CS 70 Discrete Mathematics and Probability Theory
Fall 2022 Satish Rao and Babak Ayazifar

HW 08

Due: Saturday, 10/22, 4:00 PM
Grace period until Saturday, 10/22, 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.) I did not work with anybody to complete this homework.

1 Count It!

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.

Solution: The numbers are $\{1,2,4,8\}$ which is obviously finite.

(b) The integers which 8 divides.

Solution: Call S the set of numbers which 8 divides. Then, we can take $\frac{S}{8}$ (i.e. dividing every element in the set by 8), which gives us the natural numbers. Therefore, this set is countably infinite, since the naturals are countably infinite.

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

Solution: This set is uncountably infinite, since we can imagine each function as outputting an infinite string of numbers based upon its input, for instance $f_n(1), f_n(2), f_n(3)$ and so on. Now, suppose we enumerate every natural number with a function. We can generate a new function g, where for every input k, $g(k) = f_k(k) + 1$. Therefore g is different from every function f_n in at least one spot, so it is not in our original enumeration. Therefore, this set is uncountably infinite.

(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.)

Solution: This set is finite, since having a finite length means that there are a finite number of letters and a finite length, so eventually we will run out of possible strings.

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

Solution: Just like the previous part, becuase the strings have finite length, then this set of strings is also countably infinite.

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

Solution: This set is uncountably infinite. We can form a diagonalization construction as follows: Suppose we have enumerated all infinite length strings over the English alphabet. Now, construct a new string which takes letters along the diagonal, where every letter is changed to the letter which precedes it (i.e. A $\rightarrow$ B, B $\rightarrow$ C, etc., and Z $\rightarrow$ A). Therefore, we've created a new string which is different from every other string in at least one location! Therefore, this set is uncountably infinite.

Another way to see this is the fact that the real numbers also have a finite alphabet (i.e. 0-9), but is uncountable due to diagonalization. Therefore the set of infinite-length strings over a finite alphabet is always uncountable.

2 Countability Proof Practice

(a) A disk is a 2D region of the form $\{(x,y) \in \mathbb{R}^2 : (x-x_0)^2 + (y-y_0)^2 \leq r^2\}$, for some $x_0, y_0, r \in \mathbb{R}$, $r > 0$. Say you have a set of disks in $\mathbb{R}^2$ such that none of the disks overlap. Is this set always countable, or potentially uncountable?
(*Hint*: Attempt to relate it to a set that we know is countable, such as $\mathbb{Q} \times \mathbb{Q}$.)

Solution: This set is countable, because every pair of rational numbers $(a,b) \in \mathbb{Q}^2$ exists in only one disk, since the disks do not overlap. Therefore, each rational pair (a,b) is uniquely within a single disk. Since $\mathbb{Q}^2$ is countable, then this set of disks is also countable.

(b) A circle is a subset of the plane of the form $\{(x,y) \in \mathbb{R}^2 : (x-x_0)^2 + (y-y_0)^2 = r^2\}$ for some $x_0, y_0, r \in \mathbb{R}$, $r > 0$. Now say you have a set of circles in $\mathbb{R}^2$ such that none of the circles overlap. Is this set always countable, or potentially uncountable?
(*Hint*: The difference between a circle and a disk is that a disk contains all of the points in its interior, whereas a circle does not.)

Solution: Consider the set of concentric circles centered at $(0,0)$, that have radius r. Since r can be any real number (a circle can have radius equal to any real number), then for every real number we can create a non-overlapping circle (since they're all centered at $(0,0)$), and so we've created a bijection between every circle and the real numbers. Since the real numbers are uncountable, then this set is also potentially uncountable.

(c) Is the set containing all increasing functions $f : \mathbb{N} \to \mathbb{N}$ (i.e., if $x \geq y$, then $f(x) \geq f(y)$) countable or uncountable? Prove your answer.

Solution: This set is uncountable.

(d) Is the set containing all decreasing functions $f : \mathbb{N} \to \mathbb{N}$ (i.e., if $x \geq y$, then $f(x) \leq f(y)$) countable or uncountable? Prove your answer.

Solution:

3 Unprogrammable Programs

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.

Solution: Define a test program `test(x, y)` which evaluates $F(x)$ and checks whether it outputs y on the input x. In other words, it evaluates $F(x)$, then if $F(x) = y$ then it returns true, or false otherwise.

Then, we can define P to be `Halt(x, y)` which returns `test(x, y)`. Then, by this construction, `Halt(x, y)` knows whether $F(x) = y$ or not, which is impossible since the halting problem is undecideable. Therefore, P cannot exist.

(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).

