Problem Set 4

Due: Wednesday, October 25, by 11:59pm

Instructions

Solutions format, collaboration policy, and late policy. See PSet 1 for further details. The same requirements and policies still apply. Also follow the typesetting instructions from the prior PSets. This Ed post about explanations might also help.

Solutions submission. You must submit your solution via Gradescope. In particular:

- For the solutions to Task 1-6, and the written part of Task 7, submit under "PSet 4 [Written]" a *single* PDF file containing the solution to all tasks in the homework. Each numbered task should be solved on its own page (or pages). Follow the prompt on Gradescope to link tasks to your pages. Do not write your name on the individual pages – Gradescope will handle that.
- For the programming part (part of Task 7), submit your code under "PSet 4 [Coding]" as a file called bloom.py.
- If you do the extra credit (Task 8), you can submit your work under "PSet 4 [Extra Credit]"

Task 1 – Fair Carnival Game [10 pts]

Consider the following game, defined by a parameter k: Throw 3 darts independently at a wall with 20 balloons of 3 different colors (10 purple, 3 gold and 7 white), and the wall resets each time (if you hit a balloon, it will be replaced for the next throw). You will miss completely (not hit a balloon) with probability 0.6 for each throw, and if you hit a balloon, you are equally likely to hit any particular balloon. If you hit

- no balloons, then you lose 3 dollars.
- exactly one balloon, then you lose 1 dollar.
- two purple balloons and a gold balloon, you win 2 dollars.
- three gold balloons, you win k dollars.
- any other outcome, you win 0 dollars.

For what value of k rounded to the nearest cent is this game fair? (The game is fair if your expected payoff is 0.)

Task 2 – Lefties and Righties [12 pts]

Suppose that there are 150 students taking CSE 312 and they are partitioned into pairs at random to work together on the problem set, with each partition being equally likely. If the class has 90 people that are right-handed and 60 people that are left-handed, what is the expected number of pairs that are different-handed, that is pairs where one person in the pair is right-handed and the other person in the pair is left-handed?

Task 3 – Triple fun [12 pts]

This problems concerns two kinds of graphs. The *complete graph on n vertices* has a set V of vertices with $|V| = n$ and a set E of *edges*, one edge for each pair of elements of V. It is denoted by K_n. A *tournament on n vertices* is a directed graph that consists of a set V of vertices with $|V| = n$ and a set E of directed edges, one for each pair of vertices $u, v \in V$ that has exactly one of the two directed edges (u, v) or (v, u).

a) Suppose that we independently flip **unfair** coins, one per edge of K_n, and color each edge *red* if its coin is heads and *blue* if the coin is tails. The coin is heads with probability 0.6 and tails with probability 0.4 What is the expected number of *triangles* (triples of vertices) that have their edges colored the same (all blue or all red).

b) Suppose that we choose a tournament on n vertices with $V = \{1, 2, \ldots, n\}$ randomly by independently flipping **unfair** coins, one per pair of vertices $\{u, v\} \in V$ with $u < v$ and include the edge (u, v) (in other words is u points to v) if the coin is heads and include the edge (v, u) (in other words v points to u) if the coin is tails. The unfair coin is the same as in part (a) where it is heads with probability 0.6 and tails with probability 0.4. What is the expected number of directed cycles of length three in this randomly chosen tournament? (Recall that in a directed cycle one can follow the directed edges and end up where you started.)

Task 4 – Independence [14 pts]

We are given two independent random variables, M and K, both taking integer values between 0 and 80. Further, assume that you know that the distribution of K is uniform; i.e., $\mathbb{P}(K = i) = 1/80$ for all $i \in \{0, 1, 2, \ldots, 79\}$. *You do not know anything else about the distribution of M.* Finally, we define a new random variable C as

$$C = (M + K) \mod 80 .$$

a) For all fixed values $m, c \in \{0, 1, \ldots, 79\}$, use what you know about modular arithmetic and the independence of random variables M and K to write the probability $\mathbb{P}(C = c, M = m)$ in terms of $\mathbb{P}(M = m)$.

b) Use part **a)** and the Law of Total Probability to compute $\mathbb{P}(C = c)$ as a function of $c \in \{0, 1, , \ldots, 79\}$.

c) Use the definition of independence together with your solutions to parts **a)** and **b)** to show that M and C are (mutually) independent.

