Homework 10: Distributions, Expectation, Entropy, Sampling

Problem 1 [Easy]**: Expected Winnings**

Bob insists you play a card game with him. You need to draw a card at random from a deck of 52 cards (4 suits, 13 values per suit). If you draw a face card (J,Q, or K), Bob pays you \$3. If you do not draw a face card, you pay Bob \$1.

i. How much should you expect to win or lose?

ii. At least how many dollars should Bob pay you when you draw a face card to ensure you expect to gain at least a dollar from playing?

Let c be the payout for winning. We want the expected value to be 1 or more.

Problem 2: Expectation power game

Consider the following game of n matches: player A pays B at beginning of the match a match-fee of \$11. A coin is flipped, and the flips counted until a head appears; call this count including the head k. Then player B pays to A $\$2^k$ and the match ends (for example if a head occurred first at coinflip $k = 3$ then B pays \$8 to A). A game consists of n independent matches.

i. What is the expected payback from B to A at the end of each match?

ii. Which player would you rather be, A or B, assuming n is very large?

iii. If $n = 1$ match, what is the chance that A ends up with a positive balance after the match?

iv. If A starts with \$16, what is the probability A goes broke on the second round?

v. **(optional, no credit) ★★★.** Instead of \$11 fee per match, lets assume a more general, fixed G amount that A pays to B at beginning of each match. Approximate the number of matches n as a function of G, so that the chance A ends up with a positive balance after the game is about 50%.

Problem 3: Bucket Sort

Suppose you are given a list A of n integers. Further suppose it is known to be uniformly distributed in range $[0, MAX - 1]$ (i.e. $\forall i, 1 \leq i < n, \forall v, 0 \leq v < MAX, P(A[i] = v) = \frac{1}{MAX}$).

Now consider a sorting algorithm, Bucket Sort, that works as follows

- Organize the elements into $K+1$ equal-size buckets, B_i, such that the range of $B_0 = [0, \frac{MAX}{K} - 1]$, $B_1 = [\frac{MAX}{K}, 2 * \frac{MAX}{K} - 1]$,…, $B_K = (K-1) * \frac{MAX}{K}, K * \frac{MAX}{K} - 1$.
- Sort each bucket using insertion sort.
- Produce the sorted array by enumerating elements in each bucket in sorted order, in the order of the buckets.

i. Show how to do step A in one pass through the list. Your procedure should produce K lists in an array B such that $B[i]$ holds the list of elements in bucket i. What is the run time of this procedure?

ii. ★ Given the assumption of uniform distribution, how many elements are expected in each bucket after step A? What is the expected(average) time to sort a bucket? Hint: Consider an event X_{ij} which is 1 if the j^{th} element of the array is in bucket i and 0 otherwise. Then define $n_i = \sum_j X_{ij}$. You'll need $E[n_i^2]$

iii. Why is the output of Bucket Sort guaranteed to be sorted?

iv. As a function of K and n and MAX, what is the total run time of Bucket Sort under the uniform assumption?

v. How would you choose K to obtain linear time in n?

vi. If the data is not uniform, what is the worst case? What is the run time then?

vii. ★★ **(optional, no credit)** If the data is normally (Gaussian) distributed, with μ and σ known, can you still run Bucket Sort?

Problem 4 Bogo Sort Consider a new sorting algorithm, Bogo Sort. We start with a list of n elements. In Bogo Sort, we swap all the elements in the list at random and check if the list is sorted by examining all pairs, starting from the list's head. If it is sorted, we stop. Otherwise, we try again. Assume that we never see the same ordering twice.

i. Consider a list where the first k elements are sorted. At least how many comparisons must be made to determine the list is <u>not</u> in order?

ii. What is the probability that a permutation of n elements has the first k elements sorted?

iii. If a random variable X must be non-negative, we can use an alternate formula for expectation. Mainly,

$$E[X] = \sum_{k>0}^{\infty} P(X \geq k)$$

Given this formula, what is the expected number of comparisons for a given permutation of n elements? (Hint: $e^x \geq 1 + x + x/2! + x/3! + x/4! + ... + x/n!$)

iv. At worst, about how many comparisons will Bogo Sort make?

Problem 5: Entropy and Huffman codes

You're consulting with a telecommunications company that wants to compress some information using a frequency-based, per-symbol compression method (such as Huffman codes). Two proposed ways of initially turning the information into symbols will require different frequencies for each character, and will therefore take more or less bits.

i. Calculate the entropy for each of the character streams below. Which of the two streams will take fewer bits per character, on average?

1. Stream 1: Sends "A" 50% of the time, "B", "C","D" 12.5% of the time each, "E" and "F" 6.25% of the time each .
2. Stream 2: Sends "A" and "B" 25% of the time each, ' "C","D", "E", "F" 12.5% of the time each.

ii. Draw a Huffman Tree for each stream, and assign corresponding codes.

iii. ★★ **(optional. no credit)** Prove that if the distribution is power-2 based (such as the ones given) Huffman codes achieve the entropy rate of E[#bits per symbol]

iv. (Bayes) You later come back to this project and look at a random encoded character to figure out which method they adopted. It's a "E" Assuming the project was 75% likely to go with the method that seemed more efficient from the previous problem, and given this one piece of information, what is the probability that the company went with the lower-entropy method?

Problem 6 Trees equivalent definitions. A simple undirected graph G=(V,E) is given. Prove that each two of the following properties implies the third one. You will need three separate proofs:
(a) number of edges is $V-1$
(b) no cycles
(c) all vertices are connected

Problem 7 Cliques

Let $G=(V,E)$ be a graph with $|V|=n$ vertices. G is a complete graph of n vertices, if for every pair of vertices $u,v \in V$, $u \neq v$, $(u,v) \in E$. We call a complete graph of n vertices, K_n.

We may also consider a complete subgraph of G, called a clique. A k-clique is a clique of k vertices.

i. Given a value n, how many edges are in K_n?

ii. Given a value $k \leq |V|$, how many possible k-cliques are there in an undirected graph $G=(V,E)$?

iii. Given a graph G and k where $1 \leq k \leq |V|$, we can check to see if it has a clique by checking every possible combination of k vertices, and check if there exists an edge between all pairs. At worst, how many edges do you have to check exist or not?

iv. Suppose we wish to assign colors to each vertex in a graph G such that no two vertices that are connected by an edge share a color. If a graph has a k-clique, at least how many colors do you need to color that graph under our rules? Why?

v. Suppose G' is a directed graph that has been created by taking undirected graph G and imposing an arbitrary direction on each edge; thus if $\{a,b\}$ is an edge in G, (a,b) or (b,a) is an edge in G' (but not both). Prove by strong induction on the size of the clique k that there is always a path that hits all vertices in the clique, even in the directed version of the clique, for $k \geq 1$.

vi. ★ An undirected graph G with 6 vertices has edges placed at random. Prove that either G or its complement $\bar{G}$ has a clique of size three. ($\bar{G}$ is the graph with same vertices, but edges precisely the ones missing in G).
Is this still true if G has 5 vertices instead of 6?

Problem 8 ★★★ (optional, no credit) Occupancy without neighbors. On the Unfriendly Street there are n houses, all empty to be occupied as follows: customers, one at the time, choose one house empty/available at random; then that house AND the adjacent house(s) become unavailable.

This process goes on until no available houses are left: each is either occupied or neighboring an occupied one. What is the expected number of occupied houses?

Problem 9 ★★★ (optional, no credit) Dirac's Theorem

An undirected graph G with $n \geq 3$ vertices and minimum degree $deg(x) \geq n/2$ has a Hamilton cycle (a cycle that passes through all vertices exactly once)