CS 70 Fall 2016 — Discrete Mathematics and Probability Theory by Alvin Wan, Vivek Raghuram — MT2 Practice

This worksheet contains problems for every section covered on your midterm 2. Vivek and Alvin wrote these problems from scratch, keeping in mind what has and has not been covered this Fall 2016 semester.

- Modular Arithmetic
- Bijections and RSA
- Polynomials
- Error Correcting Codes
- Infinity and Uncountability
- Self-Reference and Uncomputability
- Counting
- Introduction to Discrete Probability
- Conditional Probability

Note that problems here were designed to challenge you and to exercise your knowledge of these topics. In some cases, several topics were drawn upon to create the question. We recommend selectively working on topics where you feel weakest.

The concentration of questions in this review do not correspond to the concentration of questions on the exam.

1. **Piazza**

Between the CS70 and CS170 piazzas, if there are exactly 8 unresolved questions total, Sinho will resolve all of them (This means there can never be more than 8 unresolved questions). Let us pretend that Sinho can resolve questions instantaneously and he does not tend to Piazza if there are fewer than 8 unresolved questions. Sinho tells you that there are a total of 3 unresolved questions.

(a) Is it possible for CS70 to have as many unresolved posts as CS170?

(b) Is it possible for CS170 to have twice as many unresolved posts as CS70?

Solution:

Let x be the number of unresolved posts on CS70 Piazza and y be the number of unresolved posts on CS170 Piazza. We know that $x+y$ is always mod 8. If there are 3 unresolved posts, then

$$x+y=3 \pmod 8$$

(a) No. If CS70 has as many unresolved posts as CS170, then $x = y$. We have that

$$2y = 3 \pmod 8$$

Since 2 is not co-prime with 8, the multiplicative inverse of y does not exist. Thus, there may be multiple solutions or none. We can reduce the equation to $y = \frac{3}{2} \pmod 4$, which we note can never be true, since fractions do not exist in the (mod 4) universe. Thus, if Sinho observes 3 total posts, CS70 cannot have the same number of unresolved posts as CS170.

(b) Yes. If CS170 has twice times as many, then $y = 2x$. We have that

$$3x = 3 \pmod 8$$

Since 3 is co-prime to 8, we have that there is a unique multiplicative inverse, gaurantee-ing us a unique solution. However, we could also just note by inspection that $x = 1$ is a solution. Thus, it is possible for CS170 to have twice as many unresolved posts as CS70.

2. **Random Uniqueness**

Consider the following scenarios, where we apply probability to polynomials. We generate a new polynomial Q in GF(7), by randomly picking 6 numbers in (mod 7), for a polynomial of the form.

$$a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$$

The first number we pick is a_5, the second number is a_4 etc.

(a) What is the probability that the polynomial has degree at least 4?

(b) What is probability we have a unique polynomial of degree less than 4?

(c) Now, consider only degree 5 polynomials that were randomly generated using the scheme described above. What is the probability that the sum of its coefficients is equal to 6?

Solution:

(a) We first consider the number of total assignments. There are 7^6. We now count the number of valid assignments. We note that a valid degree 4 polynomial has $a_5 = 0$ and $a_4 \neq 0$. There are a total of $(1)(6)(7^4)$ ways. Next, we note that a valid degree 5 polynomial has $a_5 \neq 0$. Thus, there are a total of $6(7^5)$. In sum, we now have

$$\frac{6(7^4) + 6(7^5)}{7^6}$$

(b) This is 1 minus the previous part. So, we have the following.

$$1 - \frac{6(7^4) + 6(7^5)}{7^6}$$

(c) We note that there are $6(7^5)$ different coefficients; there are only 6 options for the first coefficient since it cannot be 0. We now compute all the possible ways to achieve a sum of 6. This is simply stars and bars, where the first bin must have at least one ball (a_5 cannot be zero). We distribute the sum 5 among 6 bins: $\binom{10}{5}$. Thus, our probability is

$$\frac{\binom{10}{5}}{6(7^5)}$$

3. **Maybe Lossy Maybe Not**

Let us say that Alice would like to send a message to Bob, over some channel. Alice has a message of length 4 and sends 5 packets.

