CS 70 Discrete Mathematics and Probability Theory
Spring 2025 Rao

HW 08

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.

Solution:

(a) P cannot exist, for otherwise we could solve the halting problem:

```
def halt(F, x):
    def Q(x):
        F(x)
        return 0
    return P(Q, x, 0)
```

`Halt` defines a subroutine Q that first simulates F and then returns 0, that is $Q(x)$ returns 0 if $F(x)$ halts, and nothing otherwise. Knowing the output of $P(F,x,0)$ thus tells us whether $F(x)$ halts or not.

(b) We solve the halting problem once more:

```
def Halt(F, x):
    def Q(y):
        loop
    def R(y):
        if y == x:
            F(x)
        else:
            loop
    return not P(Q, R)
```

Q is a subroutine that loops forever on all inputs. R is a subroutine that loops forever on every input except x, and runs $F(x)$ on input x when handed x as an argument.

Knowing if Q and R halt on the same inputs boils down to knowing whether F halts on x (since that is the only case in which they could possibly differ). Thus, if $P(Q,R)$ returns "True", then we know they behave the same on all inputs and F must not halt on x, so we return `not P(Q,R)`.

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.)

Solution:

(a) Since there are only a finite number of characters then there are only a finite number of positive integers that can be defined in under 280 characters. Therefore there must be positive integers that are not definable in 280 characters and by the well-ordering principle there is a smallest member of that set. However the statement "the smallest positive integer not definable in under 280 characters" defines the smallest such an integer using only 69 characters (including spaces). Hence, we have a paradox (called the Berry Paradox).

(b) The number of strings of length n is 2^n. The number of strings shorter than length n is $\sum_{i=0}^{n-1} 2^i$. We know that sum is equal to $2^n - 1$ (remember how binary works). Therefore the cardinality of the set of strings shorter than n is smaller than the cardinality of strings of length n. Therefore there must be strings of length n that cannot be compressed to shorter strings.

(c) We write such a program as follows:

```
def P(n):
```

```
complex_string = "0" * n
for j in range(1, 2 ** n):
    # some fancy Python to convert j into binary
    bit_string = "{0:b}".format(j)
    # length should now be n characters
    bit_string = (n - len(bit_string)) * "0" + bit_string
    if K(bit_string) > K(complex_string):
        complex_string = bit_string
return complex_string
```

This program essentially just iterates through all possible bitstrings of length n, keeping track of the string with the highest complexity.

(d) We know that for every value of n there must be an incompressible string. Such an incompressible string would have a Kolmogorov complexity greater than or equal to its actual length. Therefore our program P must return an incompressible string. However, suppose we choose size n_k such that $n_k \gg m$, resulting in $n_k > m + log_2(n_k)$. Our program $P(n_k)$ will output a string x of length n_k that is not compressible meaning $K(x) \geq n_k$. However we have designed a program that outputs x using fewer bits than n_k (namely, using $m + log_2(n_k)$ bits). This is a contradiction. Therefore K cannot exist.

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) Since for each coin toss, we can have either heads or tails, we have 2^5 total possible outcomes.

(b) Since we know that we have exactly 3 heads, what distinguishes the outcomes is at which point these heads occurred. There are 5 possible places for the heads to occur, and we need to choose 3 of them, giving us the following result: $\binom{5}{3}$.

(c) We can use the same approach from part (b), but since we are asking for 3 or more, we need to consider the cases of exactly 4 heads, and exactly 5 heads as well. This gives us the result as: $\binom{5}{3} + \binom{5}{4} + \binom{5}{5} = 16$.

To see why the number is exactly half of the total number of outcomes, denote the set of outcomes that has 3 or more heads as A. If we flip over every coin in each outcome in set A, we get all the outcomes that have 2 or fewer heads. Denote the new set $\overline{A}$. Then we know that A and $\overline{A}$ have the same size and they together cover the whole sample space. Therefore, $|A| = |\overline{A}|$ and $|A| + |\overline{A}| = 2^5$, which gives $|A| = 2^5/2$.

(d) Since each coin toss is an independent event, the probability of each of the coin tosses is $\frac{1}{2}$ making the probability of this outcome $\frac{1}{2^5}$. This holds for both cases since both heads and tails have the same probability.

(e) We will use the complementary event, which is the event of getting no heads. The probability of getting no heads is the probability of getting all tails. This event has a probability of $\frac{1}{2^5}$ by a similar argument to the previous part. Since we are asking for the probability of getting at least one heads, our final result is: $1 - \frac{1}{2^5}$.

(f) To have more heads than tails is to claim that we flip at least 3 heads. Since each outcome in this probability space is equally likely, we can divide the number of outcomes where there are 3 or more heads by the total number of outcomes to give us: $\frac{\binom{5}{3}+\binom{5}{4}+\binom{5}{5}}{2^5} = \frac{1}{2}$

Alternatively, we see that for every sequence with more heads than tails we can create a corresponding sequence with more tails than heads by "flipping" the bits. For example, a sequence HTHHT which has more heads than tails corresponds to a flipped sequence THTTH which has more tails than heads. As a result, for every sequence with more heads there's a sequence with more tails. Thus, the probability of having a sequence with more heads is $1/2$.

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.

Solution:

(a) Inclusion-Exclusion Principle: $\frac{4}{52}+\frac{26}{52}-\frac{2}{52}=\frac{28}{52}=\frac{7}{13}$.

