   $$\text{EquivalentFormulas} = \left\{ (\langle\varphi_1\rangle, \langle\varphi_2\rangle) \ : \ \begin{array}{c} \langle\varphi_1\rangle, \langle\varphi_2\rangle \text{ encode two formulas such that} \\ \text{for all } x \text{ we have } \varphi_1(x) = \varphi_2(x). \end{array} \right\}$$

2. **[Properties of reductions]** (5)

   Suppose L_1, L_2 be two arbitrary languages with $L_1 \leq_m^{\text{poly}} L_2$ (that is, there is a polynomial time many-one reduction from L_1 and L_2).

   Answer each of these questions as true/false with brief justifications.

   (a) If L_1 is in P, then so is L_2.

   (b) If L_2 is in P, then so is L_1.

   (c) If L_1 is NP-complete, then so is L_2.

   (d) If L_2 is NP-complete, then so is L_1.

   (e) If L_1 is in DTIME$(2^{o(n)})$, then so is L_2.

   (f) If L_2 is in DTIME$(2^{o(n)})$, then so is L_1.

3. **[Operations on languages]** (10)

   (a) If L_1, L_2 are two languages in NP, show that the languages $L_1 \cap L_2$ and $L_1 \cup L_2$ are in NP as well.

(b) For any three languages L_1, L_2, L_3,

$$\text{Maj}(L_1, L_2, L_3) = \{x \ : \ x \text{ is in at least two of the } L_i\text{'s}\}.$$

Show that, if $L_1, L_2, L_3 \in \mathsf{NP}$, then the language $\text{Maj}(L_1, L_2, L_3)$ is also in NP.

(c) For two languages L_1, L_2, let $L_1 \oplus L_2 = \{x \in \Sigma^* \ : \ x \text{ is in exactly one of } L_1, L_2\}$. If $L_1, L_2 \in \mathsf{NP} \cap \mathsf{coNP}$, show that $L_1 \oplus L_2 \in \mathsf{NP} \cap \mathsf{coNP}$ as well.

4. **[P-complete languages]** (10)

List two languages in P that are P-complete under polynomial-time many-one reductions, and two languages in P that are *not* P-complete under polynomial-time many-one reductions.

5. **[Dominating set is NP-complete.]** (10)

In an undirected graph $G = (V, E)$, a set of vertices $S \subseteq V$ is said to be a *dominating set* if every vertex $v \in V$ either is an element of S or has a neighbour in S.

Show that the following language

$$\text{DominatingSet} = \left\{ (\langle G \rangle, 1^k) \ : \ \begin{array}{l} \langle G \rangle \text{ encodes an undirected graph that} \\ \text{has a dominating set of size at most } k \end{array} \right\}$$

is NP-complete.

6. **[Tape reduction with non-determinism.]** (10)

Suppose you are given a non-deterministic Turing machine M deciding a language L uses 10827 tapes running in time $T(n) : \mathbb{N} \to \mathbb{N}$. Show that there is an equivalent Turing Machine with just two work-tapes that decides the same language running in time $T'(n)$ such that $T'(n) = O(T(n))$.

[Hint: Try to use one tape for the interleaved version of 10827 tapes of M, and use the other tape to just write down what you guess the heads do in each of the steps.]

7. **[If P = NP...]** (3 + 7 + 5)

(a) If $\mathsf{P} = \mathsf{NP}$, show that $\mathsf{NP} = \mathsf{coNP}$.

(b) A Σ_2 formula is a quantified formula ψ of the form

$$\psi := \exists x \in \{0,1\}^n \, \forall y \in \{0,1\}^n \ : \ \varphi(x, y)$$

For example,

$$\exists x \in \{0,1\}^n \, \forall y \in \{0,1\}^n \ : \ ((x_1 \wedge \neg y_1) \vee (\neg x_1 \wedge y_1)) \cdots ((x_n \wedge \neg y_n) \vee (\neg x_n \wedge y_n))$$

is the formula that asserts that for there is a string x, that is the bit-wise negation of all strings y, which of course is false.

Define the language $\text{True}\,\Sigma_2$ as $\{\langle \psi \rangle \ : \ \langle \psi \rangle \text{ encodes a } \Sigma_2 \text{ formula that is true.}\}$.

If $\mathsf{P} = \mathsf{NP}$, show that $\text{True}\,\Sigma_2$ is in P.

(c) Can you extend your proof for $\text{True}\,\Sigma_3$? That is, if $\mathsf{P} = \mathsf{NP}$, can you show that $\text{True}\,\Sigma_3 \in \mathsf{P}$ as well?

As you might expect, Σ_3 formulas of the form

$$\exists x \in \{0,1\}^n \, \forall y \in \{0,1\}^n \, \exists z \in \{0,1\}^n \ : \ \varphi(x, y, z).$$

8. **[Reductions between reductions.]** (5 + 5 + 5)

In class we saw the notion of Turing reduction, and many-one reductions. Here is a brief description of it, along with two other types or reductions. Let $\Sigma = \{0, 1\}$.

Many-one reductions. A many-one reduction from a language L_1 to a language L_2, denoted by $L_1 \leq_m L_2$, is given by a function $f : \Sigma^* \to \Sigma^*$ such that for all $x \in \Sigma^*$ we have $x \in L_1 \iff f(x) \in L_2$. The running time of the reduction is the running time of the Turing machine computing f.

