CSE 2331 Homework 5

Please do not write on the question sheet. You must show all work to receive any credit.

1. You have m bins, and n balls. Each time you throw a ball, it lands in one of the m bins with probability $1/m$. What is the expected number of bins that contain at least 2 balls?

   To calculate this we will again use the indicator variable method where X_i indicates that the i-th bin contains at least two balls. We will need to know $\mathbb{E}[X_i]$ for each i which we know is equal to the probability that the given bin contains at least 2 balls.

   Recall that we are throwing n balls into m bins where the ball must land in one of the m bins uniformly at random from amongst all bins. Let $B(i)$ denote the number of balls in bin i. The probability that k of the balls end up in any bin is, therefore,

$$\Pr[\{B(i) = k\}] = \binom{n}{k} \left(\frac{1}{m}\right)^k \left(1 - \frac{1}{m}\right)^{n-k}.$$

   We are interested in having *at least* 2 balls in the bin. There is a complicated way to write this as

$$\Pr[\{B(i) \geq 2\}] = \sum_{k=2}^{n} \Pr[\{B(i) = k\}] = \sum_{k=2}^{n} \left[\binom{n}{k} \left(\frac{1}{m}\right)^k \left(1 - \frac{1}{m}\right)^{n-k} \right],$$

   but we know better and can make our lives easier using the event algebra.

   Bin i having at least 2 balls is the complement event to *not* having 0 or 1 balls. Therefore,

$$\Pr[\{B(i) \geq 2\}] = 1 - \Pr[\{B(i) = 1\}] - \Pr[\{B(i) = 0\}].$$

   We already know the probability the bin is empty, $\Pr[\{B(i) = 0\}]$, from the previous part. It's $(1 - 1/m)^n$. Using the formula,

$$\Pr[\{B(i) = 1\}] = \binom{n}{1} \left(\frac{1}{m}\right)^1 \left(1 - \frac{1}{m}\right)^{n-1} = \frac{n}{m} \left(1 - \frac{1}{m}\right)^{n-1}.$$

   Therefore, for each bin ($1 \leq i \leq m$),

$$\mathbb{E}[X_i] = 1 - \left(1 - \frac{1}{m}\right)^n - \frac{n}{m} \left(1 - \frac{1}{m}\right)^{n-1}.$$

   It follows that,

$$\mathbb{E}[X] = \sum_{i=1}^{m} \mathbb{E}[X_i] = m - m \left(1 - \frac{1}{m}\right)^n - n \left(1 - \frac{1}{m}\right)^{n-1}.$$

At this point we are done, but with some further simplification we get

$$\mathbb{E}[X] = m - \left(1 - \frac{1}{m}\right)^{n-1} \left(m\left(1 - \frac{1}{m}\right) + n\right) = m - (m + n - 1)\left(1 - \frac{1}{m}\right)^{n-1}.$$

That's the end of the solution.

This next part is just enrichment. It's not part of the problem but in writing up the solutions I used some tricks to check if the answer I wrote down makes sense.

Okay, so, how can we check a little? Well here's one: let's imagine that n the number of balls is really big compared to m, then every bin should have at least two balls by holding m constant and taking a limit on n. Indeed,

$$\lim_{n\to\infty} \mathbb{E}[X] = \lim_{n\to\infty} \left[m - (m + n - 1)\left(1 - \frac{1}{m}\right)^{n-1}\right] \to m.$$

Because m is a constant and n is getting big we want to say that the whole right term goes to zero. The only tricky part of the limit is being sure that $\lim_{n\to\infty} n\left(1 - \frac{1}{m}\right)^{n-1}$ is actually 0 because n is getting big as $\left(1 - \frac{1}{m}\right)^{n-1}$ is getting small. If $m = 1$ this is obviously zero. For $m > 1$ apply L'Hopital's rule. Observe,

$$\lim_{n\to\infty} \frac{n}{\left(1 - \frac{1}{m}\right)^{-(n-1)}} = \lim_{n\to\infty} \frac{1}{-\left(1 - \frac{1}{m}\right)^{-(n-1)} \log(1 - \frac{1}{m})} = \lim_{n\to\infty} \frac{\left(1 - \frac{1}{m}\right)^{n-1}}{-\log(1 - \frac{1}{m})} = 0$$

Don't let the ugly expression intimidate you, $(1 - 1/m)$ is just acting as a constant smaller than 1, so we have a constant smaller than 1 being raised to a high power as n becomes large.

