CS 70 Discrete Mathematics and Probability Theory
Fall 2025 Sabin, Hug

HW 08

1 Post-Midterm Check-In Form

Please fill out this required form to let us know how the midterm went for you and what feedback you have for us: https://forms.gle/V7JgZnExmFqCAwjR7!

Solution: Filled out the form!

2 Count It!

Note 11

For each of the following collections, determine and briefly explain whether it is finite, countably infinite (like the natural numbers), or uncountably infinite (like the reals):

(a) The integers which divide 8.

(b) The integers which 8 divides.

(c) The functions from $\mathbb{N}$ to $\mathbb{N}$.

(d) The set of strings over the English alphabet. (Note that the strings may be arbitrarily long, but each string has finite length. Also the strings need not be real English words.)

(e) The set of finite-length strings drawn from a countably infinite alphabet, $\mathscr{C}$.

(f) The set of infinite-length strings over the English alphabet.

Solution:

(a) Finite. They are $\{-8,-4,-2,-1,1,2,4,8\}$.

(b) Countably infinite. We know that there exists a bijective function $f:\mathbb{N}\to\mathbb{Z}$. Then the function $g(n)=8f(n)$ is a bijective mapping from $\mathbb{N}$ to integers which 8 divides.

(c) Uncountably infinite. We use Cantor's Diagonalization Proof:

Let $\mathscr{F}$ be the set of all functions from $\mathbb{N}$ to $\mathbb{N}$. We can represent a function $f\in\mathscr{F}$ as an infinite sequence $(f(0),f(1),\cdots)$, where the i-th element is $f(i)$. Suppose towards a contradiction that there is a bijection from $\mathbb{N}$ to $\mathscr{F}$:

$$
\begin{aligned}
0 &\longleftrightarrow (f_0(0), f_0(1), f_0(2), f_0(3), \dots) \\
1 &\longleftrightarrow (f_1(0), f_1(1), f_1(2), f_1(3), \dots) \\
2 &\longleftrightarrow (f_2(0), f_2(1), f_2(2), f_2(3), \dots) \\
3 &\longleftrightarrow (f_3(0), f_3(1), f_3(2), f_3(3), \dots) \\
&\vdots
\end{aligned}
$$

Consider the function $g : \mathbb{N} \to \mathbb{N}$ where $g(i) = f_i(i) + 1$ for all $i \in \mathbb{N}$. We claim that the function g is not in our finite list of functions. Suppose for contradiction that it were, and that it was the n-th function $f_n(\cdot)$ in the list, i.e., $g(\cdot) = f_n(\cdot)$. However, $f_n(\cdot)$ and $g(\cdot)$ differ in the n-th argument, i.e. $f_n(n) \neq g(n)$, because by our construction $g(n) = f_n(n) + 1$. Contradiction!

(d) Countably infinite. The English language has a finite alphabet (52 characters if you count only lower-case and upper-case letters, or more if you count special symbols – either way, the alphabet is finite).

We will now enumerate the strings in such a way that each string appears exactly once in the list. We will use the same trick as used in Lecture note 10 to enumerate the elements of $\{0,1\}^*$ We get our bijection by setting $f(n)$ to be the n-th string in the list. List all strings of length 1 in lexicographic order, and then all strings of length 2 in lexicographic order, and then strings of length 3 in lexicographic order, and so forth. Since at each step, there are only finitely many strings of a particular length ℓ, any string of finite length appears in the list. It is also clear that each string appears exactly once in this list.

(e) Countably infinite. Let $\mathscr{C} = \{a_1, a_2, \dots\}$ denote the alphabet. (We are making use of the fact that the alphabet is countably infinite when we assume there is such an enumeration.) We will provide two solutions:

Alternative 1: We will enumerate all the strings similar to that in part (b), although the enumeration requires a little more finesse. Notice that if we tried to list all strings of length 1, we would be stuck forever, since the alphabet is infinite! On the other hand, if we try to restrict our alphabet and only print out strings containing the first character $a \in \mathscr{C}$, we would also have a similar problem: the list

$$a, aa, aaa, \dots$$

also does not end.

The idea is to restrict *both* the length of the string and the characters we are allowed to use:

1. List all strings containing only a_1 which are of length at most 1.
2. List all strings containing only characters in $\{a_1, a_2\}$ which are of length at most 2 and have not been listed before.

3. List all strings containing only characters in $\{a_1, a_2, a_3\}$ which are of length at most 3 and have not been listed before.

4. Proceed onwards.

At each step, we have restricted ourselves to a finite alphabet with a finite length, so each step is guaranteed to terminate. To show that the enumeration is complete, consider any string s of length ℓ; since the length is finite, it can contain at most ℓ distinct a_i from the alphabet. Let k denote the largest index of any a_i which appears in s. Then, s will be listed in step $\max(k, \ell)$, so it appears in the enumeration. Further, since we are listing only those strings that have not appeared before, each string appears exactly once in the listing.

