Homework 7

Out: 17 Mar. Due: 24 Mar.

Submit your solutions in pdf format on Gradescope by **5pm on Friday, March 24**. *Solutions may be written either in LaTeX (with either machine-drawn or hand-drawn diagrams) or* **legibly** *by hand. (The LaTeX source for this homework is provided in case you want to use it as a template.) Please be sure to begin the solution for each problem on a new page, and to tag each of your solutions to the correct problem! Per course policy, no late solutions will be accepted. Take time to write* **clear** *and* **concise** *answers; confused and long-winded solutions may be penalized. You are encouraged to form small groups (two to four people) to work through the homework, but you* **must** *write up all your solutions on your own. Depending on grading resources, we reserve the right to grade a random subset of the problems and check off the rest; so you are advised to attempt all the problems.*

1. In this question we show how to construct a family of d-wise independent random variables over $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$ for any prime p and any d. This is a generalization of the construction of pairwise independent r.v.'s discussed in Lecture 16 and in MU Lemma 15.2.

Define a random polynomial of degree $d-1$ over $\mathbb{Z}_p$ by picking the d coefficients $a_0, a_1, \ldots, a_{d-1}$ independently and u.a.r. from $\mathbb{Z}_p$ and letting

$$f_{\boldsymbol{a}}(x) = a_0 + a_1 x + \ldots + a_{d-1}x^{d-1} \mod p.$$

Here we are using $\boldsymbol{a}$ to denote the set of coefficients $a_0, a_1, \ldots, a_{d-1}$. We will show that the family of random variables $\{f_{\boldsymbol{a}}(x) : x \in \mathbb{Z}_p\}$ is uniform and d-wise independent over $\mathbb{Z}_p$. Make sure you understand this family of r.v.'s before proceeding!

(a) How many random variables are there in this family?

(b) Prove that the family is uniform over $\mathbb{Z}_p$. [HINT: This involves showing that $\Pr[f_{\boldsymbol{a}}(x) = y] = \frac{1}{p}$ for all $x, y \in \mathbb{Z}_p$. Note that the probability is over the random choice of coefficients $\boldsymbol{a}$. Use the principle of deferred decisions.]

(c) Prove that the family is d-wise independent. [HINT: This involves showing that $\Pr[(f_{\boldsymbol{a}}(x_1) = y_1) \cap (f_{\boldsymbol{a}}(x_2) = y_2) \cap \ldots \cap (f_{\boldsymbol{a}}(x_d) = y_d)] = \frac{1}{p^d}$ for any distinct $x_1, \ldots, x_d \in \mathbb{Z}_p$ and any $y_1 \ldots, y_d \in \mathbb{Z}_p$, where again the probability is over the choice of $\boldsymbol{a}$. Recall that any d points uniquely define a polynomial of degree $d-1$ over any field.]

2. In this question we will use the d-wise independent family constructed in the previous question in order to de-randomize the Ramsey theory construction we discussed in Lecture 12 (see also MU Theorem 6.1). Recall that, when $n \le 2^{k/2}$, there exists a 2-coloring of the edges of the complete graph K_n in which there is no monochromatic k-clique; call such a 2-coloring "k-good." We proved this by showing that, if we 2-color the edges independently and u.a.r., then the resulting random coloring is k-good with non-zero probability.

(a) Let $m = \binom{n}{2}$ and $d = \binom{k}{2}$. Let $2m > p \ge m$ be prime (such a prime always exists). Suppose we instead 2-color the edges of K_n with *d-wise independent* (rather than fully independent) random variables. (To do this, we can use the construction over $\mathbb{Z}_p$ from the previous question, and just project the values onto {red, blue} by taking the result mod 2, ignoring the minor detail that p is odd.) Show that the resulting coloring is good with non-zero probability.

(b) Why do we need to take $p \ge m$?

(c) Show how to use these d-wise independent r.v.'s to obtain a *deterministic* algorithm that finds a k-good 2-coloring in polynomial (in n) time, for any fixed k. [HINT: What is the size of the sample space in part (a)?]

