CS 70 Discrete Mathematics and Probability Theory HW 08
Spring 2025 Rao

Due: Saturday, 3/22, 4:00 PM
Grace period until Saturday, 3/22, 6:00 PM

Sundry

Before you start writing your final homework submission, state briefly how you worked on it. Who else did you work with? List names and email addresses. (In case of homework party, you can just describe the group.)

1 Unprogrammable Programs

Note 12

Prove whether the programs described below can exist or not.

(a) A program $P(F,x,y)$ that returns true if the program F outputs y when given x as input (i.e. $F(x) = y$) and false otherwise.

(b) A program P that takes two programs F and G as arguments, and returns true if for all inputs x, F halts on x iff G halts on x (and returns false if this equivalence is not always true).

Hint: Use P to solve the halting problem, and consider defining two subroutines to pass in to P, where one of the subroutines always loops.

2 Kolmogorov Complexity

Note 12

Compressing a bit string x of length n can be interpreted as the task of creating a program of fewer than n bits that returns x. The Kolmogorov complexity of a string $K(x)$ is the length of an optimally-compressed copy of x; that is, $K(x)$ is the length of shortest program that returns x.

(a) Explain why the notion of the "smallest positive integer that cannot be defined in under 280 characters" is paradoxical.

(b) Prove that for any length n, there is at least one string of bits that cannot be compressed to less than n bits, assuming that no two strings can be compressed to the same value.

(c) Say you have a program K that outputs the Kolmogorov complexity of any input string. Under the assumption that you can use such a program K as a subroutine, design another program P that takes an integer n as input, and outputs the length-n binary string with the highest Kolmogorov complexity. If there is more than one string with the highest complexity, output the one that comes first lexicographically.

(d) Let's say you compile the program P you just wrote and get an m bit executable, for some $m \in \mathbb{N}$ (i.e. the program P can be represented in m bits). Prove that the program P (and consequently the program K) cannot exist.

(*Hint*: Consider what happens when P is given a very large input n that is much greater than m.)

3 Five Up

Note 13

Say you toss a coin five times, and record the outcomes. For the three questions below, you can assume that order matters in the outcome, and that the probability of heads is some p in $0 < p < 1$, but *not* that the coin is fair ($p = 0.5$).

(a) What is the size of the sample space, $|\Omega|$?

(b) How many elements of Ω have exactly three heads?

(c) How many elements of Ω have three or more heads?

For the next three questions, you can assume that the coin is fair (i.e. heads comes up with $p = 0.5$, and tails otherwise).

(d) What is the probability that you will observe the sequence HHHTT? What about HHHHT?

(e) What is the probability of observing at least one head?

(f) What is the probability you will observe more heads than tails?

Solution:

(a) The size of the sample space Ω is $2^5 = 32$. Elements in the sample space are of the type $\omega \in \mathbb{R}^5$.

(b) For three heads out of five coin flips, this subset of Ω has size $\binom{5}{3} = 10$.

(c) This subset is the union of the subsets of eleemnts with only 3, 4, and 5 heads each, i.e., $\binom{5}{3} + \binom{5}{4} + \binom{5}{5} = 10 + 5 + 1 = 16$.

(d) For both sequences, the probability of observing each is $\frac{1}{2^5} = \frac{1}{32}$.

(e) The complement of the subset of elements with at least one head is the subset of elements containing no heads. Therefore, the probability of observing a sequence with at least one head is $1 - \frac{1}{32} = \frac{31}{32}$, since there is only one sequence without any heads (TTTTT).

(f) For there to be more heads than tails, there must be at most two tails in the sequence. The number of elements with this property is $\binom{5}{0} + \binom{5}{1} + \binom{5}{2} = 10 + 5 + 1 = 16$. The probability of observing such a sequence is then $\frac{16}{32} = \frac{1}{2}$.

4 Aces

Note 13

Consider a standard 52-card deck of cards, which has 4 suits (hearts, diamonds, clubs, and spades) with 13 cards in each suit. Each suit has one ace. Hearts and diamonds are red, while clubs and spades are black.

(a) Find the probability of getting an ace or a red card, when drawing a single card.

(b) Find the probability of getting an ace or a spade, but not both, when drawing a single card.

(c) Find the probability of getting the ace of diamonds when drawing a 5 card hand.

(d) Find the probability of getting exactly 2 aces when drawing a 5 card hand.

(e) Find the probability of getting at least 1 ace when drawing a 5 card hand.

(f) Find the probability of getting at least 1 ace or at least 1 heart when drawing a 5 card hand.

5 Past Probabilified

Note 13

In this question we review some of the past CS70 topics, and look at them probabilistically.
For the following experiments, define an appropriate sample space Ω, and give the probability function $\mathbb{P}[\omega]$ for each $\omega \in \Omega$. Then compute the probabilities of the events E_1 and E_2.

(a) Fix a prime $p > 2$, and uniformly sample twice with replacement from $\{0, \ldots, p-1\}$ (assume we have two $\{0, \ldots, p-1\}$-sided fair dice and we roll them). Then multiply these two numbers with each other in $\pmod p$ space.

$$E_1 = \text{The resulting product is } 0.$$
$$E_2 = \text{The product is } (p-1)/2.$$

(b) Make a graph on n vertices by sampling uniformly at random from all possible edges, (assume for each edge we flip a coin and if it is head we include the edge in the graph and otherwise we exclude that edge from the graph).

$$E_1 = \text{The graph is complete.}$$
$$E_2 = \text{vertex } v_1 \text{ has degree } d.$$

(c) Create a random stable matching instance by having each person's preference list be a random permutation of the opposite entity's list (make the preference list for each individual job and each individual candidate a random permutation of the opposite entity's list). Finally, create a uniformly random pairing by matching jobs and candidates up uniformly at random (note that in this pairing, (1) a candidate cannot be matched with two different jobs, and a job cannot be matched with two different candidates (2) the pairing does not have to be stable).

$$E_1 = \text{All jobs have distinct favorite candidates.}$$
$$E_2 = \text{The resulting pairing is the candidate optimal stable pairing.}$$