Alternative 2: We will encode the strings into ternary strings. Recall that we used a similar trick in Lecture note 10 to show that the set of all polynomials with natural coefficients is countable. Suppose, for example, we have a string: $S = a_5a_2a_7a_4a_6$. Corresponding to each of the characters in this string, we can write its index as a binary string: $(101, 10, 111, 100, 110)$. Now, we can construct a ternary string where "2" is inserted as a separator between each binary string. Thus we map the string S to a ternary string: 101210211121002110. It is clear that this mapping is injective, since the original string S can be uniquely recovered from this ternary string. Thus we have an injective map to $\{0, 1, 2\}^*$. From note 11, we know that the set $\{0, 1, 2\}^*$ is countable, and hence the set of all strings with finite length over $\mathscr{C}$ is countable.

(f) Uncountably infinite. We can use a diagonalization argument. First, for a string s, define $s[i]$ as the i-th character in the string (where the first character is position 0), where $i \in \mathbb{N}$ because the strings are infinite. Now suppose for contradiction that we have an enumeration of strings s_i for all $i \in \mathbb{N}$: then define the string s' as $s'[i] = (\text{the next character in the alphabet after } s_i[i])$, where the character after z loops around back to a. Then s' differs at position i from s_i for all $i \in \mathbb{N}$, so it is not accounted for in the enumeration, which is a contradiction. Thus, the set is uncountable.

Alternative 1: The set of all infinite strings containing only as and bs is a subset of the set we're counting. We can show a bijection from this subset to the real interval $\mathbb{R}[0, 1]$, which proves the uncountability of the subset and therefore entire set as well: given a string in $\{a, b\}^*$, replace the as with 0s and bs with 1s and prepend $'0.'$ to the string, which produces a unique binary number in $\mathbb{R}[0, 1]$ corresponding to the string.

3 Fixed Points

Note 12

Consider the problem of determining if a program P has any fixed points. Given any program P, a fixed point is an input x such that $P(x)$ outputs x.

(a) Prove that the problem of determining whether a program has a fixed point is uncomputable.

(b) Consider the problem of outputting a fixed point of a program if it has one, and outputting "Null" otherwise. Prove that this problem is uncomputable.

(c) Consider the problem of outputting a fixed point of a program F if the fixed point exists *and* is a natural number, and outputting "Null" otherwise. If an input is a natural number, then it has no leading zero before its most significant bit.

Show that if this problem can be solved, then the problem in part (b) can be solved. What does this say about the computability of this problem? (You may assume that the set of all possible inputs to a program is countable, as is the case on your computer.)

Solution:

(a) We can prove this by reducing from the Halting Problem. Suppose we had some program `FixedPoint(F)` that solved the fixed-point problem. We can define `TestHalt(F, x)` as follows:

```
def TestHalt(F, x):
    def F'(y):
        F(x)
        return y
    return FixedPoint(F_prime)
```

If $F(x)$ halts, we have that $F'(y)$ will always just return y, so every input is a fixed point. On the other hand, if $F(x)$ does not halt, F' won't return anything for any input y, so there can't be any fixed points. Thus, our definition of `TestHalt` must always work, which is a contradiction; this tells us that `FixedPoint` cannot exist.

(b) If this problem is solvable then the problem in part (a) is solvable, as we can output "no" to whether F has a fixed point if the program in part (b) returns "Null" and "yes" otherwise.

(c) The intuition is that there is a bijection between the set of all inputs and $\mathbb{N}$ so we can reduce the problem in part (b) to this problem. In particular, since the set of all inputs is countably infinite, there exists a bijective function g that maps a natural number n to a unique input string $g(n)$. Also, we can extend the definition of g slightly so that $g(\text{"Null"}) = \text{"Null"}$ so that it is still a bijection.

Consider the following program:

```
def FindFixedPoint(F):
    def F'(n):
        if n not in ℕ:
            error
        x = g(n)
        if F(x) = x:
            return n
        else:
            return n+1
    return g(FindFixedPointInN(F'))
```

Since g is a bijection, g^{-1} exists, and if $g(n) = x$, then $n = g^{-1}(x)$. If there is some x such that $F(x) = x$, then $n = g^{-1}(x)$ will be a fixed point for F'. Thus, if F has a fixed point, `FindFixedPointInN`(F') will return some n where $g(n)$ is a fixed point for F.

On the other hand, if $F(x) \neq x$ for every input x, then $F'(n) = n+1 \neq n$ for every n, which means F' will also not have a fixed point. Thus, our program solves the problem in part (b), which is already shown to be uncomputable. This means that this problem is also uncomputable.

4 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 $1/2$.

5 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.

6 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 entities (make the preference list for each individual job and each individual candidate a random permutation of the opposite entities). 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).

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

Solution:

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

(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 $\{1,\ldots,n\}$, 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 $\Omega = \{(\sigma_1,\ldots,\sigma_{2n},\rho) \;:\; \sigma_i,\rho \in S_n\}$, where S_n is the set of permutations of $\{1,\ldots,n\}$.

(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 $\mathbb{P}[E_2] = \frac{(n!)^{2n}}{(n!)^{2n+1}} = \frac{1}{n!}$.