CS 70 Discrete Mathematics and Probability Theory
Fall 2024 Hug, Rao

HW 08

1 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 $\mathscr{C} = \{a_1, a_2, \ldots\}$ 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

$$a, aa, aaa, \ldots$$

also does not end.

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

(a) List all strings containing only a_1 which are of length at most 1.

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

(c) 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.

(d) 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: $S = a_5 a_2 a_7 a_4 a_6$. 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 Lecture note 10, 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.

2 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 F and G halt on the same set of inputs (or false otherwise).

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)`.

3 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 72 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 $n_k \gg m$. 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. This is a contradiction. Therefore K cannot exist.

4 Probability Warm-Up

Note 13

(a) Suppose that we have a bucket of 30 green balls and 70 orange balls. If we pick 15 balls uniformly out of the bucket, what is the probability of getting exactly k green balls (assuming

$0 \le k \le 15$) if the sampling is done **with** replacement, i.e. after we take a ball out the bucket we return the ball back to the bucket for the next round?

(b) Same as part (a), but the sampling is **without** replacement, i.e. after we take a ball out the bucket we **do not** return the ball back to the bucket.

(c) If we roll a regular, 6-sided die 5 times. What is the probability that at least one value is observed more than once?

Solution:

(a) Let A be the event of getting exactly k green balls. Then treating all balls as distinguishable, we have a total of 100^{15} possibilities to draw a sequence of 15 balls. In order for this sequence to have exactly k green balls, we need to first assign them one of $\binom{15}{k}$ possible locations within the sequence. Once done so, we have 30^k ways of actually choosing the green balls, and 70^{15-k} possibilities for choosing the orange balls. Thus in total we arrive at

$$\mathbb{P}[A] = \frac{\binom{15}{k} \cdot 30^k \cdot 70^{15-k}}{100^{15}} = \binom{15}{k}\left(\frac{3}{10}\right)^k\left(\frac{7}{10}\right)^{15-k}.$$

(b) Using a similar approach, there are a total of $100 \cdot 99 \cdots 86 = \frac{100!}{(100-15)!} = \frac{100!}{85!}$ ways to grab 15 balls.

We still want k green balls and $15-k$ orange balls, which we can select in

$$\begin{aligned}&\binom{15}{k} \cdot (30 \cdot 29 \cdots (30-(k-1))) \cdot (70 \cdot 69 \cdots (70-(15-k-1))) \\ &= \binom{15}{k}\frac{30!}{(30-k)!} \cdot \frac{70!}{(70-(15-k))!}\end{aligned}$$

ways. Here, we have $\binom{15}{k}$ possible locations for the k green balls, and we use permutations rather than combinations to account for the fact that we are picking balls without replacement.

Since our sample space is uniform, the probability is thus

$$\mathbb{P}[A] = \frac{\binom{15}{k}\frac{30!}{(30-k)!} \cdot \frac{70!}{(70-(15-k))!}}{\frac{100!}{(100-15)!}} = \frac{\binom{15}{k}\frac{30!}{(30-k)!} \cdot \frac{70!}{(55+k)!}}{\frac{100!}{85!}}.$$

Alternative Solution: Instead of considering a probability space where each ball is distinct, we can also consider a probability space where each ball is indistinguishable, and order does not matter. Intuitively, this is equivalent to the previous solution, since the numerator and denominator are accounting for order in the exact same way; considering the same situation without order and with indistinguishable balls is equivalent to dividing both the numerator and denominator by 15! for the number of arrangements of the 15 balls we select.

With this in mind, we note that the size of the sample space is now $\binom{100}{15}$, since we are choosing 15 balls out of a total of 100. To find $|A|$, we need to be able to find out how many ways we can choose k green balls and $15-k$ orange balls. This means that we have $|A| = \binom{30}{k}\binom{70}{15-k}$, since we must select k green balls out of 30 total, and $15-k$ orange balls out of 70 total.

Putting this together, we have

$$\mathbb{P}[A] = \frac{\binom{30}{k}\binom{70}{15-k}}{\binom{100}{15}}.$$

(c) Let B be the event that at least one value is observed more than once. We see that $\mathbb{P}[B] = 1 - \mathbb{P}[\overline{B}]$. So we need to find out the probability that the values of the 5 rolls are distinct. We see that $\mathbb{P}[\overline{B}]$ simply the number of ways to choose 5 numbers (order matters) divided by the sample space (which is 6^5). So

$$\mathbb{P}[\overline{B}] = \frac{6!}{6^5} = \frac{5!}{6^4}.$$

And

$$\mathbb{P}[B] = 1 - \frac{5!}{6^4}.$$

5 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?

For the final three questions, you can instead assume the coin is biased so that it comes up heads with probability $p = \frac{2}{3}$.

(g) What is the probability of observing the outcome HHHTT? What about HHHHT?

(h) What about the probability of at least one head?

(i) 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 $1/2$.

(g) By using the same idea of independence we get for HHHTT:

$$\frac{2}{3}\times\frac{2}{3}\times\frac{2}{3}\times\frac{1}{3}\times\frac{1}{3}=\frac{2^3}{3^5}$$

For HHHHT, we get:

$$\frac{2}{3}\times\frac{2}{3}\times\frac{2}{3}\times\frac{2}{3}\times\frac{1}{3}=\frac{2^4}{3^5}$$

(h) Similar to the unbiased case, we will first find the probability of the complement event, which is having no heads. The probability of this is $\frac{1}{3^5}$, which makes our final result $1-\frac{1}{3^5}$.

(i) In this case, since we are working in a nonuniform probability space (getting 4 heads and 3 heads don't have the same probability), we need to separately consider the events with different numbers of heads to find our result. Thus for each $3 \le i \le 5$, we need to count the total number of outcomes with exactly i heads and multiply it by the probability of achieving any of those outcomes. This yields:

$$\begin{aligned}\mathbb{P}[\ge 3 \text{ Heads}] &= \binom{5}{3}\left(\frac{2}{3}\right)^3\left(\frac{1}{3}\right)^2 + \binom{5}{4}\left(\frac{2}{3}\right)^4\left(\frac{1}{3}\right)^1 + \binom{5}{5}\left(\frac{2}{3}\right)^5\left(\frac{1}{3}\right)^0 \\ &= \binom{5}{3}\left(\frac{2}{3}\right)^3\left(\frac{1}{3}\right)^2 + \binom{5}{4}\left(\frac{2}{3}\right)^4\left(\frac{1}{3}\right) + \binom{5}{5}\left(\frac{2}{3}\right)^5.\end{aligned}$$