Problem Set 3

- Due date: **25 Apr, 2025** (released on 5 Apr, 2025).
- The points for each problem is indicated on the side. The total for this set is **85** 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. If you submit a LaTeX document, you will get an additional **10 points**.
- 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.

---

1. **[Self-reducibility]** (10)

   A language L is said to be length-self-reducible if there is a polynomial time computable functions F, G with the following properties:

   > F takes as input a string x of length n, and outputs $m = n^2$ strings $y_1, \ldots, y_m$ with the property that each y_i satisfies $|y_i| < |x|$ (that is, each y_i is a string of smaller length than x). The function G takes inputs as bits $b_1, \ldots, b_m$ and returns 0 or 1.
   >
   > For any string x, let $b_i = \mathbb{1}[y_i \in L]$ for $i \in [m]$ where $F(x) = (y_1, \ldots, y_m)$. Then, $x \in L$ if and only if $G(b_1, \ldots, b_m) = 1$.

   In other words, we have a function F that takes x and returns many smaller length strings, and G tells you how to check if $x \in L$ from the information of which of the m strings given by F were in L.

   Show that any length-self-reducible language L is in PSPACE.

   Bonus: Suppose you instead worked with *lex-self-reducible*, where each y_i is lexicographically smaller than x. What class can you put those languages?

2. **[Consequences of improving Ryan Williams' recent result]** (10)

   Suppose we are able to show that $\mathsf{DTIME}(t) \subseteq \mathsf{DSPACE}(t^\varepsilon)$ for every $\varepsilon > 0$. Prove that this would imply $\mathsf{P} \neq \mathsf{PSPACE}$.

3. **[Kannan's theorem]** (7 + 5 + 3)

   (i) Fix any constant $c \geq 1$. Show that there is a language $L \in \mathsf{PH}$ that is not in $\mathrm{SIZE}(n^c)$.

[Hint: Can you try and encode "the lexicographically smallest circuit of size n^c that is not computable by circuits of size $n^{c/2}$" as a quantified expression?]

(ii) Show that, for any constant $c \geq 1$, there is a language in $L \in \Sigma_2^P$ that is not in $\mathsf{SIZE}(n^c)$.

[Hint: Either $\mathsf{NP} \subseteq \mathsf{SIZE}(\text{poly})$ or not...]

(iii) Note that this means in particular that, for any constant $c > 0$, we know NP is not computable by circuits of size n^c. Why does this not show that $\mathsf{NP} \not\subseteq \mathsf{SIZE}(\text{poly})$ (which, if you recall, is stronger than saying $\mathsf{P} \neq \mathsf{NP}$)?

4. **[Generalising Karp-Lipton]** (10 + 5)

A *non-deterministic circuit family* is a sequence of circuits $\{C_i(x, y)\}_{i \in \mathbb{N}}$ where C_i is on two sets of inputs with $|x| = i$ and $|y| = \text{poly}(i)$. We will say that the non-deterministic circuit family compute a language $L \subseteq \{0,1\}$ if for every $x \in \{0,1\}^*$ we have

$$x \in L \iff \exists y \ : \ C_{|x|}(x, y) = 1.$$

Let $\mathsf{NSIZE}(\text{poly})$ be the class of all languages that are computable by a non-deterministic circuit family of size $\text{poly}(n)$.

Turns out, $\mathsf{NSIZE}(\text{poly}) = \mathsf{NP}/\text{poly}$.

(a) Show that if $\mathsf{coNP} \subseteq \mathsf{NSIZE}(\text{poly})$, then $\Pi_3 = \Sigma_3 = \mathsf{PH}$. (In other words, it is unlikely that $\mathsf{coNP} \subseteq \mathsf{NP}/\text{poly}$.)

(b) To contrast this, prove that $\mathsf{coNEXP} \subseteq \mathsf{NEXP}$ / poly.

[Hint: Might help to recall the key idea in $\mathsf{NL} = \mathsf{coNL}$ to figure out what advice you would like.]

5. **[Input oblivious NP]** (10)

The complexity class ONP (also referred to as 'oblivious NP') is defined as follows:

> A language L is in ONP if and only there is a deterministic polynomial time machine M such that for all lengths n, there exists $w_n \in \{0,1\}^{\text{poly}(n)}$ such that any n-length string x is in the language if and only if M accepts (x, w_n).

The difference between NP and ONP is that for NP, the 'witness' w can depend on x, but in ONP the witness is the same for all string of that length.

Prove that $\mathsf{ONP} = \mathsf{NP}$ if and only if $\mathsf{NP} \subseteq \mathsf{SIZE}(\text{poly})$.

[Hint: ($\Leftarrow$) should be the obvious direction. As for the other direction, use Karp-Lipton.]

6. **[Circuit complexity of multiplication]** (3 + 3 + 6 + 3)

We say in class that given two n-bit numbers, we can compute their sum in AC^0. In this problem we will understand the complexity of multiplying two n-bit numbers.

(a) Show that multiplying two given n-bit numbers *cannot* be done in AC^0 by reducing Parity_m to it.
(Since we are dealing with *really low-level* classes such as AC^0, your reduction has to be *super*-low-level. Use this problem to come up with a suitable definition for such a "class" of reductions.)

[Hint: $b_2 00 b_1 00 b_0 \times 100100 1 = ?$]

(b) Build an NC^0 circuit (fan-in 2, constant depth; each output bit can therefore depend on just constantly many input bits!) that takes three n-bit numbers x, y, z and output an n-bit number u and an $(n+1)$-bit number v such that $x + y + z = u + v$.

[Hint:
$$\begin{array}{r} 101100 \\ 011010 \\ 110101 \\ \hline 000011 \\ 111000 \end{array}$$
]

(c) Show that given n numbers that are n-bits long each, we can compute their sum using an NC^1 circuit (fan-in two, $O(\log n)$-depth circuits).

(d) Conclude the product of two given n-bit numbers can be computed in NC^1.

7. **[Randomised log-space]** (4 + 6)

In this problem, we are going to try and define the class RL just like we defined the class RP. But we will have to handle a few subtleties.

Just like in the witness definition of NL, we can think of a log-space machine M that is given a special 'random tape' that will be filled with a uniformly random string. Like in the witness definition, this tape will be read-once. We will think of this as a definition of the class RL but we need to address one subtlety.

An important subtlety is what is the halting requirement for an RX (for X replaced by your favourite class)? There are two possible definitions we could think of:

- The machine M must halt on *every* input and *every* setting of the random tape.
- The machine M must halt on *every* input with probability 1 with respect to the random tape contents.

(a) Show that for any time-bounded class X (such as P, or EXP etc.) both the above variants of RX coincide. That is, the distinction is not important when talking about classes such as RP or RE etc.

(b) For space-bounded classes, this becomes an important distinction. Define RL_1 to be the variant where we require the TM to halt on each input with probability 1 over the the random tape contents. Show that RL_1 is infact equal to NL.

Thus, the *right* definition for RL is one where we should insist that every computational path is halting (and not just with probability 1).