There's another check we can check: if the number of bins is really big compared to the number of throws this should go to zero. This is much trickier to check. We have to deal with the unclear (to me, at least) limit

$$\lim_{m\to\infty} \mathbb{E}[X] = \lim_{m\to\infty} \left[m - (m + n - 1)\left(1 - \frac{1}{m}\right)^{n-1}\right]$$

With a little rewriting we get

$$\begin{aligned}\lim_{m\to\infty} \mathbb{E}[X] &= \left[\lim_{m\to\infty} m - (m + n - 1)\left(1 - \frac{1}{m}\right)^{n-1}\right] \\ &= \lim_{m\to\infty} \left[m\left(1 - \left(1 - \frac{1}{m}\right)^{n-1}\right) - (n - 1)\left(1 - \frac{1}{m}\right)^{n-1}\right]\end{aligned}$$

Now since $m \to \infty$ while we hold n constant, that right term just goes to $-(n-1)$. This leaves us with the left term which needs to be going to $n-1$ if we want to get zero. Remarkably, it is. Let $y = 1/m$ and consider,

$$m\left(1 - \left(1 - \frac{1}{m}\right)^{n-1}\right) = \frac{1}{y}\left(1 - (1-y)^{n-1}\right).$$

With this change of variable we can take the limit as $y \to 0$ instead. Let's look at the Taylor expansion of this expression around $y = 0$. We get

$$(n-1) - \frac{1}{2}(n-2)(n-1)y + O(y^2).$$

Taking $y \to 0$ all that's left is $(n-1)$ so

$$\begin{aligned}\lim_{m\to\infty} \mathbb{E}[X] &= \lim_{y\to 0}\left[(n-1) - \frac{1}{2}(n-2)(n-1)y + O(y^2)\right] - (n-1) \\ &= (n-1) - (n-1) = 0.\end{aligned}$$

2. In the code below, we define random(n) which returns $i \in [n]$ with probability $1/n$, in $\Theta(1)$-time, and CoinFlip simply returns 0 or 1, each with probability $1/2$.

```
1  import math
2  import random
3
4  # returns a random integer in [1, n]
5  def random(n):
6      return random.randint(1, n)
7
8  # returns a random integer in [0, 1]
9  def coinflip():
10     return random(2) - 1
```

For each function that follows, give each function's best case, worst case, and average case asymptotic running times. Justify your answer.

(a)

```python
import random
import math

def f1(xs):
    s = 0
    n = len(xs)
    k = random(n)

    m = int(math.sqrt(n))
    if k == 1:
        m = n // 2

    for j in range(m):
        s *= xs[j]

    return s
```

Proof. First, note that the algorithm takes $\Theta(m)$ time for one of two possible values of m. In the best case, $k \neq 1$ giving $m = \sqrt{n}$ for $\Theta(\sqrt{n})$-time. The worst case happens when $k = 1$ implying $m = n/2$ for $\Theta(n)$-time.
To figure out the average case, let $\Omega = \{k = 1, k \neq 1\}$ be the event space with probabilities $\Pr[k = 1] = 1/n$, $\Pr[k \neq 1] = 1 - 1/n$. By conditional expectation,

$$\begin{aligned} E[T(n)] &= E[T(n)|k = 1] \cdot \Pr[k = 1] + E[T(n)|k \neq 1] \cdot \Pr[k \neq 1] \\ &= \Theta(n) \cdot \Pr[k = 1] + \Theta(\sqrt{n}) \cdot \Pr[k \neq 1] \\ &= \Theta(n) \cdot 1/n + \Theta(\sqrt{n}) \cdot (1 - 1/n) \end{aligned}$$

Thus, filling in for $\Theta(n)$ and $\Theta(\sqrt{n})$, there exists c_1, c_2, c_3, and $c_4 \in \mathbb{R}^+$, such that for sufficiently large n,

$$c_1 n \cdot \frac{1}{n} + c_2\sqrt{n}\left(1 - \frac{1}{n}\right) \geq E[T(n)] \geq c_3 n \cdot \frac{1}{n} + c_4\sqrt{n}\left(1 - \frac{1}{n}\right).$$

Canceling and dividing through by $\sqrt{n}$ gives

$$c_1 + c_2\left(1 - \frac{1}{n}\right) \geq \frac{E[T(n)]}{\sqrt{n}} \geq c_3 + c_4\left(1 - \frac{1}{n}\right).$$

Taking a limit as $n \to \infty$, we see that

$$c_1 + c_2 \geq \lim_{n\to\infty} \frac{E[T(n)]}{\sqrt{n}} \geq c_3 + c_4,$$

and therefore, $E[T(n)] \in \Theta(\sqrt{n})$.

