# CSE 312: Foundations of Computing II — Fall 2020

## Problem Set 4 (due Wednesday, October 28 at 11:59pm)

**Directions**:
***Answers**: For each problem, remember you must briefly explain/justify how you obtained your answer, as correct answers without an explanation will receive **no credit**. Moreover, in the event of an incorrect answer, we can still try to give you partial credit based on the explanation you provide. It is fine for your answers to include summations, products, factorials, exponentials, or combinations; you dont need to calculate those all out to get a single numeric answer, for instance $26^7$ or $26!/7!$ or $26 \cdot \binom{26}{7}$.*

*Your solutions need to be concise and clear. We will take off points for lack of clarity or for excess verbosity. Please see section worksheet solutions (posted on the course website) to gauge the level of detail we are expecting.*

*Please clearly indicate your final answer, in such a way as to distinguish it from the rest of your explanation.*

***Groups**: This pset must be done **solo**. You will submit on Gradescope as a single student.*
*Individuals are still encouraged to discuss problem-solving strategies with other classmates as well as the course staff, but each person must write up their own solutions, without looking at any writeup other than their own.*

***Submission**: You must upload a **pdf** of your written solutions to Gradescope under "PSet 4 [Written]". Problem 9 is a coding problem, so under "Pset 4 [Coding]" you will be uploading a .py file called cse312_pset4_bloom.py. If you do the extra credit, that will be submitted separately under "Pset4 [Extra]". (Instructions as to how to upload your solutions to Gradescope are on the course web page.) The use of latex is highly recommended.*

*Note that if you want to hand-write your solutions, you'll need to scan them. We will take off points for hand-written solutions that are difficult to read due to poor handwriting and neatness.)*

***Please cite any collaboration on each problem.***

## 1. More Heads than Tails? (9 points)

Let $X$ represent the difference between the number of heads and the number of tails obtained when a fair coin is tossed independently $100$ times. What are the possible values of $X$? What is the probability that $X = 4$? (Note: $X = H - T$, not $X = |H - T|$. That is, if you have more tails than heads, then the difference is negative, not positive.)

## 2. Busy 312 Students (10 points)

CSE 312 students sometimes delay laundry for a few days (to the chagrin of their roommates).
A ***busy*** 312 student must complete 3 problem sets before doing laundry. Each problem set requires 1 day with probability 2/3 and 2 days with probability 1/3. (The time it takes to complete different problem sets is independent.) Let $B$ be the number of days a busy student delays laundry. What is the probability mass function for $B$?

## 3. CDF to PMF (10 points)

Suppose that $X$ is a discrete random variable that takes integer values from 1 to 100 (both inclusive), and has cumulative distribution function (CDF)

$$F_X(x) = Pr(X \leq x) = \frac{\lfloor x \rfloor \lfloor x + 1 \rfloor}{10100} \qquad 1 \leq x \leq 100$$

and

$$F_X(x) = 0 \quad \text{for } x < 1 \quad \text{and} \quad F_X(x) = 1 \quad \text{for } x > 100$$

(Thus, for example, $F(1) = 1 \cdot 2/10100$ and $F(2) = 2 \cdot 3/10100$ and so on.)

Find the probability mass function (pmf) for $X$. In other words, provide a formula for $p_X(x)$ that is correct for any integer $x$ in $\{1, 2, \ldots, 100\}$.

## 4. Fair Game (10 points)

Consider the following game, defined by a parameter $k$: Roll a fair, 6-sided die three times independently. If you roll a six

- no times, then you lose 1 dollar.
- exactly once, you win 1 dollar.
- exactly twice, then you win 2 dollars.
- all three times, then you win $k$ dollars

For what value of $k$ is this game fair? (The game is fair if your expected payoff is 0.)

## 5. Packet Failures (21 points)

Consider three different models for sending $n$ packets over the Internet:

1. each packet takes a different path. Each path fails independently with probability p;
2. all packets take the exact same path which fails with probability p. Thus, either all the packets get through or none get through;
3. half the packets take one path, and half take the other (assume n even), and each of the two paths fails independently with probability p.

