CS 174, Fall, 05, Steiger

Instructions: Do all your work in the blue exam books. Please write your answers IN THE GIVEN ORDER, though you may solve problems in any order. There is no need to reduce answers to simplest terms. You may use one page of prepared notes, but all work must be your own. Show ALL your work. You will get little or no credit for an unexplained answer. The value of each question appears in parentheses. Use this as a guide in allocating your time. There are 80 points, and you have 80 minutes.

1. (10 pts) This question concerns Karger's basic min-cut algorithm (ALG 0): We start with a connected graph $G = (V,E)$ having n vertices and m _ $n - 1$ undirected edges (each of weight one). While our (multi)graph has more than 2 vertices we contract a randomly chosen edge (deleting any loops). At the end we output the remaining edges that join the final two vertices; they are a cut in the original G. In this problem G is a cycle of length $n-1$ plus a single edge attached to it (so G has n vertices and edges which we take as v_1v_2, $v_2v_3, \dots, v_{n-2}v_{n-1}, v_{n-1}v_1$ and v_1v_n).

(a) Draw G. Identify all min-cuts of G. What is the probability that the first edge selected by the algorithm is NOT in a min cut?

(b) What is the probability that ALG 0 will find a min-cut? Explain your answer (you should describe how G contracts to the final min-cut during a successful contraction sequence, showing the last two contraction steps). Can you think of a connected graph with n vertices on which the algorithm is certain to find a min cut?

2. (30 pts) X and Y are random variables on the same probability space. X has mean $E(X) = 2$, variance $V(X) = 9$, and $P(X > 10) = 0$. Y has mean $E(Y) = 3$, and $E(XY) = 6$. For each of the following statements, decide whether it is TRUE or FALSE ("TRUE" means that the statement must always be true for random variables satisfying the given conditions). If you say TRUE, give a convincing reason. If you say FALSE, give a counter-example.

(a) $P(Y = 3) < 1$

(b) $P(X \geq 1) \geq 1/10$

(c) $V(X + Y) = 9 + V(Y)$

(d) $P(X \geq 8) \leq 1/4$

(e) $P(X \geq 5) \leq 2/5$

(f) X and Y are independent.

3. (20 pts) This question deals with random permutations. The probability space is $S = \{\pi = (\pi_1 < \cdots < \pi_k)\}$ of permutations of $1, \dots, n$ under equally likely probability. Here $n = 2k + 1$ is odd.

(a) Let A be the event that the first k elements of π are increasing ($\pi_1 < \cdots < \pi_k$)? Find P(A) and explain how you did it. Next, explain in English how you would efficiently generate a permutation in A under equally likely probability. What is the running time of your algorithm?

(b) Let B be the event that the last k +1 elements of π are decreasing ($\pi_{k+1} > \cdots > \pi_{2k+1}$). Find P(B) and explain how you did it. Are A and B independent? Explain.

(c) Let C be the event that π has two cycles, one of length k, and one of length k + 1, the odd numbers in one cycle and the even numbers in the other. Find P(C) and explain how you did it.
(d) Let D be the event that even and odd values alternate (if π i is even then π i+1 is odd, and vice-versa, all i < n. Find the probability of D and explain how you did it.

4. (5 pts) n balls are placed in n bins at random, each outcome equally likely. Let A be the event that there are exactly three empty boxes, and NO box has more than two balls. Find the probability of A and explain how you did it.

5. (15 pts) Given a set S of n = 4k real inputs ai, i = 1 . . . , n, all distinct, the task is to return an element x in S that is in the top quarter; i.e., the returned value x must be bigger than at least 3k elements of S. An obvious algorithm is to take ANY 3k +1 elements of S and return their maximum. The cost is 3k (= .75n) comparisons. Carefully describe a probabilistic algorithm for this task which has running time o(n) as n ! 1 and which returns a correct answer with probability at least 1 − ε", where "ε > 0 is a given constant (the faster the algorithm, the better). Make it clear what the running time of your algorithm is (measured in terms of (i) the number of comparisons and (ii) the number of calls to UNIF). [(*) If you wish, and if you have extra time, comment on the possibility of a better deterministic algorithm - no penalty if you don’t]