Theory of Computation

Homework 5

Due: 2012/01/03

Problem 1. Show that if NP $\subseteq$ BPP then NP = RP. (Hints: It suffices to show SAT $\in$ RP.)

Proof. As RP $\subseteq$ NP (see the slides), it suffices to show that NP $\subseteq$ RP. We prove this claim by showing that if NP $\subseteq$ BPP, then SAT $\in$ RP. Let a formula ϕ with n variables $x_1,\ldots,x_n$, be the input. Note that ϕ is satisfiable iff there exists a truth assignment for $x_1,\ldots,x_n$ such that $\phi(x_1,\ldots,x_n) = 1$. Let $\boldsymbol{A}$ be a BPP algorithm with error probability at most 2^{-k} (see the slides pp. 526—528) for SAT, where $k = |\phi|$ is the length of the formula ϕ. Such an $\boldsymbol{A}$ exists because of the assumption that SAT $\in$ BPP. We first run $\boldsymbol{A}$ on ϕ. If $\boldsymbol{A}$ rejects, we reject. Otherwise, we try to construct a satisfying assignment for ϕ one variable at a time. We initialize x_1 to 0, and then call $\boldsymbol{A}$ to determine if the resulting formula is satisfiable: if $\boldsymbol{A}$ returns "accept", then we permanently set x_1 to 0; otherwise, we set x_1 to 1. We then proceed with x_2 similarly. If we manage to construct a satisfying assignment at the end, then we verify this assignment for ϕ. If $\phi(x_1,\ldots,x_n) = 1$, then we accept; otherwise, we reject.

Here is the analysis. If ϕ is unsatisfiable, then we always reject either because $\boldsymbol{A}$ rejects in the process or we do not arrive at a satisfying assignment at the end. On the other hand, suppose ϕ is satisfiable. We proceed to show that we accept with probability at least 1/2. We invoke $\boldsymbol{A}$ a total of $n + 1$ times. If ϕ is satisfiable and $\boldsymbol{A}$ returns "accept" each time only for an assignment for variable x_i which is part of a satisfying assignment, then we end up with a satisfying assignment. We now show that the probability that at least one of the $n + 1$ invocations returns "reject" for an assignment for variable x_i which is part of a satisfying assignment is at most 1/2. The probability that an invocation of $\boldsymbol{A}$ returns does so is at most 2^{-k}. So the probability that we encounter it is at most $(n + 1)\cdot 2^{-k}$, which is at most 1/2 because $n + 1 \leq k$. Since both the algorithm $\boldsymbol{A}$ and the construction of satisfying assignment run in polynomial time, the whole procedure clearly runs in polynomial time.

□

Problem 2. Show that BPP ⊆ PSPACE.

Proof. Let M be a probabilistic TM that runs in polynomial time. We can modify M such that it makes exactly n^k coin tosses on each branch of its computation, for some constant k. Note that there are a total of $2^{(n^k)}$ computation paths. Hence, the problem of determining the probability that M accepts its input reduces to counting how many branches, B, are accepting and comparing this number with $P = (3/4)\cdot 2^{(n^k)}$. If $B \geq P$, then we accept; otherwise, we reject. This deterministic task can be performed in polynomial space by generating all possible paths sequentially following M's program but recycling the space used by the previous path.

☐