Sub-linear Algorithms	October 19, 2015

Homework 0

Lecturer: Ronitt Rubinfeld — *Due Date: October 26, 2015*

Homework guidelines: The following problems are for your understanding. Do not turn it in, but make sure you can solve it.

1. Show that given any algorithm A that runs in time $T(n)$ on inputs of size n with probability of error $1/4$, one can convert it into a new algorithm B that runs in time $O(T(n)\log 1/\beta)$ with probability of error at most β. (Hint: run A $O(\log 1/\beta)$ times and take the majority answer. Use Chernoff bounds.)

2. You are given an approximation scheme $\mathcal{A}$ for f such that $Pr[\frac{f(x)}{1+\epsilon} \leq \mathcal{A}(x) \leq f(x)(1+\epsilon)] \geq 3/4$, and $\mathcal{A}$ runs in time polynomial in $1/\epsilon, |x|$. Construct an approximation scheme $\mathcal{B}$ for f such that $Pr[\frac{f(x)}{1+\epsilon} \leq \mathcal{B}(x) \leq f(x)(1+\epsilon)] \geq 1-\delta$, and $\mathcal{B}$ runs in time polynomial in $\frac{1}{\epsilon}, |x|, \log\frac{1}{\delta}$.

3. (Coupon Collector Problem). Given a die with n sides. What is the expected number of times you need to roll the die in order to see each of the n sides? (Hint: Given that you saw i sides, how many times do you need to roll the die to see the $(i+1)^{st}$ side? Then use linearity of expectation.)

4. Check that you followed the lecture details:
   - On slide 31, why do you need at least $\Omega(n^{1/2})$ samples to see a problem?
   - Given a coin with bias p (i.e. probability of heads is p), what is the expected number of coin tosses t to see your first heads. Use Markov's inequality to show that the probability of tossing ct coins and never seeing a heads is at most $1/c$. What do you get using Chebyshev's? Chernoff?
   - Use the previous to argue that there is a proper choice of constant c such that repeating the test c/ϵ times satisfies the following: If at least ϵn of the i's are bad, then the test fails with probability at least $3/4$.
   - (modification of algorithm on slide 40) Show that picking $\theta(\sqrt{n})$ samples and comparing *all pairs* is sufficient/necessary for finding a duplicate in the special input on that slide.