MT2 Review

CS 70, Summer 2019

Bonus Lecture, 8/2/19

The Tribe Has Spoken...

We'll go through these questions in order:

- Short Answer (a selection of more frequently missed problems)
- Probability (Shuffling): parts (b) and (c)
- Recursive Enumerability

2a) Suppose I have a deck of 52 cards and I lost 5 cards in the deck because I was careless. I shuffle the deck and take the top card. What is the probability that the card is a spade?

$$\frac{13}{52} = \frac{1}{4}$$

* no info on 5 lost cards.

$$\frac{\quad}{52!}$$

2c) Find the number of non-negative integer solutions to $x_1 + x_2 + x_3 = 30$ where we have that at least one $x_i \leq 5$.

Total sols to $x_1 + x_2 + x_3 = 30$

Stars + bars! 30 $*$'s, 2 bars

$$\binom{32}{2}$$

Complement: $x_1 + x_2 + x_3 = 30$ s.t. $x_i \geq 6 \ \forall i$

fix 6 $*$'s in each

$$\underbrace{6* \ *}_{x_1} \ \big| \ \underbrace{6* \ *}_{x_2} \ \big| \ \underbrace{6* \ * \ *}_{x_3}$$

12 'free" stars
2 bars

$$\binom{14}{2}$$

$\Rightarrow$ Ans: $\binom{32}{2} - \binom{14}{2}$

2e) Suppose we want to send n packets, and we know that our channel drops a fraction p of our packets, where $0 < p < 1$. Using the R-S encoding from class, how many *total* packets should we send?

n packets + k "padding" packets.

Erasures: $\underbrace{k}_{\#\text{ extra packets}} = \underbrace{(n+k)p}_{\#\text{ erasures.}}$

Solve for k: $\frac{np}{1-p} = k$

$n+k = \frac{n}{1-p}$

2l) A dormitory has $n \geq 4$ students, all of whom like to gossip. One of the students hears a rumor, and tells it to one of the other $n - 1$ students picked at random. After that, each student who hears the rumor tells it to another student picked uniformly at random, excluding themselves and the student who just told them the rumor. Let p_r be the probability that the rumor is told at least r times without coming back to a student who has already heard it.

2l) Continued...

R_i = rumor is told for i^{th} time w/out going back to someone.

$$\mathbb{P}[R_1 \cap R_2 \cap R_3 \dots \cap R_r]$$

$$= \mathbb{P}[R_1] \times \mathbb{P}[R_2 | R_1] \times \mathbb{P}[R_3 | R_1, R_2] \times \dots$$

$$= 1 \times 1 \times \frac{n-3}{n-2} \times \frac{n-4}{n-2} \times \dots$$

can't tell person #1

7a) Given a playlist, the shuffle feature on Apple Music will play songs as a series of independent *shuffle cycles*. In each shuffle cycle, all songs in the list will be reordered, with each ordering equally likely. For instance, for a playlist of four songs a, b, c, d, one possible sequence of plays could be

$$a\ b\ c\ d \mid b\ d\ c\ a \mid d\ a\ c\ b \mid \ldots$$

where we use $\mid$ to separate the shuffle cycles.

7ai) Suppose I have an Apple Music playlist with **exactly two songs**, a and b. I have this playlist on shuffle while I'm away, so when I return, I could be at any position within a shuffle cycle with equal probability. When I return, a is playing. What is the probability that the next song is b?

7aii) The next song played happened to be b. What is the probability that when I returned (i.e. when a was playing), it was the start of a shuffle cycle?

7b) Spotify's shuffle feature works a little differently. It instead selects any copy of any song from the playlist uniformly at random to play each time. I have a Spotify playlist with 5 copies of song a, 3 copies of song b, and 2 copies of song c (10 copies total).

select song in a playlist
uniformly at random
with replacement.

7bi) I shuffle my Spotify playlist for 6 song plays. If I *ignore their play order*, how many different sets of 6 plays could I have gotten? Give your answer as an integer.

sets: $\{a, a, a, b, b, c\} \neq \{a, a, b, b, c, c\}$

$$\underbrace{* * *}_{a} \; | \; \underbrace{* *}_{b} \; | \; \underbrace{*}_{c}$$

6 stars

2 bars

$$\binom{8}{2} = 28$$

7bii) What is the probability that across the 6 songs played on my Spotify shuffle, I get song a twice, song b twice, and song c twice? (*You may leave your answer unsimplified.*)

1) aa bb cc
ba cb ac } count these. $\frac{6!}{2!2!2!}$

2) prob. of an individual seq.