Solution:

4 The Complexity Hierarchy

The complexity hierarchy is a monument to our collective understanding of computation and its limitations. In fact, you may already be familiar with the classes P and NP from CS61B. In this problem, we will focus on decision problems like the Halting Problem, where the output is "Yes" (True) or "No" (False), and explore the classes RE, coRE, and R.

(a) A problem is recursively enumerable (RE) if there exists a program P that can print out all the inputs for which the answer is "Yes", and no inputs for which the answer is "No". The program P can print out a given input multiple times, so long as every input gets printed eventually. The program P can run forever, so long as every input which should be printed is at a finite index in the printed output.

Prove that the Halting Problem belongs in RE. Namely, prove that it is possible to write a program P which:

- runs forever over all possible programs M and inputs x, and prints out strings to the console,
- for every (M,x), if $M(x)$ halts, then P eventually prints out (M,x),
- for every (M,x), if $M(x)$ does NOT halt, then P never prints out (M,x).

In this context, P is called an *enumerator*. (Hint: Consider the tail of a dove.)

Solution:

(b) An equivalent definition of RE is as follows: A problem belongs in RE if there exists a program P' that will output "Yes" when given an input x for which the answer is "Yes". If the answer is "No", then $P'(x)$ may output "No" or loop forever. As an optional exercise, you should be able to convince yourself that this is indeed an equivalent definition.

Prove that the Halting Problem belongs in RE using this equivalent definition. Namely, prove that it is possible to write a program P' which:

- takes as input a program M and input x.
- if M halts on input x, then P' should print "Yes".
- if M does not halt on input x, then P' may output "No" or loop forever.

In this context, P' is called a *recognizer*.

(c) As you might suspect, a problem is co-recursively enumerable (coRE) if its complement is in RE. The complement of a decision problem A is another problem A' where $A'(x)$ is "Yes" iff $A(x)$ is "No", and $A'(x)$ is "No" iff $A(x)$ is "Yes". State the complement of the Halting Problem.

(d) Finally, a problem belongs in the class R if it is computable, meaning there exists a program P that answers "Yes" when the answer is "Yes", and answers "No" when the answer is "No". By definition then, the problem is a computable function if it is computable.

We know that the TestHalt is not computable, and that the Halting Problem belongs in RE. Prove by contradiction that the Halting Problem cannot belong in coRE.

5 Probability Warm-Up

(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 $0 \leq k \leq 15$) 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?

Solution: The probability of drawing a green ball is $\frac{3}{10}$ and the probability for an orange ball is $\frac{7}{10}$. Therefore, the probability of drawing k green balls is

$$\left(\frac{3}{10}\right)^k \left(\frac{7}{10}\right)^{15-k}$$

(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. **Solution:** Note that each ball picked up now reduces the number of available balls. Therefore, the probaiblity is

$$\mathbb{P} = \prod_{n=0}^{k-1} \left(\frac{30-k}{100-k}\right)\left(\frac{70-(15-k)}{100-k}\right)$$

(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: There are 6^5 combinations of rolling a 6-sided die 5 times, this is our sample space. If we want more than one value to be observed, we choose a specific value (for the sake of argument, say we choose 1), and ask how many ways are there to choose k of those results to be 1 - this gives us $\binom{6}{k}$. Then, for all the other rolls, there are 5^{5-k} ways of choosing which numbers are rolled. Therefore,

$$\mathbb{P} = \frac{\sum_{k=2}^{5} \binom{6}{k} \cdot 5^{5-k}}{6^5}$$

6 Past Probabilified

In this question we review some of the past CS70 topics, and look at them probabilistically. For the following experiments,

i. Define an appropriate sample space Ω.

ii. Give the probability function $\mathbb{P}[\omega]$.

iii. Compute $\mathbb{P}[E_1]$.

iv. Compute $\mathbb{P}[E_2]$.

(a) Fix a prime $q > 2$, and uniformly sample twice with replacement from $\{0, \ldots, q-1\}$ (assume we have two $\{0, \ldots, q-1\}$-sided fair dice and we roll them). Then multiply these two numbers with each other in $(\bmod q)$ space.
$E_1 =$ The resulting product is 0.
$E_2 =$ The product is $(q-1)/2$.

Solution:

i. The sample space is the set of pairs $(a,b)|a,b \in \{0,1,\ldots,q-1\}$. Since a and b can be chosen independently

(b) Make a graph on v 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).
$E_1 =$ The graph is complete.
$E_2 =$ vertex v_1 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) The sample space is the total number of ways of listing job preferences, candidate preferences, and an arbitrary matching.

(b) For the job preferences, there are n candidates, and each candidate can have $n!$ ways of arranging a preference list. The same argument follows from the candidates, since they are independent of each other. Further, there are $n!$ ways of matching a job to a candidate, so therefore we have

$$|\Omega| = (n!)^n \cdot (n!)^n \cdot n! = (n!)^{2n+1}$$

(c)