□

(b)

```
1  import random
2  import math
3
4  def f2(xs):
5      s = 0
6      n = len(xs)
7      k = random(n)
8
9      m = int(math.log2(n))
10     if k < n**0.25:
11         m = n // 2
12
13     for i in range(n//2):
14         for j in range(m):
15             s = s * xs[j] + xs[i]
16
17     return s
```

Proof. First note that the algorithm takes $\Theta(n \cdot m)$ time for one of two possible values of m. A best case $k = 1$ and we get $m = \log_2(n)$ for $\Theta(n \log(n))$-time. On the other hand, $k = n$ is a worst case and we get $m = n/2$ for $\Theta(n^2)$-time.

To figure out the average case, let

$$\Omega = \left\{k < n^{\frac{1}{4}}, k \geq n^{\frac{1}{4}}\right\}$$

be the event space with probabilities $\Pr[k < n^{\frac{1}{4}}] = \frac{n^{1/4}}{n} = \frac{1}{n^{3/4}}$, $\Pr[k \geq n^{\frac{1}{4}}] = 1 - \frac{1}{n^{3/4}}$. By conditional expectation,

$$\begin{aligned} E[T(n)] &= E[T(n)|k < n^{\frac{1}{4}}] \cdot \Pr[k < n^{\frac{1}{4}}] + E[T(n)|k \geq n^{\frac{1}{4}}] \cdot \Pr[k \geq n^{\frac{1}{4}}] \\ &= \Theta(n^2) \cdot \Pr[k < n^{\frac{1}{4}}] + \Theta(n \log(n)) \cdot \Pr[k \geq n^{\frac{1}{4}}] \\ &= \Theta(n^2) \cdot \frac{1}{n^{3/4}} + \Theta(n \log(n)) \cdot (1 - n^{-3/4}) \end{aligned}$$

Thus,

$$c_1 n^{5/4} + c_2 n \log(n)(1 - n^{3/4}) \geq E[T(n)] \geq c_3 n^{5/4} + c_4 n \log(n)(1 - n^{3/4}).$$

Dividing through by $n^{5/4}$ and taking a limit as $n \to \infty$, we see that

$$c_1 \geq \lim_{n \to \infty} \frac{E[T(n)]}{n^{5/4}} \geq c_3,$$

and therefore, $E[T(n)] \in \Theta(n^{5/4})$.

□

(c)

```
1  import random
2
3  def f3():
4      s = 0
5      while coinflip() == 0:
6          s += 1
7      return s
```

Solution 1. First note that a best case is $\Theta(1)$ where we exit after some constant number of *while*-body executions. The worst case is unbounded, as it may run forever.

Let N denote the number of executions of the while-body. Then the expected running time, $E[T]$, is $c \cdot E[N] + c'$ Recall that for random variables X taking values in $\mathbb{N}_0$,

$$E[X] = \sum_{i=0}^{\infty} (i \cdot \Pr[X = i]) = \sum_{i=1}^{\infty} \Pr[X \geq i].$$

It follows that

$$E[N] = \sum_{i=1}^{\infty} \Pr[N \geq i] = \sum_{i=1}^{\infty} \left(\frac{1}{2}\right)^i = \frac{1}{1 - \frac{1}{2}} - 1 = 1$$

since the probability that the loop-body executes *at least* i times is $(1/2)^i$ and the resulting sum is a geometric series without the first term. Thus, $E[T] = c \cdot E[N] + c' \in \Theta(1)$. □

Solution 2. First note that the runtime of the algorithm is $\Theta(k+1)$ where k is the number of times the while-body executes.

In a generic run of the algorithm, the while-body executes k times before terminating for any $k \in \mathbb{N}_0$. Let E^kT denote such an event (that the while-body runs k times before terminating).

