6.856 — Randomized Algorithms

David Karger

Handout #20, November 14, 2002 — Homework 11, Due 11/20

M. R. refers to this text:
Motwani, Rajeez, and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge: Cambridge University Press, 1995.

1. Based on MR 11.2. Consider the following variant of the DNF counting algorithm from class. For the t-th trial, pick a satisfying assignment a uniformly at random from the *disjoint* union of satisfying assignments, just as described in class. But now, instead of checking whether a is the "first" copy of itself, try the following. Let N be the number of assignments in the disjoint union. Let c_a be the number of clauses that a satisfies. Define $X_t = 1/c_a$.

   (a) Prove that $N \cdot E[X_t]$ is the number of satisfying assignments to the DNF formula

   (b) Prove that $O(m\mu_{\epsilon\delta})$ trials (and computation of the resulting $\sum X_t$) suffice to DNF-count to within $(1 \pm \epsilon)$ with probability $1 - \delta$. **Hint:** use (without proof) the Chernoff bound generalization from MR 4.7.

   (c) Once a has been chosen in the previous problem, give an algorithm for quickly estimating c_a to within $(1 \pm \epsilon)$. Argue that this is sufficient to give us the $(1 \pm \epsilon)$ approximation for DNF-counting.

   (d) Analyze the running time of the DNF counting algorithm using the above scheme, in terms of the number of clauses evaluated. Assuming all clauses are the same size, what is that actual running time of the scheme?

   (e) (Optional) Justify the assumption that all clauses are the same size to within a logarithmic factor, thus extending the runtime analysis to arbitrary formulae.

2. We showed how to estimate the number of satisfying assignments for a DNF formula. In some cases, we might want to estimate the probability of certain events occurring *when the formula is true.* An obvious way to do this is to generate and test a large sample of satisfying assignments. So consider the following algorithm for *generating a satisfying assignment uniformly at random.* Repeat the following loop until an assignment is output:

   - As for counting, start by choosing an assignment at random from the disjoint union of satisfying assignments.
   - Compute the coverage (number of satisfied clauses) c for this assignment.
   - Output the assignment with probability $1/c$

   Show that this works. In particular:

(a) Prove that the expected time to output an assignment is polynomial in the formula size.

(b) Prove that the output assignment is chosen uniformly at random from among all possible assignments. Be careful: remember that you need to condition on the fact that the assignment was output!

The fact that we can generate satisfying assignments (exactly) uniformly at random, while we can only approximately count (unless $\mathcal{P} = \sharp\mathcal{P}$), suggests that generating is easier than counting in general.

(c) Suppose we are willing to settle for *approximately* uniform generation, in which with probability $1 - \delta$ each assignment is output with probability $(1 \pm \epsilon)$ times uniform. Recall from the previous problem that we can estimate c_a quickly. Use this method to show that you can approximately generate a solution in $\tilde{O}(m\mu^2_{\epsilon\delta})$ time.