Problem Set 3

- Due date: **3 May, 2026** (released on 18 Apr, 2026).
- The points for each problem is indicated on the side. The total for this set is **70** 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).
- The use of LLMs for this problem set is allowed ONLY FOR question 5(a).
- 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. **[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 $\mathsf{P} \neq \mathsf{PSPACE}$.

2. **[$\mathsf{CL} \subseteq \mathsf{ZPP}$]** (3 + 5 + 5 + 2)

   (a) Let M be a $\mathsf{CSPACE}(s(n), c(n))$ machine with k tapes, Q states and tape alphabet Γ. Compute the number of configurations of M on an input x of length n.

   (b) For $\tau \in \Gamma^{c(n)}$ which is an instantiation of the catalytic tape, let $\mathrm{Path}(x, \tau)$ refer to the set of configurations visited by M when run on input x with catalytic tape contents τ. Show that whenever $\tau \neq \tau'$, we have $\mathrm{Path}(x, \tau) \cap \mathrm{Path}(x, \tau') = \emptyset$.

   (c) Prove that $\mathbb{E}_\tau \left[|\mathrm{Path}(x, \tau)| \right] = 2^{O(s(n))}$.

   (d) Conclude that $\mathsf{CL} \subseteq \mathsf{ZPP}$.

3. **[Pebbling games]** (5 + 5)

   Let T be a tree. A *pebbling-game* on T proceeds as follows:

   - You may place a pebble on a leaf at any point.
   - You may remove a pebble from any node at any point.
   - For an internal node, if all its children have a pebble on them, then you may move a pebble from one of its children to that node.

   The goal is to pebble the root of the tree using as few pebbles as possible.

(a) Let T_h be a binary tree of height h (with 2^h leaves). You saw in one of your quizzes that you can pebble T_h using at most $h+1$ pebbles. Formally prove that $h+1$ pebbles is also necessary to pebble T_h.

[Hint: Consider the last time one of the two subtrees had h pebbles.]

(b) Extend your argument to a-ary trees and show that $(a-1)h+1$ pebbles are necessary.

4. **[Reversible traversal of a tree]** (7 + 3)

In class, we considered the following *reversible traversal* of an out-degree 1 graph that was given by the following two functions.

Function `Fwd`(u, i)**:**
  $(v, j) \leftarrow \mathrm{Rot}(u, i)$;
  return $(v, j+1)$;

Function `Back`(u, i)**:**
  $(v, j) \leftarrow \mathrm{Rot}(u, i-1)$;
  return (v, j);

If we begin at some vertex u, show that running `Fwd` repeatedly starting from $(u, 1)$ traverses every other node in the connected component (when considered as an undirected graph) of the vertex u.

Given an example of a graph that is *not* out-degree 1 for which the above function fails to traverse all the nodes in the connected component.

5. **[Subtlety with parallel repetition]** (7 + 2 + 6)

Consider the following simple one-round private coin interactive game. Verifier tosses some coins r to generate $q \in Q$ and sends to the Prover. The Prover answers with an answer $a \in A$. The Verifier accepts if $F(r, q, a) = 1$ (where F is known to both the Prover and Verifier).

Define $\mathrm{val}(G)$ to be the maximum acceptance strategy for any prover P. That is,

$$\mathrm{val}(G) = \max_{P:Q\to A} \Pr_r[F(r, q, P(q)) = 1].$$

Given a game G, consider the t-fold parallel repetition of G (called G^t) as follows:

> Verifier tosses t sets of random coins $r_1, \ldots, r_t$ and generates t questions $q_1, \ldots, q_t$, and sends $(q_1, \ldots, q_t)$ to the Prover. The Prover responds with $(a_1, \ldots, a_t)$. The Verifier accepts if $F(r_i, q_i, a_i) = 1$ for all $i = 1, \ldots, t$.

Now, "obviously" $\mathrm{val}(G^t) = (\mathrm{val}(G))^t$, right? This is true, but it is *not obvious at all!*

(a) Use your favourite LLM and ask it to give you full proof that $\mathrm{val}(G^2) = \mathrm{val}(G)^2$. Do note that it is very likely to make a mistake. If it does, point out the flaw and eventually obtain a full proof. (If it helps, you may try different LLMs and see which one is better.)
Write the final formal proof yourself, and also share the LLM conversation URL(s).