Let $\Omega = \{E^kT : k \in \mathbb{N}_0\}$ and observe that $\Pr[E^kT] = (\frac{1}{2})^k \cdot \frac{1}{2}$. This is because the probability of executing the while-body each of the k times is 0.5 each time, followed by the probability of terminating which is also 0.5.

Therefore,

$$E[T] = \sum_{e \in \Omega} T(e) Pr[e] = \sum_{k=0}^{\infty} T(E^kT) Pr[E^kT].$$

Using the fact that $T(k) \in \Theta(k+1)$, and $\Pr[E^kT] = (\frac{1}{2})^{k+1}$, it follows that

$$c_2 \sum_{k=0}^{\infty} (k+1) \left(\frac{1}{2}\right)^{k+1} \geq E[T] \geq c_1 \sum_{k=0}^{\infty} (k+1) \left(\frac{1}{2}\right)^{k+1},$$

for some $c_1, c_2 \in \mathbb{R}^+$.

Note that we may reindex the sums,

$$\sum_{k=0}^{\infty}(k+1)\left(\frac{1}{2}\right)^{k+1} = \sum_{k=1}^{\infty} k\left(\frac{1}{2}\right)^{k} \in \Theta(1).$$

You may stop there. However you may be wondering why this sum is constant. We worked out this sum in class. Specifically, let $S_t^{(n)}$ denote sums of the form

$$\sum_{i=1}^{n} i \cdot t^i.$$

Then,

$$S_t^{(n)} = \sum_{i=1}^{n} i \cdot t^i = t \cdot \frac{\mathrm{d}}{\mathrm{d}t} \sum_{i=0}^{n} t^i = t \cdot \frac{\mathrm{d}}{\mathrm{d}t} \frac{1-t^{n+1}}{1-t},$$

where the last equality follows by replacing the geometric series with its closed-form formula.

Carrying out the derivative we see that

$$S_t^{(n)} = t \cdot \frac{(1-t)(-(n+1)t^n) + (1-t^{n+1})}{(1-t)^2} = t \cdot \frac{1-t^{n+1}}{(1-t)^2} - \frac{(n+1)t^{n+1}}{1-t}.$$

For $0 < t < 1$, as $n \to \infty$,

$$S_t^{(\infty)} = \frac{t}{(1-t)^2}.$$

So, in particular, $S_{1/2}^{(\infty)} = 2$, and $S_t^{(\infty)} \in \Theta(1)$ for all $0 < t < 1$. □

(d)

```
1  import random
2
3  def f4(xs):
4      i = len(xs)
5      while i > 0:
6          if random(10000) == 1:
7              return -1
8          i -= 1
9      return i
```

Proof. This one is a little tricky. Even though it is not explicitly recursive we can model it recursively. (We can also apply some of the other techniques of problem 7*b* from the previous Homework here.)

First note that a best case is $\Theta(1)$ where we exit after some constant number of *while*-body executions (say during the first). The worst case is linear in n (the length of xs).
Let $T(n)$ denote the time spent in the *while*-loop, and let q be the probability that we quit the *while*-loop on line 5. A generic execution of the *while*-body either exits in constant time with probability q or n is reduced by 1 and we try again.

$$E[T(n)] = (1-q) \cdot E[T(n-1)] + q \cdot \Theta(1).$$

After one substitution,

$$E[T(n)] \approx (1-q)^2 \cdot E[T(n-2)] + q \cdot (1-q) \cdot c + q \cdot c$$

After two substitutions,

$$E[T(n)] \approx (1-q)^3 \cdot E[T(n-3)] + q \cdot (1-q)^2 \cdot c + q \cdot (1-q) \cdot c + q \cdot c$$

After $k-1$ substitutions, we have that

$$E[T(n)] \approx (1-q)^k \cdot E[T(n-k))] + c \cdot q \sum_{i=1}^{k-1} \cdot (1-q)^i$$

This terminates when $n-k=1$. That is, when $k=n-1$. So plugging back in,

$$E[T(n)] \quad \approx \quad (1-q)^{n-1} \cdot E[T(1)] + c \cdot q \sum_{i=1}^{n-2} (1-q)^i \in \Theta(1).$$

This is because $E[T(1)] \in \Theta(1)$, the sum is bounded above by a geometric series, and both q and $1-q$ are between 0 and 1.

