CS–174 Combinatorics & Discrete Probability, Spring 2023

Homework 7 Solutions

Note: These solutions are not necessarily model answers. Rather, they are designed to be tutorial in nature, and sometimes contain a little more explanation than an ideal solution. Also, bear in mind that there may be more than one correct solution. The maximum total number of points available is 35.

1. (a) The random variables in the family are indexed by $x \in \mathbb{Z}_p$, so there are p of them. *1pt*

(b) As stated in the hint, our goal is to prove that $\Pr[f_{\boldsymbol{a}}(x) = y] = \frac{1}{p}$ for all $x, y \in \mathbb{Z}_p$, for $\boldsymbol{a}$ chosen u.a.r. This event is equivalent to $\sum_{i=0}^{d-1} a_i x^i = y$. Using the principle of deferred decisions, we can first choose the coefficients $a_1, \ldots, a_{d-1}$, at which point the event becomes $a_0 = y - \sum_{i=1}^{d-1} a_i x^i$. Thus there is a unique value of $a_0 \in \mathbb{Z}_p$ that makes the event true, so since a_0 is chosen u.a.r. from $\mathbb{Z}_p$, independently of the other a_i, the probability of the event is exactly $\frac{1}{p}$, as required. *4pts*

(c) Again following the hint, our goal is to prove 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 random choice of $\boldsymbol{a}$. But this event defines d points on the degree-$(d-1)$ polynomial $f_{\boldsymbol{a}}$, so the polynomial, and hence its coefficients $a_0, a_1, \ldots, a_{d-1}$, are uniquely determined. Since each such polynomial is picked with uniform probability from the set of p^d polynomials, the probability of this events is exactly $\frac{1}{p^d}$, as required. *4pts*

2. (a) Let C be any set of k vertices in K_n. The probability that C is monochromatic is equal to two times $\Pr[\text{all edges in } C \text{ are red}]$. But this probability in turn is equal to $2^{-\binom{k}{2}}$, since the edge colors are d-wise independent. Hence the probability that *any* C is monochromatic is (by a union bound) at most $\binom{n}{k} 2^{-\binom{k}{2}+1}$, which is less than 1 when $n \leq 2^{k/2}$, as we saw in class using a simple computation. *2pts*

(b) We need at least m d-wise independent random variables (one for each edge of K_n). Using the construction of part (a), this entails that $p \geq m$. *1pt*

(c) Using the parameters $m = \binom{n}{2}$ and $d = \binom{k}{2}$, we see that the size of the sample space is $p^d \leq n^{k^2}$. (Each point in the sample space is a choice of d coefficients from $\mathbb{Z}_p$.) Since this is polynomial in n for any fixed k, we can therefore deterministically cycle through every sample point $\boldsymbol{a}$: for each such point, we then use the construction of the previous question to generate m d-wise independent colors for the edges, and for each such 2-coloring we check each of the $\binom{n}{k} = O(n^k)$ cliques in K_n to see if it is monochromatic. We stop when we find a 2-coloring that is k-good (which must happen, since by part (a) some such coloring in our sample space exists and we are trying all of them). The resulting algorithm is deterministic and requires time $O(n^{k^2+k}) = O(n^{k^2})$. *4pts*

(d) We can allocate one group of $\binom{n}{k}$ processors to each of the $O(n^{k^2})$ sample points. We first use $p \leq 2m \leq n^2$ of these processors to evaluate, in parallel, the polynomial $f_{\boldsymbol{a}}(x)$ at m points x, for this particular sample point $\boldsymbol{a}$, thus generating the colors for the edges of K_n. Each of these evaluations of a degree-$(d-1)$ polynomial can be done in constant time (since $d = \binom{k}{2}$ is a constant[1]). Then, each of the $\binom{n}{k}$ processors checks one of the k-cliques of K_n to see if it is monochromatic: this takes constant time since k is constant. All processors finding a monochromatic clique report back to one single "lead" processor in the group: if none have reported, then the lead processor announces to the world that a k-good coloring has been found. The total (parallel) time required is $O(\log n)$ since the final reporting can be done in logarithmic time. *4pts*

[1] Technically we should charge $O(\log n)$ time for this since we are working with integers with $O(\log n)$ bits. But this does not affect the overall time complexity.

3. (a) The running time is $O(n)$. For each of the two sets S_1, S_2, we run n hashing operations and increment counters n times. Finally, we need to make n comparisons. *2pts*

(b) If S_1 and S_2 are identical, then clearly the ith counters for both tables will match for all i, regardless of the choice of hash function. *1pt*

(c) Suppose S_1 and S_2 are not identical, and furthermore S_1, S_2 are disjoint. Fix some $x \in S_1$, and note from our assumptions that $x \notin S_2$. For each $y \in S_2$, $y \neq x$, we have by the property of universal hash functions that $\Pr_h[h(x) = h(y)] \leq 1/cn$. Taking a union bound over all $y \in S_2$, we have $\Pr_h[\exists y \in S_2 : h(x) = h(y)] \leq 1/c$. Hence, with probability $1 - 1/c$ over h, we have that $h(y) \neq h(x)$ for all $y \in S_2$. In that case, the $h(x)$th counter for S_2 is 0, whereas that for S_1 is at least 1, so the algorithm outputs "no". *4pts*

4. (a) It is clear that the new algorithm still outputs "no" with probability 1 on input $x \notin L$. For input $x \in L$, let $Y = \sum_{i=1}^{t} Y_i$, where Y_i is the indicator r.v. for the event that $\mathcal{A}(x, r_i)$ outputs "yes". Then, $\mathrm{E}[Y_i] \leq 1/2$ and $\mathrm{Var}[Y_i] \leq 1/4$. Hence, $\mathrm{E}[Y] \leq s/2$ and (since the Y_i are *pairwise* independent) $\mathrm{Var}[Y] = \sum_{i=1}^{t} \mathrm{Var}[Y_i] \leq s/4$. The probability that the new algorithm outputs "no" is given by $\Pr[Y = 0]$. Applying Chebyshev's inequality, we have *4pts*

$$\Pr\big[Y = 0\big] \leq \Pr\big[|Y - \mathrm{E}[Y]| \geq s/2\big] \leq \frac{\mathrm{Var}[Y]}{(s/2)^2} = 1/s.$$

This yields the required bound on the error probability.

(b) To implement part (a), we need to generate $s = 1/\delta \leq 2^t$ pairwise independent uniform random strings in $\{0,1\}^t$. We can do this using the construction discussed in class and in [MU, Lemma 15.2]. Namely, let $2^t < q < 2^{t+1}$ be a prime and use the construction to generate q pairwise independent samples from $\mathbb{Z}_q$, using only two independent random integers in $\mathbb{Z}_q$. This requires only $O(t)$ random bits, as required. *2pts*

(c) We need to run $\mathcal{A}$ $s = 1/\delta$ times in the new scheme, versus $O(\log(\delta^{-1}))$ times in the standard approach. Hence the running times are $O(t/\delta)$ and $O(t \log \delta^{-1})$ respectively. Note, therefore, that the new scheme uses significantly fewer random bits, but at the expense of an increased running time. *2pts*