(a) Packets are dropped with probability p. What is probability that Bob can successfully reconstruct Alice's message?

(b) Again, packets can be dropped with probability p. The channel may additionally corrupt 1 packet. Alice realizes this and sends 3 additional packets. What is the probability that Bob receives enough packets to successfully reconstruct Alice's message?

(c) Again, packets can be dropped with probability p. This time, packets may be corrupted with probability q. Consider the original scenario where Alice sends 5 packets for a message of length 4. What is probability that Bob can successfully reconstruct Alice's message?

Solution:

(a) Alice's message requires a polynomial of degree 3, which can be uniquely identified by 4 points. Thus, at least 4 points need to make it across the channel. The probability that Bob can recover the message is thus the probability that none of the packets are lost. Since the packets are lost with probability with probability p, we have the probability of losing 1 packet is

$$\binom{5}{1}(1-p)^4 p$$

The probability of losing 0 packets is $(1-p)^5$. Thus, the probability of losing 0 or 1 packets is

$$\binom{5}{1}(1-p)^4 p + (1-p)^5$$

This is the probability that Bob receives 4 packets, meaning he can successfully reconstruct the 3-degree polynomial.

(b) Bob needs $n+2k=6$ packets to guarantee successful reconstruction of Alice's message. There are a total of 8 packets sent, so this guarantee occurs only if 0 packets, 1 packet or 2 packets are lost. The probability of 0 packets lost is

$$(1-p)^6$$

The probability of one packet lost is

$$\binom{8}{1}p(1-p)^5$$

The probability of two packets lost is

$$\binom{8}{2}p^2(1-p)^4$$

Thus, the probability of success is

$$(1-p)^8+\binom{8}{1}p(1-p)^7+\binom{8}{2}p^2(1-p)^6$$

(c) Again, Bob can reconstruct the message if and only if none of the packets are corrupted. We use the same idea as in part (a). The probability that none of the packets are corrupted is $(1-q)^5$. We know that *on top of* being uncorrupted, we can only at lose at most 1 packet. Thus, we can either lose one packet, which has probability

$$\binom{5}{1}p(1-p)^4$$

Or, we can lose no packets, which has probability $(1-p)^5$.

As a result, we have the following.

$$(1-q)^5[\binom{5}{1}p(1-p)^4+(1-p)^5]$$

4. **Hilbert's Paradox of the Grand Hotel**

Consider a magical hotel with a countably infinite number of rooms numbered according to the natural numbers where all the rooms are currently occupied. Assume guests don't mind being moved out of their current room as long as they can get to their new room in a finite amount of time.

(a) Suppose one new guest arrived in their car, how would you shuffle guests around to accommodate them? What if k guests arrived, where k is a constant positive integer?

(b) Suppose a countably infinite number of guests arrived in an infinite length bus with seat numbers according to the natural numbers, how would you accommodate them?

(c) Suppose a countably infinite number of infinite length buses arrive carrying countably infinite guests each, how would you accommodate them? (Hint: There are infinitely many prime numbers)

Solution:

(a) Shift all guests into the room number k greater than their current room number. So for a guest in room i move them to room $i+k$. Then place the k new guests in the k first rooms in the hotel which will now be unoccupied.

(b) Place all existing guests in the room $2i$ where i is their current room number. Place all the new guests in the room $2j+1$ where j is their seat number on the bus.

(c) Place all existing guests in the room 2^i where i is their current room number. Assign the $(k+2)$th prime, p_{k+2}, to the kth bus (e.g. the 0th bus will be assigned the 2nd prime, 3). We then place each new guest in the room p_{k+2}^{j+1} where j is the seat number of the new guest in their bus.

This works because any power of a prime p will not have any other prime factors than p.

Yes, there will be plenty of empty rooms, but that's okay because every guest will still have somewhere to stay.

5. **Impossible Programs**
Show that none of the following programs can exist.

(a) Consider a program P that takes in any program F, input x and output y and returns true if $F(x)$ outputs y and returns false otherwise.

(b) Consider a program P that takes in any program F and returns true if $F(F)$ halts and returns false if it doesn't halt.