□

(e)

```
1  import random
2
3  def f5(xs):
4      s = 0
5      n = len(xs)
6      if n <= 10:
7          return xs[1]
8
9      k = random(100)
10     if random(n**k) == 1:
11         for j in range(n**k):
12             s = s * xs[j] + xs[j]
13
14     return s
```

Proof. First, note that a best case is $\Theta(1)$ where we exit without running the loop. The worst case is $\Theta(n^{100})$ when $k = 100$ and *random*(n^k) returns 1.

To figure out the average case, suppose that k is fixed and let R_k denote the random variable representing the outcome of random(n^k). Observe that $E[T|R_k = 1]$ is $\Theta(n^k)$ and $E[T|R_k \neq 1]$ is $\Theta(1)$. Further, note that $\Pr[R_k = 1] = 1/n^k$ since *random* returns a value uniformly in the range $1, \ldots, n^k$. By conditional expectation,

$$E[T] = E[T|R_k = 1] \cdot \Pr[R_k = 1] + E[T|R_k \neq 1] \cdot \Pr[R_k \neq 1].$$

Thus,

$$E[T] = \Theta(n^k) \cdot \frac{1}{n^k} + \Theta(1) \cdot \left(1 - \frac{1}{n^k}\right).$$

It follows immediately that there exists constants c_1, c_2, c_3, and c_4, such that for sufficiently large n,

$$c_1 \cdot n^k \cdot \frac{1}{n^k} + c_2 \cdot \left(1 - \frac{1}{n^k}\right) \geq E[T] \geq c_3 \cdot n^k \cdot \frac{1}{n^k} + c_4 \cdot \left(1 - \frac{1}{n^k}\right).$$

Canceling the fractions and taking limit as $n \to \infty$,

$$c_1 + c_2 \geq E[T] \geq c_3 + c4.$$

Thus, $E[T] \in \Theta(1)$ for any k. Thus $E[T] \in \Theta(1)$. □

(f)

```
1  import random
2
3  def f6(xs):
4      s = 0
5      n = len(xs)
6      while random(n) != random(n):
7          s += xs[random(n) - 1]
8      return s
```

Proof. First note that a best case is $\Theta(1)$ where we exit after some constant number of *while*-body executions (say 0). The worst case is unbounded, as it may run forever.

Let $\Omega = \{(i, j) : 1 \leq i, j \leq n\}$ where (i, j) represents the event that the first call to *random*(n) returns i and the second call returns j. Note that the probability of the while-body not executing is $1/n$ corresponding to the n events $(1, 1), (2, 2), \ldots, (n, n)$. Thus the probability that it executes is $(1 - 1/n)$.

At this point the problem is very similar to Problem 2*c* and either solution is a good approach. We outline only one solution here.

Let N denote the number of executions of the while-body. Then the expected running time, $E[T]$, is $c \cdot E[N] + c'$ It follows that

$$E[N] = \sum_{i=1}^{\infty} \Pr[N \geq i] = \sum_{i=1}^{\infty} \left(1 - \frac{1}{n}\right)^i = \frac{1}{1 - \left(1 - \frac{1}{n}\right)} - 1 = n - 1$$

since the probability that the loop-body executes *at least* i times is $\left(1 - \frac{1}{n}\right)^i$ and the resulting sum is a geometric series without the first term. Thus, $E[T] = c \cdot E[N] + c' = c \cdot (n-1) + c' \in \Theta(n)$. □

(g)

```
1  import random
2
3  def f7(xs):
4      s = 0
5      n = len(xs)
6      if n <= 5:
7          return xs[1]
8
9      while random(n) != random(n):
10         s += xs[random(n) - 1]
11
12     if random.randint(1, n) <= 2*n/3:
13         return f7(xs[:n//2])
14
15     return s
```

Proof. First note that a best case is $\Theta(1)$ where we exit after some constant number of *while*-body executions (say 0) and fail to recurse. The worst case is unbounded, as it may run forever.