Turing reductions. A Turing-reduction from a language L_1 to a language L_2, denoted by $L_1 \leq_{\text{TM}} L_2$, is given by an oracle Turing machine M such that M^{L_2} decides L_1. The running time fo the reduction is the running time of the M.

Projection reductions. A projection-reduction from L_1 to L_2, denoted by $L_1 \leq_{\text{proj}} L_2$, is given by a sequence of functions $\left\{f_i : \Sigma^i \to \Sigma^{m_i} \ : \ i \in \mathbb{N}\right\}$, one for each length n such that

- For all $x \in \Sigma^*$ with $|x| = n$, $x \in L_1 \Leftrightarrow f_n(x) \in L_2$.
- Each of the functions f_i's are just *projections*, in the sense that every output bit is either a constant, or equals one of the input bits or its negation. For example, $f_3(x_1, x_2, x_3) = (0, \neg x_3, x_3, 1, x_1, 1, \neg x_2, \neg x_1)$ is a projection function.

The running time of this reduction is given by just the function $i \mapsto m_i$.

Truth-table reductions. A truth-table reduction from L_1 to L_2, denoted by $L_1 \leq_{\text{TT}} L_2$, is given by a Turing machine M that, on input x, outputs a set of finite strings $y_1, \ldots, y_k$ and boolean function $h : \Sigma^k \to \Sigma$ such that $x \in L$ if and only if $h(L_2(y_1), \ldots, L_2(y_k)) = 1$. (In other words, it is a Turing reduction where the queries and the post-processing function have to be written down beforehand before actually running the oracle for L_2.)

The running time of the reduction is the running time of the Turing machine M.

(a) Among the four reductions above, which ones imply another? That is, name all $X, Y \in \{\text{m}, \text{TM}, \text{proj}, \text{TT}\}$ from above such that you can formally show that $L_1 \leq_{\text{X}}^{\text{poly}} L_2$ implies $L_1 \leq_{\text{Y}}^{\text{poly}} L_2$ (where the poly refers to only polynomial time reductions.)

(b) For what $X, Y \in \{\text{many-one}, \text{Turing}, \text{projection}, \text{truth-table}\}$ can you prove the following statement:

> If L is NP-complete under polynomial-time X-reductions, then L is NP-complete under Y-reductions as well.

(c) Give one non-trivial example of each of the above kinds for reductions.

9. **[System of quadratic equations]** (8 + 7)

A system of quadratic equations $\mathcal{Q}$, is specified by a set of quadratic equations. For example,

$$\left\{\begin{array}{rl} x_1x_2 + x_3x_4 + \cdots x_{n-1}x_n & = 0 \\ x_1 + 7x_3x_n + 3x_1^2 & = 0 \\ \vdots & \\ x_2x_3 - 7x_9x_{12} & = 0 \end{array}\right\}$$

Let $\text{QuadEqns} = \left\{ \langle \mathcal{Q} \rangle \ : \ \begin{array}{c} \langle \mathcal{Q} \rangle \text{ encodes a system of quadratic equations} \\ \text{such that some } x \in \{0,1\}^* \text{ satisfies all of them.} \end{array} \right\}$.

(a) Show that QuadEqns is NP-complete.

(b) Turns out, there is an important result in mathematics that states the following:

> Suppose $f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}) \in \mathbb{C}[x_1, \ldots, x_n]$ are polynomials such that there is no $\mathbf{x} \in \mathbb{C}^n$ that satisfies all of them. Then, there are polynomials $g_1, \ldots, g_m$ such that $f_1 g_1 + \cdots + f_m g_m = 1$.

Now, given a system $\mathcal{Q}$, we can always add equations of the form $x_i^2 - x_i = 0$ to ensure that only $x_i \in \{0, 1\}$ can satisfy them. Given such a system $\mathcal{Q}$, here is a "proof" for QuadEqns $\in$ coNP.

> It suffices to show that the complement of QuadEqns is in NP. Given any input $\mathcal{Q} = \{f_1, \ldots, f_m\}$ that is *not* satisfied by any $\mathbf{x} \in \{0, 1\}^n$, let $\mathcal{Q}'$ be the system
>
> $$\mathcal{Q}' = \mathcal{Q} \cup \left\{ x_i^2 - x_i = 0 \ : \ i = 1, \ldots, n \right\}.$$
>
> Since the system $\mathcal{Q}'$ has no complex solutions either (since we enforced that $x_i^2 - x_i = 0$ for all i), from the above fact, we know that there must exist polynomial $g_1, \ldots, g_m, h_1, \ldots, h_n$ such that
>
> $$g_1 f_1 + \cdots + g_m f_m + h_1 \cdot (x_1^2 - x_1) + \cdots + h_n \cdot (x_n^2 - x_n) = 1.$$
>
> So the non-deterministic Turing machine can just guess $g_1, \ldots, g_m, h_1, \ldots, h_m$ and verify if the above equality holds.
> This shows that the complement of QuadEqns is in NP, and hence QuadEqns $\in$ coNP.

What is wrong with the above "proof"?