CS 535: Complexity Theory, Fall 2026

Homework 9

Due: 11:59PM, Friday, November 6, 2026.

Reminder. Homework must be typeset with LaTeX preferred. Problems 1, 2, and 3 are required. Problem 3 is optional.

Problem 1 (*Required* Characterizing $\mathbf{ZPP}$)**.** Recall that we defined the class $\mathbf{ZPP}$ to consist of languages that are decidable by zero-error probabilistic TMs in expected polynomial time. In this problem, you will explore two different alternative characterizations of this class.

(a) First, let us complete the proof we started in class (Thursday, 10/29) that $\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{coRP}$. We already showed in class that $\mathbf{ZPP} \subseteq \mathbf{RP} \cap \mathbf{coRP}$. For the opposite containment, let $L \in \mathbf{RP} \cap \mathbf{coRP}$ be decided by an $\mathbf{RP}$ algorithm M_0 and a $\mathbf{coRP}$ algorithm M_1, each running in (worst-case) time $p(n)$ for some polynomial p. Show that the following is a zero-error randomized algorithm deciding L in expected polynomial time, and thus $\mathbf{RP} \cap \mathbf{coRP} \subseteq \mathbf{ZPP}$.

> On input x:
> Repeat the following indefinitely:
>
> 1. Run M_0 on input x. If it accepts, accept; else, proceed to step 2.
> 2. Run M_1 on input x. If it rejects, reject; else, continue.

(b) An *abstaining* probabilistic TM is like a normal probabilistic TM for a decision problem, except that it may output any of the answers $0, 1$, or $?$ (standing for "I abstain from answering"). Show that a language $L \in \mathbf{ZPP}$ if and only if there exists an abstaining probabilistic TM M running in *worst-case* polynomial time such that

- If $x \in L$, then on input x, the TM M always outputs either 1 or $?$,
- If $x \notin L$, then on input x, the TM M always outputs either 0 or $?$, and
- For every input x, we have $\Pr[M(x) = ?] \leq 1/3$.

That is, when M produces an answer of 0 or 1, it is always correct, and on every input it abstains from answering with probability at most $1/3$.

Problem 2 (*Required* $\mathbf{NP}$, $\mathbf{BPP}$, and $\mathbf{RP}$)**.**

(a) Suppose $\mathbf{NP} \subseteq \mathbf{BPP}$. Show that SearchSAT can be solved in randomized polynomial-time. That is, show that there is a probabilistic poly-time algorithm M such that for all satisfiable CNF formulas φ, we have that $M(\varphi)$ outputs a satisfying assignment to φ with probability at least $2/3$.

(b) Use part (a) to conclude that if $\mathbf{NP} \subseteq \mathbf{BPP}$, then $\mathbf{NP} = \mathbf{RP}$.

Problem 3 (*Required* $\mathbf{NP}$, $\mathbf{BPP}$, and $\mathbf{ZPP}$)**.** Show that $\mathbf{BPP} \subseteq \mathbf{ZPP}^{\mathbf{NP}}$.

Hint: Follow the proof of the Sipser-Gács-Lautemann Theorem. Feel free to reuse the setup and notation from that proof if it's helpful. Guess the shifts at random and use the $\mathbf{NP}$ oracle to check them.

Problem 4 (*Optional* $\mathbf{SearchBPP} \neq \mathbf{SearchP}$)**.** While many complexity theorists believe that $\mathbf{BPP} = \mathbf{P}$, in this problem you will show that the search analogs of these classes are different *unconditionally*. But first we need to define these classes. Recall that we model a "search problem" by a function $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$. Say an input x is "valid" if there exists a y such that $f(x,y) = 1$. A search problem is polynomially-bounded if there exists a polynomial p such that $|y| \leq p(|x|)$ for all x, y such that $f(x,y) = 1$.

- A search problem f is in the class $\mathbf{SearchP}$ if it is polynomially-bounded and there is a deterministic polynomial-time algorithm that, given as input a valid string $x \in \{0,1\}^*$, outputs a string $y \in \{0,1\}^*$ such that $f(x,y) = 1$. (If x is invalid, the algorithm can do whatever it wants.)

- A search problem f is in the class $\mathbf{SearchBPP}$ if it is polynomially-bounded and there is a probabilistic polynomial-time algorithm A such that, for every $\varepsilon > 0$, and every valid $x \in \{0,1\}^*$ we have $\Pr[f(x,y) = 1] > 1 - \varepsilon$, where the probability is taken over the randomness in computing $y \leftarrow A(x, 1^{1/\varepsilon})$.

Define the search problem MISS by taking $\mathsf{MISS}(x,y) = 1$ if and only if $|y| = |x|$ and $M_x(x)$ does not halt and output y within at most $2^{|x|}$ steps of computation. As usual, you can assume that Turing machine encodings are "nice" in the sense that every binary string represents some TM and that every TM is encoded by infinitely many strings.

(a) Use a diagonalization argument to show that $\mathsf{MISS} \notin \mathbf{SearchP}$.

(b) Show that $\mathsf{MISS} \in \mathbf{SearchBPP}$. *Hint:* An You will need to do something different depending on whether ε is greater or less than, say, 2^{-n}.