(d) Briefly explain how the algorithm of part (c) can be run in parallel on a polynomial number of processors in $O(\log n)$ time. [HINT: You may assume that s processors can combine their results in $O(\log s)$ time.]

3. Consider the problem of deciding whether two integer *multisets* S_1 and S_2 are identical (in a multiset, each element can appear multiple times) in the sense that each integer occurs the same number of times in both sets. This problem can obviously be solved by sorting in $O(n \log n)$ time, where n is the cardinality of the multisets. In this problem we will consider a more efficient randomized algorithm based on hashing. Here is the algorithm:

- Hash each element of S_1 into a hash table with cn counters (where $c > 1$ is a constant), using some 2-universal family of hash functions. The counters are initially 0, and the ith counter is incremented each time the hash value of an element is i. Using another table of the same size and *the same* hash function, do the same for S_2.
- If the ith counter in the first table matches the ith counter in the second table for all i, output "yes"; otherwise, output "no".

(a) What is the running time of this algorithm? Assume that hashing and arithmetic operations (for incrementing and comparing counters) take constant time.

(b) Verify that if S_1 and S_2 are identical, the algorithm outputs "yes" with probability 1.

(c) Show that if S_1 and S_2 are not identical, the algorithm outputs "no" with probability at least $1 - 1/c$. You may assume for simplicity that S_1 and S_2 are disjoint (indeed, this is WLOG because we can remove elements common to S_1 and S_2 in the analysis). [HINT: suppose S_1 and S_2 are disjoint. Fix some element $x \in S_1$. Now show that with probability $1 - 1/c$ over h, the $h(x)$th counter for S_2 is 0.]

4. Let L be a language (which we can think of as a decision problem, where "yes" instances correspond to strings $x \in L$ and "no" instances to strings $x \notin L$). Suppose we have a randomized algorithm $\mathcal{A}$ for L with one-sided error; i.e., on any input x,

(i) if $x \in L$ then $\mathcal{A}(x)$ outputs "yes" with probability at least $\frac{1}{2}$;
(ii) if $x \notin L$ then $\mathcal{A}(x)$ outputs "no" with probability 1.

As we know very well, we can reduce the error probability by performing repeated independent trials of $\mathcal{A}$; to get the error probability down to δ, we need $\lceil \log_2(\delta^{-1}) \rceil$ trials, which requires $O(t \log(\delta^{-1}))$ random bits. In this problem we will see how to use pairwise independence to achieve error probability δ using only $O(t)$ random bits (for any $\delta \geq 2^{-t}$).

To do this, it is helpful to think of $\mathcal{A}$ as taking *two* inputs, namely x and a string r of random bits of length t, where t is the running time of $\mathcal{A}$ on x. Then the above properties translate to the following:

(i) if $x \in L$ then $\mathcal{A}(x, r)$ outputs "yes" for at least half of the strings r;
(ii) if $x \notin L$ then $\mathcal{A}$ outputs "no" for all strings r.

(a) Suppose now that we pick s *pairwise independent* uniform random strings $r_1, \ldots, r_s \in \{0, 1\}^t$, and output "yes" if at least one of the trials $\mathcal{A}(x, r_i)$ outputs "yes". Show that the error probability for this algorithm is $\frac{1}{s}$. [HINT: Let $Y = \sum_{i=1}^{s} Y_i$, where Y_i is the indicator r.v. for the event that $\mathcal{A}(x, r_i)$ outputs "yes". Show that $\mathrm{Var}[Y] \leq \frac{s}{4}$, and use Chebyshev's inequality.]

(b) Explain briefly why only $O(t)$ random bits are needed to implement the scheme in part (a). Note that the number of random bits needed is independent of δ.

(c) What is the running time of this scheme (as a function of t and δ), and how does it compare to the standard approach based on independent trials? Ignore the time taken to generate pairwise independent random strings.