Let us generalise the above to two provers. The Verifier now tosses coins to get r, and generates a pair $(q, q') \in Q \times Q$ and sends q to Prover P and q' to Prover P'. The Provers answer with $a \in A$ and $a' \in A$ respectively. The Verifier accepts if $F(r, q, q', a, a') = 1$.

(The Provers could have decided on a strategy beforehand, but they cannot communicate during the protocol.)

Once again, let $\mathrm{val}(G)$ denote the maximum probability that any pair of Provers can make the verifiers accept. That is,

$$\mathrm{val}(G) = \max_{P,P':Q\to A} \Pr_r[F(r,q,q',P(q),P'(q')) = 1].$$

(b) Consider the following concrete game G. The verifier picks two random bits (b,b') uniformly at random, and sends $q = b$ to the first prover P and $q' = b'$ to the second prover P'. The Provers are expected to answer $(i,\beta) \in \{1,2\} \times \{0,1\}$ (where $i \in \{1,2\}$ and $\beta \in \{0,1\}$; think of the answer (i,β) as asserting "Prover i got bit β"). The Verifier accepts the answers (i,β) and (i',β') from the two Provers if

- $i = i'$ and $\beta = \beta'$,
- Prover i did indeed get bit β.

In other words, their answers must be identical, and their assertion must be correct.

Show that $\mathrm{val}(G) = 1/2$.

(c) Consider the above 2-fold parallel repetition G^2 of the above game. That is, the Verifier picks two pairs of bits (b_1,b_1') and (b_2,b_2') and sends (b_1,b_2) to Prover 1 and (b_1',b_2') to Prover 2. The Prover 1 responds with $(i_1,\beta_1,i_2,\beta_2)$ and Prover 2 responds with $(i_1',\beta_1',i_2',\beta_2')$. The Verifier accepts if their answers were identical, and the two assertions are indeed correct (i.e., Prover i_1 did indeed get β_1 in their first bit, and Prover i_2 did indeed get β_2 in their second bit).

Come up with a strategy for the provers that makes the verifier accepts with probability $1/2$.
(Thereby, you will show that $\mathrm{val}(G^2) = 1/2$, and not $1/4$ as one might expect!)

6. **[Promise problems]** (3 + 1 + 6)

(a) Show that $\mathsf{P}^{\mathsf{NP}\cap\mathsf{coNP}} = \mathsf{NP} \cap \mathsf{coNP}$.

Recall promise problems $\Pi = (\Pi_Y, \Pi_N)$ (given by a set of yes-instances and no-instances). Running with a promise problem as an oracle is a little subtle to define correctly as it is unclear what the oracle should return if queried on a "don't care" input. Thus, we will say that a problem Γ can be solved in polynomial time with oracle access to a promise problem Π to mean that there is an algorithm A such that for *every oracle* $M : \{0,1\}^* \to \{0,1\}$ that solves Π (i.e., M is correct on $\Pi_Y \cup \Pi_N$), it holds that A^M solves Γ.

Consider the following concrete promise problem $\Pi = (\Pi_Y, \Pi_N)$:

$$\Pi_Y := \{(\Phi,\Psi) \ : \ \Phi \in \mathrm{SAT}\ ,\ \Psi \notin \mathrm{SAT}\}$$
$$\Pi_N := \{(\Phi,\Psi) \ : \ \Phi \notin \mathrm{SAT}\ ,\ \Psi \in \mathrm{SAT}\}$$

(b) Show that $\Pi \in \mathsf{pr}\,\mathsf{NP} \cap \mathsf{pr}\,\mathsf{coNP}$

(c) Show that $\mathrm{SAT} \in \mathsf{P}^{\Pi}$.

[Hint: If you want to know if $\Phi \in \mathrm{SAT}$, and you asked $(\Phi\,|_{x_1=0}, \Phi\,|_{x_1=1})$ to the oracle, is the answer helpful?]

(The last part can be extended to show that every promise problem $\Gamma \in \mathsf{pr}\,\mathsf{NP}$ is in $\mathsf{pr}\,\mathsf{P}^{\Pi}$. Thus, it implies that $\mathsf{pr}\,\mathsf{NP} \subseteq \mathsf{pr}\,\mathsf{P}^{\mathsf{pr}\,\mathsf{NP}\cap\mathsf{pr}\,\mathsf{coNP}}$. An analogous statement for languages seems unlikely as $P^{\mathsf{NP}\cap\mathsf{coNP}} = \mathsf{NP} \cap \mathsf{coNP}$.)