Let R denote the random variable representing the outcome of random(n). Note that by linearity of expectation

$$E[T(n)] = E[\textit{Problem } \textbf{??}] + E[T(n/2)] \cdot \Pr\left[R \leq \frac{2}{3}n\right].$$

Since R is uniform on $1, \ldots, n$, $\Pr\left[R \leq \frac{2}{3}n\right] = 2/3$. Plugging in,

$$E[T(n)] = \frac{2}{3}E[T(n/2)] + \Theta(n).$$

Solving this recursion, $E[T(n)] \in \Theta(n)$. □

3. Calamari Contest. This year's Calamari Contest[1] involves an infinite bridge made of blacked-out digital displays aligned end-to-end like concrete squares in a sidewalk. A player, in attempting to cross the bridge, advances square-by-square. When a square in a row is stepped on, it either reveals a picture of fried calamari with probability p (and the player who stepped on it is eliminated) or turns solid green, and is allowed to advance to the next square. The player who goes the longest before being eliminated wins.

   (a) A player's lifetime is defined as the number of squares stepped on before calamari is revealed. In expectation, what is a player's lifetime? Equivalently, what is $\mathbb{E}[i]$ where i is the return-value of $walk(p)$ in the following pseudocode?

```
1  from random import uniform
2
3  def is_calamari(p):
4      # assume uniform returns a float
5      # in the range [0, 1) uniformly at random.
6      return uniform(0, 1) < p
7
8  def walk(p):
9      i = 0
10     while not is_calamari(p):
11         i+=1
12     return i
```

Method 1. Okay, so this is very similar to the coin-flipping while-loop we did in class. One way to compute the result, then, is just to average the return value over all possible runs of the above code.

We can think of a generic run as some number of executions of the loop-body before an eventual quit. Let's use the notation $C^{(i)}Q$ to denote a run that *continued* (ran the loop body) i-times and then *quit*.

Therefore, $\Omega = \{C^{(i)}Q : i \in \mathbb{N}_0\}$ covers all possible runs of the code. It's nice because i is also the return value. To figure out $\Pr(C^{(i)}Q)$ note that in a generic run the probability of quitting is p. Therefore, the probability of continuing is $1-p$. Thus, $\Pr(C^{(i)}Q)$, the probability of continuing i-times and then quitting is $(1-p)^i p$.

Method 1a. Okay so now one way to solve this is just by directly computing $\mathbb{E}[i]$. Observe,

$$\mathbb{E}[i] = \sum_{C^{(i)}Q \in \Omega} i \Pr(C^{(i)}Q) = \sum_{i=0}^{\infty} i \Pr(C^{(i)}Q) = \sum_{i=0}^{\infty} i(1-p)^i p = p \sum_{i=0}^{\infty} i(1-p)^i.$$

[1] This is a poorly-disguised squid games reference.

We worked out this last sum in class. (Remember? It was related to the derivative of a geometric series.) Specifically, for $r < 1$, we saw that

$$\sum_{i=0}^{\infty} ir^i = \lim_{n\to\infty} \sum_{i=0}^{n} ir^i = \frac{r}{(1-r)^2}.$$

Therefore,

$$\mathbb{E}[i] = p\sum_{i=0}^{\infty} i(1-p)^i = p\frac{1-p}{p^2} = \frac{1}{p} - 1.$$

Method 1b. This is similar to method 1 except we will convert it into something that's one less than a geometric series. Note that i is integer-valued and so we can use this trick:

$$\mathbb{E}[i] = \sum_{q=0}^{\infty} q \Pr[\{i = q\}] = \sum_{q=1}^{\infty} \Pr[\{i \geq q\}].$$

To apply it, we need to figure out what is the probability that a run contains *at least* q continues, but this is just $(1-p)^q$. (Think about drawing the probability tree. Having at least q continues means that the run terminates somewhere in the subtree following the q continues. The only way to get into that subtree is by having, from the start, exactly q consecutive continues. The probability of ending up in that subtree is, therefore, $(1-p)^q$.) It follows that

$$\mathbb{E}[i] = \sum_{q=1}^{\infty}(1-p)^q = \sum_{q=0}^{\infty}(1-p)^q - 1 = \frac{1}{(1-(1-p))} - 1 = \frac{1}{p} - 1.$$

Method 2. Recursion. This is perhaps the easiest method math-wise, but maybe one of the harder ones to understand.

