The more challenging problems are marked with $*$. Be concise when describing a Turing machine. It is like writing pseudocodes. It suffices to present the most important ideas behind your Turing machines. You do not need to give all the details, e.g., the set of states and the transition function. Check the course website for more info about homeworks. CC: Computational Complexity. TC: Introduction to the Theory of Computation. MA: Computational Complexity: A Modern Approach. LC: Lectures in Computational Complexity (at `http://pages.cs.wisc.edu/~jyc/book.pdf`).

1. Problem 11.5.18 on page 275 of CC: Show that, if NP $\subseteq$ BPP, then RP = NP. (That is, if SAT can be solved by randomized machines, then it can be solved by randomized machines with no false positives, presumably by computing a satisfying truth assignment as in Example 10.3.)

2. Let $0 < \epsilon_1 < \epsilon_2 < 1$ denote two constants. Let $D(\cdot, \cdot)$ be a deterministic polynomial-time computable Boolean function, and let L be a language (the setting so far is exactly the same as the definition of BPP). D and L satisfy the following property: Given any $x \in \{0,1\}^n$, if y is sampled uniformly at random from $\{0,1\}^m$ for some m polynomial in n, then

$$x \in L \implies \Pr_{y \in \{0,1\}^m}\big[D(x,y) = 1\big] \geq \epsilon_2 \quad \text{and} \quad x \notin L \implies \Pr_{y \in \{0,1\}^m}\big[D(x,y) = 1\big] \leq \epsilon_1.$$

Show that $L \in$ BPP. (Note that ϵ_2 can be smaller than 1/2. Use the Chernoff bound.)

3. Similar to P/poly, one can define P/$\log n$, where the advice string is of length only $O(\log n)$ for input size n. Show that, if SAT $\in$ P/$\log n$, then P = NP. (Hint: Self-reducibility.)

4*. Show that, if PSPACE $\subseteq$ P/poly, then PSPACE $= \Sigma_2^{\mathrm{P}}$. (Hint: Use self-reducibility to "implicitly" build a winning strategy for the existential player in the TQBF game.)