(c) Consider a program P that takes in any programs F and G and returns true if F and G halt on all the same inputs and returns false otherwise.

Solution:

(a) If P exists we can solve the halting problem. We show this by constructing machine $HALT(F,x)$ where F is a program and x is the input. We will use P as a subroutine to derive a contradiction.

```
def HALT(F, x):
  y = 0 # arbitrarily chosen
  def F_prime(x):
    F(x)
    return y
  return P(F_prime, x, y)
```

We modify F to create F that runs $F(x)$ and if $F(x)$ halts it outputs an arbitrarily chosen output y. We then call $P(f,x,y)$ and if it returns true then $F(x)$ halts and if it returns false then $F(x)$ must not halt. Therefore we have solved the halting problem. This is a contradiction because halting problem is uncomputable. Therefore the program P cannot exist.

(b) If P exists we can solve the halting problem. We show this by constructing machine $HALT(F,x)$ where F is a program and x is the input. We will use P as a subroutine to derive a contradiction.

```
def HALT(F, x):
  def F_prime(ignore):
    return F(x)
  return P(F_prime)
```

We construct function F which ignores its input and simply runs $F(x)$. We then call $P(F)$. If $P(F')$ returns true then $F(x)$ must have halted otherwise $P(F')$ will have returned false. Therefore we have solved the halting problem. This is a contradiction because halting problem is uncomputable. Therefore the program P cannot exist.

(c) If P exists we can solve the halting problem. We show this by constructing machine $HALT(F,x)$ where F is a program and x is the input. We will use P as a subroutine to derive a contradiction.

```
def HALT(F, x):
  def F_prime(y):
    F(x)
    while x != y:
      pass
    return

  def G(y):
    while x != y:
      pass
    return
  return P(F_prime, G)
```

We construct functions G and F'. Both functions loop forever unless the input is x. Additionally, F' runs $F(x)$ and so only halts if $F(x)$ halts. We then call $P(F',G)$ and if the answer is true then the the F halts on x otherwise it does not halt on x. Therefore we have solved the halting problem. This is a contradiction because halting problem is uncomputable. Therefore the program P cannot exist.

6. **Kolmogorov Complexity**

Compression of a bit string x of length n involves creating a program shorter than n bits that returns x. The Kolmogorov complexity of a string $K(x)$ is the length of shortest program that returns x (i.e. the length of a maximally compressed version of x).

(a) Explain why "the smallest positive integer not definable in under 100 characters" is paradoxical.

(b) Prove that for any length n, there must be at least one bit string that cannot be compressed.

(c) Imagine you had the program K, which outputs the Kolmogorov complexity of string. Design a program P that when given integer n outputs the bit string of length n with the highest Kolmogorov complexity. If there are multiple strings with the highest complexity, output the lexicographically first (i.e. the one that would come first in a dictionary).

(d) Suppose the program P you just wrote can be written in m bits. Show that P and by extension, K, cannot exist, for a sufficiently large input n.

(e) Consider the program I, which can be written in m bits, that when given any input string x returns true if x is incompressible and returns false otherwise. Show that such a program cannot exist.

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 100 characters. Therefore there must be positive integers that are not definable in 100 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 100 characters" defines the smallest such an integer using only 67 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
```

(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 $n_k >> m$. 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.

(e) We prove K does not exist by writing a program S that will output a string x where $|S| < K(x)$. This would be a contradiction and so we must conclude K does not exist.