Let's think about L, the number of times we execute the body of while-loop that quits with probability p. In a generic run, we either have 0 loops with probability p, or we have one loop plus some random number R more loops to go with probability $(1-p)$. So, conditioned on the event we quit right away, Q,

$$\begin{aligned}\mathbb{E}[L] &= \mathbb{E}[L|Q]\Pr[Q] + \mathbb{E}[L|Q^c]\Pr[Q^c] \\ &= 0p + (1 + \mathbb{E}[R])(1-p) \\ &= (1 + \mathbb{E}[R])(1-p).\end{aligned}$$

Okay, so here is the mind-blowing part: Recall that $\mathbb{E}[L|Q^c]$ denotes the expected number of loops conditioned on the fact that we did not quit. Well, since we didn't quit we ran it once for sure followed by R more times, so, by linearity of expecation, we have $1 + \mathbb{E}[R]$, where $\mathbb{E}[R]$ is the expected number of additional loops. But what's $\mathbb{E}[R]$? Well, it's the expected number of while-loop body executions in a

loop that quits with probability p. But wait!? Isn't that $\mathbb{E}[L]$!? Yes! That means that $\mathbb{E}[L] = (1 + \mathbb{E}[L])(1 - p)$. Solving for $\mathbb{E}[L]$ we get

$$\mathbb{E}[L] = \frac{1}{p} - 1.$$

(b) Turns out we don't have the budget for an infinite bridge. What if the length of the bridge is limited to $n \geq 0$ rows (In other words, what is $\mathbb{E}[i]$ where i is the return-value of *walk2*(p, n) in the following pseudocode)?

```
1  def walk2(p, n):
2      i = 0
3      for _ in range(n):
4          if is_calamari(p):
5              break
6          i+=1
7      return i
```

Method

1. Okay, so this is very similar to the previous part except now it is finite. One way to compute the result, then, is just to average the return value over all possible runs of the above code. In analogy with the previous part we will consider the loop to have *continued* if Line 6 is run, otherwise we have *quit.*

Note that Line 6 can be run anywhere from 0 to n times, resulting in values 0 through n as possible return values. Therefore, $\Omega = \{C^{(i)}Q : 0 \leq 1 \leq n\}$ covers all possible runs of the code. To figure out $\Pr(C^{(i)}Q)$ note that in a generic run the probability of quitting is p, except for the n-th run where it is 1. The one hard thing about this problem is accounting for all of the possible off-by-ones. Again, the probability of continuing is $1 - p$, therefore,

$$\Pr(C^{(i)}Q) = \begin{cases} (1-p)^i p & \text{if } 0 \leq i < n \\ (1-p)^i & \text{if } i = n \end{cases}$$

Method 1a. We can now directly compute $\mathbb{E}[i]$,

$$\mathbb{E}[i] = \sum_{C^{(i)}Q \in \Omega} i \Pr(C^{(i)}Q) = \sum_{i=0}^{n} i \Pr(C^{(i)}Q) = \sum_{i=0}^{n-1} i(1-p)^i p + n(1-p)^n.$$

Using the formula we derived in class,

$$\sum_{i=0}^{n} i r^i = r\left[\frac{1 - r^{n+1}}{(1-r)^2} - \frac{(n+1)r^n}{1-r}\right] = r\frac{1 - r^{n+1}}{(1-r)^2} - \frac{(n+1)r^{n+1}}{1-r},$$

we see

$$\begin{aligned}
\mathbb{E}[i] =& p\sum_{i=0}^{n-1} i(1-p)^i + n(1-p)^n \\
=& p\left[(1-p)\frac{1-(1-p)^n}{p^2} - \frac{n(1-p)^n}{p}\right] + n(1-p)^n \\
=& (1-p)\frac{1-(1-p)^n}{p} \\
=& \frac{(1-p)}{p} - \frac{(1-p)^{n+1}}{p} = \frac{1}{p} - 1 - \frac{(1-p)^{n+1}}{p} \\
=& \frac{1-(1-p)^{n+1}}{p} - 1
\end{aligned}$$

As a check, note that we recover the answer to the previous part in the limit that $n \to \infty$.

Method 1b. Note that i is integer-valued and so we can use this trick:

$$\mathbb{E}[i] = \sum_{q=0}^{n} q \Pr[\{i = q\}] = \sum_{q=1}^{n} \Pr[\{i \geq q\}].$$

To apply it, we need to figure out what is the probability that a run contains *at least* q continues, but this is just $(1-p)^q$. (Think about drawing the probability tree.) Note that we don't have to worry about any edge-cases here due to the for-loop limit. (If this is unclear again draw the probability tree.)

