Randomized Algorithms

CME309/CS365, Winter 2012-2013, Stanford University
Instructor: Ashish Goel
Handout 3: Homework 1. Due 1/31/2013 @ 11:00am.

Collaboration policy: Limited collaboration is allowed you can discuss the problem with other students in this class but cannot collaborate on writing the actual final answer. Please do not look at someone elses solution and do not share your solution with anyone else.

Late HW policy: You are allowed one late day for any one HW (i.e. submit the HW on Friday as opposed to Thursday).

Problem numbers refer to the text-book (Motwani and Raghavan).

Non-letter grade students: please do any three problems. If you do more, we will grade any three.

1. Problem 1.8
2. Problem 1.9
   Hint: Define a random variable X_i that indicates whether the size of the set S reduces by a constant factor (e.g. $\frac{1}{4}$) in the i-th iteration.
3. Problem 2.3
4. Problem 4.14
   Hint: Try an analysis similar to the hint for Problem 1.9
5. Let $X_1, X_2, \ldots, X_n$ be i.i.d. geometric random variables and $S = \sum_i X_i$. Give Chernoff-like bounds on S. Specifically, find bounds for $\mathbf{Pr}[S > (1+\delta)\mathbb{E}[S]]$ and $\mathbf{Pr}[S < (1-\delta)\mathbb{E}[S]]$.
   Hint: A geometric random variable with mean $\frac{1}{p}$ can be interpreted as the number of coin flips needed to get a Heads, when each coin flip gets Heads with probability p.
6. Exercise 4.2 (Probability and Computing): We have a standard six-sided die. Let X be the number of times that a 6 occurs over n throws of a die. Let p be the probability of the event $X \geq \frac{n}{4}$. Compare the best upper bounds on p that you can obtain using Markov's inequality, Chebyshev's inequality, and Chernoff bounds.
7. Suppose you have a set of n numbers. You want to find the smallest element of this set, but you are working with a computer where writing to memory is much more expensive than reading from memory or doing computation. Design a randomized algorithm that computes the minimum of the given numbers with high probability, while only using $O(\log n)$ writes to memory when you are allowed to sample with replacement from S as a unit time operation, and when you are allowed to sample without replacement from S as a unit time operation. For these two cases, your algorithm should take time $O(n \log n)$ and $O(n)$ respectively. Any writes done by the sampling data structure are not charged to your algorithm.