```
def S():
  i = 1
  while i > 0:
    for j in range(1, 2**i):
      bit_string = "0:b".format(j)
      bit_string = (i - len(bit_string)) * "0" + bit_string
      if I(bit_string) and len(bit_string) > m + c
        return bit_string
    i += 1
```

The program S must take a finite number of bits. We select a constant c such that it is larger than the length of S. Therefore the total program including S and K must take less than $m+c$ bits. S goes through every binary strings and returns the first incompressible one that is longer than $m+c$ bits. Notice the contradiction? We have found a string x that I says cannot be compressed. However, we have a program of length less than $|x|$ that outputs x, satisfying our definition of compression. Therefore I cannot exist.

7. **Binomial Theorem**

The binomial theorem states the following:

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k$$

Prove this theorem using a combinatorial proof.

Solution: Imagine you are throwing n numbered balls into bins. There are x red bins and y blue bins

LHS: This is the number of ways of throwing n distinct balls into the bins.

RHS: Consider each term $\binom{n}{k}x^{n-k}y^k$. This is the number of ways of throwing the balls with k landing in the blue bins and $n-k$ landing in the red bins. The choose gives us the number of ways of dividing the balls between landing in blue and landing in red. The value x^{n-k} gives us the number of ways that $n-k$ balls can be thrown into the red bins. The value y^k gives us the number of ways that k balls can be thrown into the blue bins. Therefore for each way of allocating k balls to blue and rest to red, we have x^{n-k} ways for them to land in red and y^k ways for them to land into blue. Therefore the product gives the number of ways to throw the balls given that k land in the blue bins and $n-k$ land in the red bins.

If we sum up the terms on the right hand side, we get the total number of ways of throwing the balls into the bins.

8. **Crazy Balls and Bins**
Imagine you had 5 distinct bins and randomly threw 7 identical balls into the bins with uniform probability.

(a) What is likelihood that the first bin has at least 3 balls in it?

(b) What is likelihood that the first bin has exactly 3 balls in it?

(c) What is likelihood that at least one bin has exactly 3 balls in it?

(d) What is likelihood that the at least one bin has at least 3 balls in it?

Solution:

(a) $\sum_{k=3}^{7} \binom{7}{k}(1/5)^k(4/5)^{7-k}$

(b) $\binom{7}{3}(1/5)^3(4/5)^4$

(c) Use inclusion-exclusion. Let A_i be the event that bin i has exactly 3 balls. Then $\sum_{i=1}^{5} \Pr[A_i] = 5\binom{7}{3}(1/5)^3(4/5)^4$. We have to subtract the events $A_i \cap A_j$, of which there are $\binom{5}{2}$. We have $\Pr[A_i \cap A_j] = 7!/(3!)^2(1/5)^6(3/5)$. Therefore our answer is

$$5\binom{7}{3}\left(\frac{1}{5}\right)^3\left(\frac{4}{5}\right)^4 - \binom{5}{2}\frac{7!}{3!3!}\left(\frac{1}{5}\right)^6\frac{3}{5}$$

(d) We build off our answer above. For the case where k balls fall in bin i, where $k = 4,5,6,7$, then we have a probability $\binom{7}{k}(1/5)^k(4/5)^{7-k}$. Now, these events are disjoint and the probabilities add. Finally, note that there is one case in which we overcount: when 3 balls land in bin i and 4 balls land in bin j. The probability of this case is $\binom{5}{2}\binom{7}{3}(1/5)^7$.

$$5\binom{7}{3}\left(\frac{1}{5}\right)^3\left(\frac{4}{5}\right)^4 - \binom{5}{2}\frac{7!}{3!3!}\left(\frac{1}{5}\right)^6\frac{3}{5} + 5\sum_{k=4}^{7}\binom{7}{k}\left(\frac{1}{5}\right)^k\left(\frac{4}{5}\right)^{7-k} - \binom{5}{2}\binom{7}{3}\left(\frac{1}{5}\right)^7$$

9. **Teams and Leaders**
Prove the following identities using a combinatorial proof.

(a) $\sum_{k=0}^{n} \binom{n}{k}^2 = \binom{2n}{n}$

(b) $\sum_{k=1}^{n} k\binom{n}{k}^2 = n\binom{2n-1}{n-1}$

Solution:

(a) Imagine you are a teacher picking students to be on a team for some competition. You have $2n$ students, n of whom are boys and the other n are girls.

RHS: This is simply the number of ways you can pick n students to be on the team.