(b) Inclusion-Exclusion, but we exclude the intersection: $\frac{4}{52}+\frac{13}{52}-2\cdot\frac{1}{52}=\frac{15}{52}$.

(c) Ace of diamonds is fixed, but the other 4 cards in the hand can be any other card: $\frac{\binom{51}{4}}{\binom{52}{5}}$.

(d) Account for the number of ways to draw 2 aces and the number of ways to draw 3 non-aces: $\frac{\binom{4}{2}\cdot\binom{48}{3}}{\binom{52}{5}}$.

(e) Complement to getting no aces: $\mathbb{P}[\text{at least one ace}]=1-\mathbb{P}[\text{zero aces}]=1-\frac{\binom{48}{5}}{\binom{52}{5}}$.

(f) Complement to getting no aces and no hearts: $\mathbb{P}[\text{at least one ace OR at least one heart}]=1-\mathbb{P}[\text{zero aces AND zero hearts}]=1-\frac{\binom{36}{5}}{\binom{52}{5}}$. This is because $52-13-3=36$, where 13 is the number of hearts and 3 is the number of non-heart aces.

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 $(\bmod p)$ 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).

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

Solution:

(a) (i) This is essentially the same as throwing two $\{0,\dots,p-1\}$-sided dice, so one appropriate sample space is $\Omega = \{(i,j) \,:\, i,j \in \mathrm{GF}(p)\}$.

(ii) Since there are exactly p^2 such pairs, the probability of sampling each one is $\mathbb{P}[(i,j)] = 1/p^2$.

(iii) Now in order for the product $i \cdot j$ to be zero, at least one of them has to be zero. There are exactly $2p-1$ such pairs, and so $\mathbb{P}[E_1] = \frac{2p-1}{p^2}$.

(iv) For $i \cdot j$ to equal $(p-1)/2$ it doesn't matter what i is as long as $i \neq 0$ and $j \equiv i^{-1}(p-1)/2 \pmod p$. Thus $|E_2| = \left|\left\{(i,j) \,:\, j \equiv i^{-1}(p-1)/2\right\}\right| = p-1$, and whence $\mathbb{P}[E_2] = \frac{p-1}{p^2}$.
Alternative Solution for $\mathbb{P}[E_2]$: The previous reasoning showed that $(p-1)/2$ is in no way special, and the probability that $i \cdot j = (p-1)/2$ is the same as $\mathbb{P}[i \cdot j = k]$ for any $k \in \mathrm{GF}(p)$. But $1 = \sum_{k=0}^{p-1} \mathbb{P}[i \cdot j = k] = \mathbb{P}[i \cdot j = 0] + (p-1)\mathbb{P}[i \cdot j = (p-1)/2] = \frac{2p-1}{p^2} + (p-1)\mathbb{P}[i \cdot j = (p-1)/2]$, and so $\mathbb{P}[E_2] = \left(1 - \frac{2p-1}{p^2}\right)/(p-1) = \frac{p-1}{p^2}$ as desired.

(b) (i) Since any n-vertex graph can be sampled, Ω is the set of all graphs on n vertices.

(ii) As there are $N = 2^{\binom{n}{2}}$ such graphs, the probability of each indivdual one g is $\mathbb{P}[g] = 1/N$ (by the same reasoning that every sequence of fair coin flips is equally likely!).

(iii) There is only one complete graph on n vertices, and so $\mathbb{P}[E_1] = 1/N$.

(iv) For vertex v_1 to have degree d, exactly d of its $n-1$ possible adjacent edges must be present. There are $\binom{n-1}{d}$ choices for such edges, and for any fixed choice, there are $2^{\binom{n}{2}-(n-1)}$ graphs with this choice. So $\mathbb{P}[E_2] = \frac{\binom{n-1}{d}2^{\binom{n}{2}-(n-1)}}{2^{\binom{n}{2}}} = \binom{n-1}{d}\left(\frac{1}{2}\right)^{n-1}$.

(c) (i) Here there are two random things we need to keep track of: The random preference lists and the random pairing. A person i's preference list can be represented as a permutation σ_i of $\{1,\dots,n\}$, and the pairing itself is encoded in another permutation ρ of the same set (indicating that job i is paired with candidate $\rho(i)$). So $\Omega = \{(\sigma_1,\dots,\sigma_{2n},\rho) \,:\, \sigma_i,\rho \in S_n\}$, where S_n is the set of permutations of $\{1,\dots,n\}$.

(ii) $|\Omega| = (n!)^{2n+1}$, and so $\mathbb{P}[\mathscr{P}] = 1/|\Omega|$ for each $\mathscr{P} \in \Omega$.

(iii) For E_1, we observe that there are $n!$ possible configurations of all jobs having distinct favourite candidates, and that each job has $(n-1)!$ ways of ordering their non-favourite candidates, so $|E_1| = \underbrace{n!}_{\text{distinct favourites}} \cdot \underbrace{[(n-1)!]^n}_{\text{ordering of non-favourites}} \cdot \underbrace{(n!)^n}_{\text{candidate's preferences}} \cdot \underbrace{n!}_{\rho}$. Consequently, $\mathbb{P}[E_1] = n!\left(\frac{(n-1)!}{n!}\right)^n = \frac{n!}{n^n}$.

(iv) No matter what $\sigma_1, \ldots, \sigma_{2n}$ are, there is exactly one candidate-optimal pairing, and so $\mathbb{P}[E_2] = \frac{(n!)^{2n}}{(n!)^{2n+1}} = \frac{1}{n!}$.