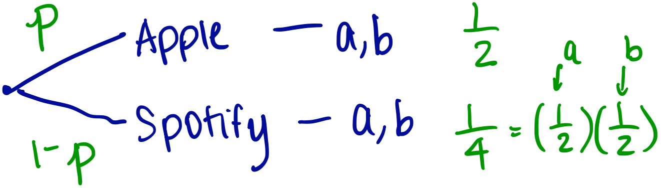
$\mathbb{P}r[aabbcc] = (0.5)^2(0.3)^2(0.2)^2$

$\Rightarrow \frac{6!}{2!2!2!}(0.5)^2(0.3)^2(0.2)^2$

7c) YouTube Music's (YTM) shuffle functionality is somewhere in between Apple Music's and Spotify's. Specifically, given a playlist of n songs, YTM will still play songs as a series of *independent* length-n shuffle cycles. However, each YTM cycle will behave like Apple Music's shuffle feature (from part (a)) with probability p, and behave like Spotify's shuffle feature (from part (b)) with probability $1 - p$.

I have a playlist with **exactly two songs** (one copy of each), a and b. I return when a (YTM) shuffle cycle is about to begin. (*Note: Each of the following answers may be in terms of* p.)

7ci) What is the probability that the first song I hear is a and the second is b?

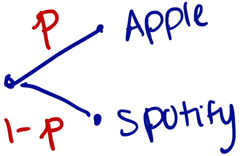


Total probability:

$$\mathbb{P}[a \text{ then } b] = \mathbb{P}[(a \text{ then } b) \cap \text{Apple}] + \mathbb{P}[(a \text{ then } b) \cap \text{Spotify}]$$

$$= p\left(\frac{1}{2}\right) + (1-p)\left(\frac{1}{4}\right) = \frac{2p+1-p}{4} = \frac{p+1}{4}$$

7cii) What is the probability that the second song I hear is b given that the first is a?



first	~~second~~
a	$\frac{1}{2}$
a	$\frac{1}{2}$

last part.

$$\mathbb{P}[\text{second } b \mid \text{first } a] = \frac{\mathbb{P}[\text{second } b \cap \text{first } a]}{\mathbb{P}[\text{first } a]}$$

$$\mathbb{P}[\text{first } a] = \frac{p}{2} + \frac{1-p}{2} = \frac{1}{2}$$

$$= \frac{\frac{p+1}{4}}{\frac{1}{2}} = \frac{p+1}{2}$$

5) A “halting converter” for a problem A is a program `C` that takes an instance of A as input and:

program string "x"

- If the correct answer for `x` is true, `C(x)` outputs a pair `(P,y)` such that `P(y)` halts.
- If the correct answer for `x` is false, `C(x)` outputs a pair `(P,y)` such that `P(y)` loops forever.

5ai) Suppose we have a program `C` that is a halting converter for A. Fill in the description of `R` such that it is a recognizer for A.

Recognizer:

1) If answer to x true:
   R returns true
2) If answer to x false:
   R return false
   or
   loop.

R(x):

P, y = C(x)
P(y)
return true.

5aii) Prove that if the correct answer for x is true, R(x) will return true in finite time.

$R(x)$:

$P, y = C(x)$
$P(y)$
return true.

If x true
$\Rightarrow P(y)$ halts
$\Rightarrow$ return true ✓

5aiii) Prove that if the correct answer for x is false, R(x) will $\underbrace{\text{return false}}_{\text{doesn't happen}}$ or loop forever.

If x false
$\Rightarrow P(y)$ loop
$\Rightarrow R(x)$ loop ✓

5bi) Suppose we have a recognizer `R` for A. Fill in the description of `P` such that, for an instance `x` of the problem A, `P(x)` halts if and only if the correct answer for `x` is true.

```
P(x):
  y=R(x)
  if y is true:
      return
  else:
      while (0==0) ← loop
                       forever.
```

5bii) Prove that if the correct answer for `x` is true, `P(x)` halts.

```
P(x):
  y = R(x)
  if y is true:
      return
  else:
      while (0 == 0)
```

x true

⇒ R(x) true

⇒ return ✓

5biii) Prove that if the correct answer for `x` is false, `P(x)` loops forever.

x false

⇒ 1) R(x) false ⇒ else clause

2) R(x) loops ⇒ y = R(x) will loop

⇒ P(x) doesn't halt

5biv) Fill in the description of `C` below such that it is a halting converter for A. You may use the program `P` from part (bi), even if you did not complete that part.

C(x):
    return P, x