LHS: We begin by noticing that $\binom{n}{k}^2 = \binom{n}{k} * \binom{n}{n-k}$. This product gives us the number of ways of picking k girls and $n-k$ boys to be on the team. We add up all the products involving anywhere from 0 girls all the way to n girls. This gives us the total number of ways to pick a team of n students.

(b) Imagine the same scenario as part A except now you have to choose a female team leader amongst the n students on the team.

RHS: This is the number of ways of picking the team leader multiplied with the number of ways of picking the rest of the team from the remaining students. The product gives the total number of teams with a female leader.
LHS: We begin similarly by noticing that $k * \binom{n}{k}^2 = k * \binom{n}{k} * \binom{n}{n-k}$. Here as before we are picking k girls and $n-k$ boys to be on the team. However amongst the k girls on the team, we choose one of them to be the team leader. We add up all the products involving anywhere from 0 girls all the way to n girls. This gives us the total number of ways to pick a team of n students with a female leader.

10. **Finicky Bins**

If a bin has at least 5 balls in a bin, the 5 balls will fall out and not be counted (e.g., 6 balls in a bin is the same as 1). Compute the number of ways to distribute 7 indistinguishable balls among 4 bins.

Solution:

Consider a normal bin, in which balls do not disappear. With stars and bars we see there are $\binom{10}{3}$ ways to distribute the balls. What if one bin has ≥ 5 balls? There are 4 ways to choose which bin has ≥ 5 balls, and once we throw 5 balls into that bin, we are left to distribute 2 balls among 4 bins in $\binom{5}{3}$ ways.

Now, for the bin in which balls *do* disappear. There are $\binom{10}{3} - 4\binom{5}{3}$ ways to distribute the balls such that no balls disappear. There are $4\binom{5}{3}$ ways to distribute the balls such that 5 balls disappear, except that no matter where the disappearing balls are, there the resulting distribution of balls is the same. Therefore, we have to divide by 4 and we obtain $\binom{5}{3}$ ways to distribute the balls such that 5 balls disappear. In total, we have

$$\binom{10}{3} - 4\binom{5}{3} + \binom{5}{3} = \binom{10}{3} - 3\binom{5}{3}$$

11. **Bag of Coins**

Your friend Forest has a bag of n coins. You know that k are biased with probability p (i.e., These coins have probability p of being heads). Let F be the event that Forest picks a fair coin, and let B be the event that Forest picks a biased coin.. Forest draws three coins from the bag, but he does not know which are biased and which are fair.

(a) What is the probability of FFH?

(b) What is the probability of picking at least two fair coins?

(c) Given that Forest flips the second coin and sees heads, what is the probability that this coin is biased?

Solution:

(a) The probability of picking F for the first coin is

$$\frac{n-k}{n}$$

The probability of picking F for the second coin, after picking one fair coin already is

$$\frac{n-k-1}{n-1}$$

The probability of picking B for the third coin is

$$\frac{k}{n-2}$$

Thus, the probability of picking the exact sequence FFB is

$$\frac{(n-k)(n-k-1)k}{n(n-1)(n-2)}$$

(b) Note that the probability of picking any sequence of two fair coins and a biased coin is the same. It is in fact the probability from part a. We need to multiply by the number of arrangements of biased and fair coins, however. So, the probability of picking any sequence with two fair coins is

$$\binom{3}{1}\frac{(n-k)(n-k-1)k}{n(n-1)(n-2)}$$

We additionally need to consider the probability of getting 3 fair coins.

$$\frac{(n-k)!(n-3)!}{n!(n-k-3)!}$$

We simply sum the two to get our answer:

$$\binom{3}{1}\frac{(n-k)(n-k-1)k}{n(n-1)(n-2)}+\frac{(n-k)!(n-3)!}{n!(n-k-3)!}$$

(c) We can apply Bayes' Rule. Let H denote the event that Forest sees heads.

$$\Pr(B|H)=\frac{\Pr(H|B)\Pr(B)}{\Pr(H)}$$

Note that $\Pr(H|B)=p$ and that $\Pr(B)=\frac{k}{n}$. We can now compute the denominator. Using the law of total probability, we can expand $\Pr(H)$

$$\begin{aligned}\Pr(H)&=\Pr(H|B)\Pr(B)+\Pr(H|F)\Pr(F)\\&=p\frac{k}{n}+\frac{1}{2}\frac{n-k}{n}\\&=\frac{2pk+n-k}{2n}\end{aligned}$$

We now combine both parts to get our answer:

$$\frac{p(k/n)}{(2pk+n-k)/(2n)} = \frac{2pk}{2pk+n-k}$$