

CMSC 858T: Randomized Algorithms

Spring 2003

Ungraded Homework Assignment #3, handed out April 4, 2003

The suggested deadline by which to finish this assignment is April 17th; since this assignment is ungraded, you don't need to turn it in – just compare your solutions with the solutions I give.

Notation: The set $\{1, 2, \dots, k\}$ is denoted $[k]$.

0. Read Handouts 5 and 6.

1. Recall that we showed in class that for any graph with m edges, there is a cut with at least $m/2$ edges; derandomize this using the method of conditional probabilities. Look at your derandomization carefully, and see if the deterministic algorithm obtained is actually quite intuitive.

2. Recall Problem 4 of HW 1, where we construct (n, k) -universal sets. Suppose you are given a *constant* k ; give a deterministic polynomial-time algorithm to construct (n, k) -universal sets with the same number of rows (of the matrix) as guaranteed by the probabilistic argument.

3. For parts (a) and (b) of this problem, let X_1, X_2, \dots, X_ℓ be *independent* r.v.s, each taking values in $\{0, 1\}$. We will let $\vec{X} \doteq (X_1, X_2, \dots, X_\ell)$, and all events and r.v.s considered here are completely determined by the value of \vec{X} . Suppose \mathcal{E} is some event. A random variable $g = g(\vec{X})$ is said to be a *well-behaved* estimator for \mathcal{E} (w.r.t. \vec{X}) iff it satisfies the following properties (P1), (P2), (P3) and (P4), $\forall t \leq \ell$, $\forall T = \{i_1, i_2, \dots, i_t\} \subseteq [\ell]$, $\forall b_1, b_2, \dots, b_t \in \{0, 1\}$; for convenience, let \mathcal{B} denote " $\bigwedge_{s=1}^t (X_{i_s} = b_s)$ ".

(P1) $\mathbf{E}[g|\mathcal{B}]$ is efficiently computable;

(P2) $\Pr[\mathcal{E}|\mathcal{B}] \leq \mathbf{E}[g|\mathcal{B}]$;

(P3) if \mathcal{E} is increasing, then $\forall i_{t+1} \in ([\ell] - T)$, $\mathbf{E}[g|(X_{i_{t+1}} = 0) \wedge \mathcal{B}] \leq \mathbf{E}[g|(X_{i_{t+1}} = 1) \wedge \mathcal{B}]$; and

(P4) if \mathcal{E} is decreasing, then $\forall i_{t+1} \in ([\ell] - T)$, $\mathbf{E}[g|(X_{i_{t+1}} = 1) \wedge \mathcal{B}] \leq \mathbf{E}[g|(X_{i_{t+1}} = 0) \wedge \mathcal{B}]$.

Taking g to be the indicator variable for \mathcal{E} will satisfy (P2), (P3) and (P4), but not necessarily (P1). So the idea is that we want to approximate quantities such as $\Pr[\mathcal{E}|\mathcal{B}]$ “well” (in the sense of (P2), (P3) and (P4)) by an *efficiently computable* value ($\mathbf{E}[g|\mathcal{B}]$).

For any r.v. X and event \mathcal{A} , let $\mathbf{E}'[X]$ and $\mathbf{E}'[X|\mathcal{A}]$ respectively denote $\min\{\mathbf{E}[X], 1\}$ and $\min\{\mathbf{E}[X|\mathcal{A}], 1\}$. Suppose $\Pr[X_i = 1] = p_i$ for each i . Let $\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_k$ all be *increasing* events with respective well-behaved estimators h_1, h_2, \dots, h_k .

(a). Define, for all $i \in [k]$,

$$\begin{aligned}
 u_i &= \mathbf{E}[h_i | \bigwedge_{j=1}^t (X_j = b_j)]; & u'_i &= \min\{u_i, 1\}; \\
 v_i &= \mathbf{E}[h_i | ((X_{t+1} = 0) \wedge \bigwedge_{j=1}^t (X_j = b_j))]; & v'_i &= \min\{v_i, 1\}; \\
 w_i &= \mathbf{E}[h_i | ((X_{t+1} = 1) \wedge \bigwedge_{j=1}^t (X_j = b_j))]; & w'_i &= \min\{w_i, 1\}.
 \end{aligned}$$

Show that for all $i \in [k]$, $u'_i \geq (1 - p_{t+1}) \cdot v'_i + p_{t+1} \cdot w'_i$.

(b). Prove that for any non-negative integer $t \leq \ell - 1$ and any $\vec{b} = (b_1, b_2, \dots, b_t) \in \{0, 1\}^t$,

$$\prod_{i=1}^k (1 - \mathbf{E}'[h_i | \bigwedge_{j=1}^t (X_j = b_j)]) \leq (1 - p_{t+1}) \cdot \prod_{i=1}^k (1 - \mathbf{E}'[h_i | ((X_{t+1} = 0) \wedge \bigwedge_{j=1}^t (X_j = b_j))]) + p_{t+1} \cdot \prod_{i=1}^k (1 - \mathbf{E}'[h_i | ((X_{t+1} = 1) \wedge \bigwedge_{j=1}^t (X_j = b_j))]).$$

One approach is to use the result of part (a) along with an induction on k .

(c). Use part (b) to derandomize the approach using FKG that we developed for the edge-disjoint paths problem.

4(a). Suppose $X = \sum_{i=1}^t X_i$, where the random variables X_i lie in $[0, 1]$ and are *pairwise* independent. Let the mean of X be μ . Is it true that for any $a > 0$, $\Pr[|X - \mu| \geq a] \leq \mu/a^2$?

(b). Suppose the X_i are in fact d -wise independent for some $d \geq 4$. Suggest a possibly better approach to bound $\Pr[|X - \mu| \geq a]$ than the above.

5. Suppose we have a BPP algorithm A for a language L and an input instance x for which the following holds: A uses R perfectly random bits on input x , and is correct with probability at least $2/3$. (That is, if $x \in L$, A will say “Yes” with probability at least $2/3$; if $x \notin L$, then A will say “No” with probability at least $2/3$. Also, R is bounded by some fixed polynomial of $|x|$.) Suppose we want to boost this “ $2/3$ ” to $1 - 1/R$; recall that the standard way to do this boosting is to run A on x an odd number of times and then taking the majority outcome.

Show how to do this boosting in polynomial time, using $O(R \log R)$ random bits. Next, use problem 4 to show how to do this boosting in polynomial time, using only $O(R)$ random bits.

6. Recall the two-party problem of deciding Equality, in the communication complexity setting: Alice and Bob have n -bit strings x and y respectively, and want to determine if $x = y$ or not. Consider the following protocol, where t is a predetermined parameter. Alice chooses a random integer r from the range $[t]$, and sends the pair $(r, x \bmod r)$ to Bob; Bob replies “Yes” to Alice if $x \bmod r = y \bmod r$, and replies “No” otherwise. Note that the protocol communicates $O(\log t)$ bits. Suppose you desire the error probability to be at most p ; then, how large a t is sufficient?

7. We are given parameters n , $k \leq n$, and $\epsilon \in (0, 1)$. Design a randomized hashing strategy whose domain is $[n]$ and such that the hash function can be specified using only $O(\log k + \log(1/\epsilon) + \log \log n)$ random bits, such that the following holds: for any given subset S of $[n]$ such that $|S| = k$, the hash values of all the elements of S are different with probability at least $1 - \epsilon$. (We are not claiming that this *simultaneously* holds for all S ; instead, for every *given* S , the claim is true.)