6.842 Randomness and Computation March 9, 2020

Homework 3

Lecturer: Ronitt Rubinfeld *Due Date: April 2, 2020*

Homework guidelines: You may work with other students, as long as (1) they have not yet solved the problem, (2) you write down the names of all other students with which you discussed the problem, and (3) you write up the solution on your own. No points will be deducted, no matter how many people you talk to, as long as you are honest. If you already knew the answer to one of the problems (call these "famous" problems), then let me know that in your solution writeup – it will not affect your score, but will help me in the future. It's ok to look up famous sums and inequalities that help you to solve the problem, but don't look up an entire solution.

1. You are given a *satisfiable* 2-SAT (CNF) formula $\phi(X_1, \ldots, X_n)$. For example,

   $$\phi(X_1, X_2, X_3) = (X_1 \vee \neg X_2) \wedge (X_2 \vee \neg X_3) \wedge (\neg X_1 \vee \neg X_3)$$

   is satisfied by setting all three variables to *false*.

   Consider the following algorithm for finding a satisfying assignment:

   - Start with an arbitrary assignment of literals to values

     $$\langle X_1 = x_1, X_2 = x_2, \cdots, X_n = x_n \rangle,$$

     where $x_i \in \{true, false\}$. If it's satisfying, output it and halt.
   - Repeat s times:
     - Pick an arbitrary *un-satisfied* clause C_k that involves the literals X_{k_1} and X_{k_2}.
     - Pick one of the two literals $\{X_{k_1}, X_{k_2}\}$ in C_k, uniformly at random.
     - Complement (flip between $\{true, false\}$) the current value of the chosen literal.
     - If the new assignment satisfies the formula ϕ, output the assignment and halt.

   Show that if we pick s to be $c \cdot n^2$, for a large enough constant c, this algorithm will output a satisfying assignment with probability at least 3/4.

   Note that we assume ϕ is satisfiable.

   Hint: Let $\mathbf{x}^* = \langle x_1^*, x_2^*, \cdots, x_n^* \rangle$ be a satisfying assignment, and let $\mathbf{x} = \langle x_1, x_2, \cdots, x_n \rangle$ be the assignment chosen by the algorithm. Define $d(\mathbf{x}^*, \mathbf{x})$ to be the number of locations at which $\mathbf{x}^*$ and $\mathbf{x}$ differ. What happens to $d(\mathbf{x}^*, \mathbf{x})$ in each iteration of the algorithm?

2. Given a probability distribution p over domain A such that $|A| = n$. Define the *collision probability* of p,

   $$c(p) \equiv Pr_{i,j \in_p A}[i = j] = \sum_{i \in A} p_i^2$$

(where the notation $i \in_p A$ denotes that i is chosen according to distribution p from the domain A). Define the L_1 distance between p, q as

$$|p-q|_1 = \sum_{i \in A} |p_i - q_i|$$

Define the L_2 distance between p, q as

$$||p-q||_2 = (\sum_{i \in A} (p_i - q_i)^2)^{1/2}$$

(Note that L_1, L_2 distances are defined for any pairs of real-valued vectors, not just probability distribution vectors). Recall the Cauchy-Schwartz inequality that $|v|_1 \leq \sqrt{d}||v||_2$ (where d is the dimension of the vector). Let U_X denote the uniform distribution over set X.

Let H be a family of pairwise independent hash functions mapping A to T such that $|A| = n$ and $|T| = t$. Assume that the functions in H are indexed by elements in the set B (i.e., each element in B corresponds to exactly one hash function).

Let W be any subset of A. Let distribution q over $B \times T$ be defined as follows: Choose h uniformly from H and x chosen uniformly from W, then output the pair $\langle h, h(x) \rangle$.

(a) (warmup!) For h chosen uniformly from H, say that x, y are a colliding pair if $h(x) = h(y)$. Show that the expected number of colliding pairs is $\binom{n}{2} \cdot \frac{1}{t}$

(b) For any distribution p over the set A, show that if $c(p) \leq (1+\epsilon^2)/|A|$ then $|p - U_A| \leq \epsilon$.

(c) For q defined as above, show that $c(q) \leq \frac{1+|T|/|W|}{|B \times T|}$

(d) Using the previous two items, show that $|q - U_{B \times T}| \leq \sqrt{|T|/|W|}$.

3. The NP-complete problem CIRCUIT-SAT takes as input a description of a boolean circuit C (assume that the gates are two-input “and”, “or” gates or “not” gates) and asks if there is any set of inputs $x = x_1, \ldots, x_r$ such that $C(x) = 1$. So, $L_{\text{CIRCUIT-SAT}} = \{C \mid C \text{ describes a circuit with a satisfying assignment } x \text{ such that } C(x) = 1\}$. Suppose C is a description of a circuit which is *guaranteed* to either have only one solution or to have no solutions at all. Our goal in this problem is to show that determining whether $C \in L_{\text{CIRCUIT-SAT}}$ cannot be much easier than the general CIRCUIT-SAT problem.

   Let us first define the problem Π as follows: Given circuit C which takes an r-bit input, a polynomial time computable function h mapping $\{0,1\}^r$ to a set of values T, and a value $\alpha \in T$, is there an input y to C such that $C(y) = 1$ and $h(y) = \alpha$?

   We say that algorithm A *unique solves* Π, if for all inputs (C, h, α) with no satistying assignments y, A outputs “no”, and for all inputs (C, h, α) with exactly one satisfying assignment y, A outputs “yes”. Note that for any (C, h, α) which has more than two satisfying assignments, “no” or “yes” is a perfectly legal answer.

   Prove that if there is a randomized one-sided error polynomial time algorithm A which unique solves Π, then $RP = NP$. To do this, design an algorithm $B \in RP$ that decides membership in CIRCUIT-SAT, using oracle calls to A.

Hint: Let N be the number of satisfying assignments to the CIRCUIT-SAT instance. First show how to design algorithm B_k that decides membership in CIRCUIT-SAT, using oracle access to A when it is guaranteed that either $N = 0$ or $2^{k-1} \leq N \leq 2^k$. You might want to recall the result you proved in the first part of the previous homework problem.