CS109 Lecture 17 Worksheet: Algorithmic Analysis

Conditional Expectation, the Law of Total Expectation, and Expected Runtime

Summer 2026

Lecture 17 Worksheet: Algorithmic Analysis

Topics: conditional expectation, the law of total expectation, expected runtime of code, analyzing recursive code, indicator random variables, and the coupon collector's problem.

How to use this sheet. Work in small groups. A TA and the instructor will circulate to help. We will go over selected solutions on the board. You do not need to finish every part; aim for understanding over speed. Keep fractions exact where you can.

Problem 1 (Review of Lecture 16): Reading a Bootstrap

To test whether a UI change affects session length, you pool both groups' data and bootstrap 10,000 differences of means under the null hypothesis. Your observed difference is 2.1 minutes, and 140 of the 10,000 resampled differences are at least as extreme (i.e., $|\text{diff}| \geq 2.1$).

(a) What is the p-value?

(b) In one sentence, interpret it. Is this strong or weak evidence against the null hypothesis?

Problem 2: Conditional Expectation from a Joint Table

Let X be the number of bugs found in a code review and Y the number of reviewers, with joint PMF:

	$X=0$	$X=1$	$X=2$
$Y=1$	0.15	0.10	0.05
$Y=2$	0.10	0.30	0.30

(a) Compute $E[X \mid Y=1]$ and $E[X \mid Y=2]$.

(b) Use the law of total expectation to compute $E[X]$ from your answers in (a).

(c) Verify your answer by computing $E[X]$ directly from the marginal PMF of X.

Problem 3: Expected Runtime with a Cache Hierarchy

A web request for an image is served from one of three places: the browser cache with probability 0.5 (taking 1 ms), a CDN edge server with probability 0.35 (taking 40 ms), or the origin server with probability 0.15 (taking 300 ms). Let T be the time to serve a request.

(a) Using the law of total expectation, compute $E[T]$.

(b) Compute $E[T \mid \text{browser cache miss}]$.

(c) A product manager says "the page is fast: average image load is under 60 ms." Give one reason why $E[T]$ alone might be a misleading summary of user experience here.

Problem 4: Roll Until Big (pset5: code_analysis_dice)

Consider the following function, which simulates repeatedly rolling a 6-sided die (where each integer value from 1 to 6 is equally likely to be "rolled") until a value ≥ 3 is "rolled".

```
def roll():
  total = 0
  while True:
    # equally likely to return 1,...,6
    roll = random_integer(1, 6)
    total += roll
    # exit condition
    if roll >= 3: break
  return total
```

Let X be the value returned by the function `roll()`. What is $E[X]$? (Hint: condition on the first roll and let the law of total expectation set up an equation for $E[X]$.)

Problem 5: Analyzing Recursive Code

```
def mystery():
    x = random_integer(1, 4)   # equally likely values 1..4
    if x == 1:
        return 2
    elif x == 2:
        return 1 + mystery()
    else:
        return 3 + mystery()
```

Let Y be the value returned by `mystery()`. Set up an equation for $E[Y]$ using the law of total expectation, and solve it.

Problem 6: Hash Table Collisions

You insert $m = 20$ keys into a hash table with $n = 10$ buckets. Each key hashes independently and uniformly to one of the buckets.

(a) Let X = the number of *empty* buckets. Using indicator variables, compute $E[X]$. (A numeric answer is fine; it is about 1.22.)

(b) Let C = the number of *colliding pairs* of keys (pairs that land in the same bucket). Define one indicator per pair of keys and compute $E[C]$.

(c) The indicators in (a) are *not* independent (if 9 buckets are empty, the last one isn't). Why is that not a problem for your calculation?

Problem 7: Complete the Set

A trading-card game has $n = 8$ distinct hero cards. Each pack you open contains one hero, equally likely to be any of the 8, independent of other packs. Let X be the number of packs opened until you own all 8 heroes.

(a) Write X as a sum $X = X_0 + X_1 + \cdots + X_7$, where X_i is the number of packs to find a *new* hero once you own i distinct heroes. What is the distribution of X_i?

(b) Compute $E[X]$. (A numeric answer is fine; an exact sum earns style points.)

(c) Compute the expected number of packs to reach just 4 distinct heroes. Why is this so much smaller than half of $E[X]$?

Challenge Problem (optional): Llama-Flu (pset5: code_analysis_flu)

Our ability to fight contagious diseases depends on our ability to model them. One person is exposed to llama-flu (a made up disease). The method below returns the number of individuals who will get infected.

```
def num_infected():
    """
    Returns the number of people infected by one individual
    """
    # most people are immune to llama-flu
    immune = bernoulli(p = 0.99)
    if immune: return 0

    # people who are not immune spread the disease far
    spread = 0

    # they make contact with k people (up to 100)
    k = binomial(n = 100, p = 0.25)
    for i in range(k):
      spread += num_infected()

    # total infections should include this individual
    return spread + 1
```

What is the expected return value of `num_infected()`?