End-semester exam: Take-home part

- The points for each problem is indicated on the side. The total for this set is **80** points. You are expected to answer **any 60 points for full credit**.
- The deadline for **Thursday, 15th May 2025, 7pm (IST)**. This is a hard deadline! Turn in your solutions as PDFs (LaTeXed or scanned/hand-written etc.) via email.
- Collaboration with other students is **not allowed**!
- You can refer to the scribe notes or any other notes you have from the class. All other sources are disallowed (except one question that explicitly asks you to use ChatGPT).
- Even if you do not manage to solve some questions, explain your attempts and partial thoughts for partial credit.
- Be clear in your writing.

---

1. A polynomial-time *single-query Turing reduction* from a language $L_1 \subseteq \{0,1\}^*$ to $L_2 \subseteq \{0,1\}^*$ is given by a polynomial time computable function $f : \{0,1\}^* \to \{0,1\}^* \times \{0,1\}$ with the following guarantee:

   For any $x \in \{0,1\}^*$, let $f(x) = (y, b)$. Then,

   $$x \in L_1 \iff \begin{cases} y \in L_2 & \text{if } b = 0 \\ y \notin L_2 & \text{if } b = 1 \end{cases}$$

   In other words, you can solve an instance of L_1 by making a single oracle query to L_2 and possibly flipping the answer (the first coordinate tells you what you should query, and the second coordinate tells you whether you should flip the final answer).

   (a) **(10 points)** Generalise the proof of Mahoney's theorem and show the following:

   Suppose L is a sparse-language that is NP-hard under polynomial-time single-query Turing reductions, show that $\mathsf{P} = \mathsf{NP}$.

   [Hint: Suppose you have a bag of formulas $\varphi_1, \ldots, \varphi_m$, run the reduction on $\varphi_1 \vee \varphi_i$ for $i = 2, \ldots, m$ and prune.]

   (b) **(10 points)** Suppose we have deterministic polynomial time TM M that *almost solves* SAT in the following sense — $L(M) \Delta \text{SAT}$ is a sparse set (where $A \Delta B$ refers to the symmetric difference, namely $(A \setminus B) \cup (B \setminus A)$). Prove that this implies $\mathsf{P} = \mathsf{NP}$.

[Hint: There is an obvious sparse set to consider. Can you show that it is NP-hard under polynomial time single-query reductions?]

2. Recall that $\mathsf{CSPACE}(w(n), c(n))$ is the set of languages that can be computed by catalytic TMs with $O(w(n))$-sized work tape and $O(c(n))$-sized catalytic tape; and $\mathsf{CL} = \mathsf{CSPACE}(\log n, \mathrm{poly}(n))$. The goal of this question is to show that $\mathsf{CL} \subseteq \mathsf{ZPP}$.

   (a) **(3 points)** Fix a catalytic TM M and an input x of length n. Let the size of the catalytic tape used by the machine M on x be s and let the size of the work tape used be w. Assume that the tape alphabet (for both work and catalytic tapes) is Γ.

   Compute the total number vertices in the configuration graph (where configurations include the contents of the catalytic tape also) as a function of n, w, s, $|\Gamma|$ and any other relevant parameters. (Get the right dependence in s; you can be loose with the other parameters).

   (b) **(12 points)**
   For any $\tau \in \Gamma^s$, let S_τ be the sequence of configurations when the machine M is run on input x with catalytic tape initialised to τ.
   Prove that $\mathop{\mathbb{E}}_{\tau \in \Gamma^s}[|S_\tau|] = \mathrm{poly}(n, 2^w)$.

   [Hint: What can you say about S_τ and S'_τ when $\tau \neq \tau'$?]

   (c) **(5 points)** Construct a ZPP-algorithm for $L(M)$.

3. 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.

   Let T_h be a binary tree of height h (with 2^h leaves).

   (a) **(3 points)** Construct a pebbling strategy to pebble the root of T_h using at most $h + 1$ pebbles. (That is, at no point in the sequence should the tree have more than $h + 1$ pebbles placed on it.)

   (b) **(2 points)** How many pebbles do you need to pebble a d-ary tree of height h (which has d^h leaves)?

   (c) **(10 points)** Formally prove that any strategy to pebble the root of T_h will require at least $h + 1$ pebbles.

   [Hint: Consider the last time the left child of the root was pebbled (if that was the earlier one) and try to induct.]

   (d) **(5 points)** Using the above, can elaborate on why it was conjectured that $\mathrm{TEP}_{h,k}$ requires space $\Omega(hk)$? (until Cook and Mertz proved otherwise, of course!)

4. A function $f : \{0,1\}^n \to \{0,1\}$ is said to be a *monotone* function if flipping an input bit from 0 to 1 can only increase the function value. Formally, if $x \succeq y$ (which denotes that $x_i \geq y_i$ for all i) then $f(x) \geq f(y)$. Clearly, $\text{Majority}_n : \{0,1\}^n \to \{0,1\}$ is a monotone function.

   As a result of one of the questions in problem set 3, you could construct an NC^1 circuit for Majority using iterated addition, however the circuit uses $\neg$ gates even though Majority is a monotone function. This question aims at showing that Majority has a *monotone* NC^1 circuit (i.e., an NC^1 circuit without any $\neg$ gates) using the probabilistic method.

   Here is a partial sketch.

   > Let $x_1, \ldots, x_n$ be bits in a certain layer (to begin with, say these are the inputs). We will build a new set of n bits $y_1, \ldots, y_n$ as follows — each y_i is $\text{Majority}_3(x_{i_1}, x_{i_2}, x_{i_3})$ where each i_1, i_2, i_3 are independently and uniformly at random chosen from $[n]$.
   > We repeat the above for $O(\log n)$ layers, and assign one of the bits of the last layer as the output.

   Intuitively, the majority value of $x_1, \ldots, x_n$ is *more pronounced* in $y_1, \ldots, y_n$.

   (a) **(2 points)** Construct a *monotone* constant-depth fan-in 2 circuit for Majority_3.

   (b) **(3 points)** Consider $\text{Majority}_3 : \{0,1\}^3 \to \{0,1\}$. Suppose each x_i was set to 1 independently with probability p. Compute $\Pr[\text{Majority}_3(x_1, x_2, x_3) = 1]$ as a function of p. (For example, if it was AND_3 instead, this function would have been p^3).

   (c) **(5 points)** Let $F(p)$ be the function in the above part. Suppose you are told that $F^{(k)}(p)$ (which is F applied to itself k times) has the following property for $k = O(\log n)$:

   $$F^{(k)}(p) = \begin{cases} \geq 1 - \frac{1}{2^{O(n)}} & \text{if } p \geq \frac{1}{2} + \frac{1}{n}, \\ \leq \frac{1}{2^{O(n)}} & \text{if } p \leq \frac{1}{2} - \frac{1}{n} \end{cases}.$$

   Use this to show that Majority_n indeed has a monotone circuit of fan-in 2 and depth $O(\log n)$.

   (d) **(10 points)** Prove the above property with the help of ChatGPT. Start with a prompt as follows:

   > Consider the function F(p) = (fill this in). I am trying to show that, if I start with p = 1/2 + eps and repeatedly keep applying F to it, then it very quickly approaches 1. And if I start with p = 1/2 - eps, then it very quickly approaches 0. I need your help in proving this formally.

   It will give you a response that isn't quite correct, but is half-way there. Find out what ChatGPT missed, let it know and nudge it towards a complete solution. Use these conversations to give a formal proof of the above. (Please also submit the transcript of your conversation with ChatGPT.)

■