Let $X_i$ be the number of packets lost in case i, for i = 1, 2, 3. Write down the probability mass function, the expectation of $X_i$ and the variance of $X_i$ for $i = 1, 2, 3$.

## 6. Doggies and Koalas (10 points)

Suppose that $4n$ animals are partitioned into pairs at random, with each partition being equally likely. If the set consists of $n$ doggies and $3n$ koalas, what is the expected number of doggie-koala couples?

## 7. Friends (10 points)

Consider a group of $n$ people where each pair of people is friends independently with probability $1/2$. What is the expected number of triples of people that are all friends, i.e. triples $A, B, C$ such that $A$ is friends with $B$ and $B$ is friends with $C$ and $A$ is friends with $C$. Use linearity of expectation and carefully define indicator random variables (0/1 valued random variables) and justify your work.

## 8. More Coin Flipping (10 points)

A coin with probability $p$ of coming up heads is tossed independently $n$ times. What is the expected number of maximal "runs", where a "run" is a maximal sequence of consecutive flips that are the same? For example, the sequence HHHTTHTHHH has 5 runs, the first three H, the following two T, and so on. Use linearity of expectation and carefully define indicator rvs and justify your work.

## 9. Bloom Filter [Coding] (10 points)

Google Chrome has a huge database of malicious URLs, but it takes a long time to do a database lookup (think of this as a typical Set). They want to have a quick check in the web browser itself, so a space-efficient data structure must be used. A **bloom filter** is a **probabilistic data structure** which only supports the following two operations:

I. add(x): Add an element $x$ to the structure.

II. contains(x): Check if an element $x$ is in the structure. If either returns "definitely not in the set" or "could be in the set".

It does **not** support the following two operations:

I. Delete an element from the structure.

II. Give a collection of elements that are in the structure.

The idea is that we can check our bloom filter if a URL is in the set. The bloom filter is always correct in saying a URL definitely isn't in the set, but may have false positives (it may say a URL is in the set when it isn't). Only in these rare cases does Chrome have to perform an expensive database lookup to know for sure.

Suppose we have $k$ **bit arrays** $t_1, \ldots, t_k$ each of length $m$ (all entries are 0 or 1), so the total space required is only $km$ bits or $km/8$ bytes (as a byte is 8 bits). Suppose the universe of URL's is the set $\mathcal{U}$ (think of this as all strings with less than 100 characters), and we have $k$ ***independent and uniform*** hash functions $h_1, \ldots, h_k : \mathcal{U} \to \{0, 1, \ldots, m-1\}$. That is, for an element $x$ and hash function $h_i$, pretend $h_i(x)$ is a **discrete** $Unif(0, m-1)$ random variable. Suppose we implement the add and contains function as follows:

Bloom Filter Operations

**function** INITIALIZE(k,m)
    **for** $i = 1, \ldots, k$: **do**
        $t_i$ = new bit array of m 0's
**function** ADD(x)
    **for** $i = 1, \ldots, k$: **do**
        $t_i[h_i(x)] = 1$
**function** CONTAINS(x)
    **return** $t_1[h_1(x)] == 1 \wedge t_2[h_2(x)] == 1 \wedge \cdots \wedge t_k[h_k(x)] == 1$

Refer to the notes from Chapter 9.4 in the textbook for more detail on bloom filters. Implement the functions add and contains in the BloomFilter class of cse312_pset4_bloom.py.

## 10. Extra Credit Problem (10 points)

The names of 100 people are placed into 100 closed bags, one name per bag, and the bags are lined up on a table in a room. One by one the people are led into the room; each may look in at most 50 bags, but must leave the room exactly as they found it. From the moment a person is led into the room, that person is not allowed any further communication with the others.

The people have a chance to plot their strategy in advance (and come up with a coordinated strategy if they so desire) and they really want to, because *if every single person finds their own name among the 50 bags they look in, they will each win a million dollars.* Otherwise, nobody wins anything.