It follows that

$$\mathbb{E}[i] = \sum_{q=1}^{n}(1-p)^q = \sum_{q=0}^{n}(1-p)^q - 1 = \frac{1-(1-p)^{n+1}}{(1-(1-p))} - 1 = \frac{1-(1-p)^{n+1}}{p} - 1.$$

Method 2. Recursion. This is like the recursion method for the previous part, except now the expected number of while-loop executions depends on the maximum number of for-loop body executions, n.

Let $L(n)$ denote the expected number of loop executions when there are n possible (remaining) executions. In a generic run, we either have 0 loops with probability p, or we have one loop-body execution plus some random number $R(n)$ more loops to go with probability $(1-p)$. Note: R depends on n this time because we need to know how close we are to the limit. So, conditioned on the event we quit right away, Q,

$$\begin{aligned}
\mathbb{E}[L(n)] &= \mathbb{E}[L(n)|Q]\Pr[Q] + \mathbb{E}[L(n)|Q^c]\Pr[Q^c] \\
&= 0p + (1 + \mathbb{E}[R(n)])(1-p) \\
&= (1 + \mathbb{E}[R(n)])(1-p).
\end{aligned}$$

So what's $\mathbb{E}[R(n)]$? Well its the same as the expected number of loop-body executions if we were running the same for-loop but with one less possible execution, so it's like we are starting fresh but with the limit lowered by one. That is, $\mathbb{E}[R(n)] = \mathbb{E}[L(n-1)]$. Observe,

$$\mathbb{E}[L(n)] = (1 + \mathbb{E}[L(n-1)])(1-p) = (1-p) + (1-p)\mathbb{E}[L(n-1)].$$

This is a recurrence relation, and we can solve it using the base case that $\mathbb{E}[L(0)] = 0$ (when the for-loop is at limit the number of loop-body executions is 0).

After one substitution,

$$\begin{aligned}\mathbb{E}[L(n)] &= (1-p) + (1-p)((1-p) + (1-p)\mathbb{E}[L(n-2)]) \\ &= (1-p) + (1-p)^2 + (1-p)^2\mathbb{E}[L(n-2)]\end{aligned}$$

After two substitutions,

$$\begin{aligned}\mathbb{E}[L(n)] &= (1-p) + (1-p)((1-p) + (1-p)\mathbb{E}[L(n-2)]) \\ &= (1-p) + (1-p)^2 + (1-p)^3 + (1-p)^3\mathbb{E}[L(n-3)]\end{aligned}$$

After k substitutions,

$$\mathbb{E}[L(n)] = (1-p)^{k+1}\mathbb{E}[L(n-(k+1))] + \sum_{i=1}^{k+1}(1-p)^i.$$

This stops when $n-(k+1) = 0$, so $k = n-1$. Plugging in,

$$\mathbb{E}[L(n)] = (1-p)^n\mathbb{E}[L(0)] + \sum_{i=1}^{n}(1-p)^i = \sum_{i=1}^{n}(1-p)^i.$$

This last sum is one less than a geometric series. We have,

$$\mathbb{E}[L(n)] = \sum_{i=1}^{n}(1-p)^i = \sum_{i=0}^{n}(1-p)^i - 1 = \frac{1-(1-p)^{n+1}}{(1-(1-p))} - 1 = \frac{1-(1-p)^{n+1}}{p} - 1.$$

(c) Wait a minute! If you did that right, the game got easier yet the life expectancy went down. This is correct. Explain why.

Indeed, life expectancy went down, for the infinite case we had $\mathbb{E}_\infty = 1/p - 1$, and for the finite case we had,

$$\mathbb{E}_{\text{finite}} = \frac{1-(1-p)^{n+1}}{p} - 1 = \left(\frac{1}{p} - 1\right) - \frac{(1-p)^{n+1}}{p} = \mathbb{E}_\infty - \frac{(1-p)^{n+1}}{p}.$$

Yet even though life expectancy went down, the finite games are obviously easier since, for instance, there's a chance you survive to the end of the game. But it's

not like the squares are more likely to break, so why is it that when we make the game shorter the player is expected to survive fewer squares?

Recall we are computing an average over all possible walks of the bridge! In the infinite case there are some very long walks in the average (though they come with low probability). These long walks are skewing the average. In the finite game any player that would be lucky enough to survive for a while past the end of a finite game of length n instead survives for only length n in the finite game.

Perhaps a better way to define life expectancy, then, is as the expected fraction of players that win the game.