What can they possibly do to win the money? For example, if they each look into 50 random bags, then the chance any one of them finds their name is $1/2$. But that means that the chance that they **all** find their names is $2^{-100}, miniscule$!

Amazingly, we will now develop a strategy that guarantees that they win with probability more than 0.3. The strategy is the following. The players first agree amongst themselves on a random labelling of the bags with their names. Then, when a person walks into the room, he first looks at his own bag (the one that has his label in the random labelling). If he doesn't find his name inside, he looks into the bag belonging to the name he

just found, and then into the bag belonging to the name he found in the second bag, etc. until he either finds his own name, or has opened 50 bags.

Let me run a quick example assuming 5 people that are each allowed to look into at most 3 bags.

Say the random labelling $\pi$ the people agree on for the bags on the table is (Carol, Alice, Darin, Elise, Bob) in order and suppose that what's inside the corresponding bags in their order on the table are the names (Alice, Bob, Elise, Darin, Carol)

Then when Alice comes in, she will go to the bag labelled with Alice's name which is bag 2, see Bob's name inside it, then go to the bag labelled with Bob's name (bag 5), see Carol's name inside it and then go to the bag labelled with Carol's name and see her own name inside it. Yippee, she has found her own name without looking into more than 3 bags. In fact, in this example, using the above strategy, everyone will find their name without looking into more than 3 bags.

On the other hand, suppose that the random labelling $\pi'$ of the bags on the table in order was (Darin, Bob, Alice, Carol, Elise) (but still inside the bags the names are in the same order: Alice, Bob, Elise, Darin, Carol) Then, using the above strategy, Bob will find his own name, but none of the rest of them will find their names after looking in 3 bags, and so no money for any of them.

Think of the names *inside* the bags in the order they are on the table as $1, 2, \ldots n$, and the random labelling (random permutation) $\pi$ the people agree on ahead of time as $\pi_1, \pi_2, \ldots, \pi_n$. Then the strategy for the person whose name is $i$ is to go to the bag j such that $\pi_j = i$, then go to the bag $k$ such that $\pi_k = j$ and then go to the bag $\ell$ such that $\pi_\ell = k$, until you either find your name or have taken too many steps. Such a sequence is called a *cycle* in a permutation $\pi$. It is a sequence of indices $i_1, \ldots, i_k$ such that $\pi_{i_1} = i_k$, $\pi_{i_2} = i_1$, $\pi_{i_3} = i_2$ $\ldots$, $\pi_{i_k} = i_{k-1}$.[1]

So in the first example above, if we think of (Alice, Bob, Elise, Darin, Carol) as (1, 2, 3, 4, 5) then the random permutation $\pi$ is (5, 1, 4, 3, 2) (i.e. $\pi_1 = 5$, $\pi_2 = 1$ and so on). Thus, in this permutation (2, 5, 1) is a cycle (because $\pi_2 = 1$, $\pi_5 = 2$ and $\pi_1 = 5$). Similarly, (3, 4) is a cycle. and in the second example, the permutation $\pi'$, (2) is a cycle and (3, 5, 4, 1) is a cycle. (You might want to convince yourself that every permutation is the union of disjoint cycles).

Now the questions:

- (No credit for this part.) Convince yourself that if the random permutation that people pick has no cycle of length more than 50, then they will win the money.
- What is the probability that a random permutation of the numbers 1 to $2n$ does not contain any cycle of length greater than $n$?
- Using the approximation

$$\sum_{1 \le i \le n} \frac{1}{i} \approx \ln(n),$$

evaluate the probability that the 100 people each win a million dollars (i.e., *all* of them find their own name). (FYI: $\sum_{1 \le i \le n} \frac{1}{i}$ is called the $n$-th harmonic number and is denoted by $H_n$).

**We will require concise and very clear answers to this question in order to earn any credit.**

[1] This is a non-standard definition of a cycle.