Task 5 – Some Really Odd Dice! [14 pts]

You are playing a game that uses a fair 8-sided die whose faces are numbered by the odd numbers, 1, 3, 5, ..., 15. The value of a roll is the number showing on the top of the die when it comes to rest. Give all answers as simplified fractions.

a) Let X be the value of one roll of the die. Compute $\mathbb{E}[X]$ and $\mathrm{Var}(X)$.

b) Let Y be the sum of the values of 5 independent rolls of the die. Compute $\mathbb{E}[Y]$ and $\mathrm{Var}(Y)$. Use independence, and state precisely where in your computation you are using it.

c) Let Z be the **average** of the values of n independent rolls of the die. Compute $\mathbb{E}[Z]$ and $\mathrm{Var}(Z)$. Use independence, and state precisely where in your computation you are using it.

Task 6 – Investment Strategies [18 pts]

Consider three different investment strategies.

1. (6 points) You buy one share in each of n different stocks. Each share of a different stock "pays off" independently with probability p.

2. (6 points) You buy n shares of the same stock. A share of that stock pays off with probability p. Since all the shares are of the same stock, either all of them pay off (probability p) or none of them pay off.

3. (6 points) You buy $n/2$ shares of stock A and $n/2$ shares of stock B. A share of stock A and a share of stock B each independently pays off with probability p. All shares of the same stock either all pay off (probability p) or none of them pay off (probability $1-p$).

Let X_i be the number of shares that pay off in strategy i, for $i = 1, 2, 3$. Write down the probability mass function, the expectation of X_i and the variance of X_i for each of $i = 1, 2, 3$.

Task 7 – Bloom filters [Coding+Written] [10+7 pts]

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:

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

- delete an element from the structure.
- return an element that is in the structure.

The idea is that we can check our Bloom filter to see 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 that 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 that 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 that 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** $\text{Unif}[0, m-1]$ random variable. Suppose that we implement the add and contains function as follows:

Algorithm 1 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 slides from Lecture 10 and Section 9.4 of the textbook for more details on Bloom filters.

a) Implement the functions add and contains in the BloomFilter class of bloom.py.

To solve this task, we have set up a corresponding edstem lesson here. Press the Mark button above the terminal to run the unit tests we have written for you. Passing these unit tests is not enough. We have written a number of different tests for the Gradescope autograder. Your score on Gradescope will be your actual score - you have unlimited attempts to submit.

b) Let's compare this approach to using a typical Set data structure. Google wants to store 1 million URLs, with each URL taking (on average) 23 bytes.

- How much space (in MB, 1 MB $=$ 1 million bytes) is required if we store all the elements in a set?
- How much space (in MB) is required if we store all the elements in a Bloom filter with $k = 10$ hash functions and $m = 800,000$ buckets? Recall that 1 byte $=$ 8 bits.

c) Let's analyze the time improvement as well. Let's say an average Chrome user attempts to visit 36,500 URLs in a year, only 1,000 of which are actually malicious. Suppose it takes half a second for Chrome to make a call to the database (the Set), and only 1 millisecond for Chrome to check containment in the Bloom filter. Suppose the false positive rate on the Bloom filter is 4%; that is, if a website is not malicious, the Bloom filter will incorrectly report it as malicious with probability 0.04.

- What is the time (in seconds) taken if we only use the database?
- What is the *expected* time taken (in seconds) to check all 36,500 strings if we used the **B**loom filter $+$ database combination described earlier?

Task 8 – Extra Credit Problem

You are shown two envelopes and told the following facts:

- Each envelope has some number of dollars in it, but you don't know how many.
- The amount in the first envelope is different from the amount in the second.
- Although you don't know exactly how much money is in each envelope, you are told that it is an integer number of dollars that is at least 1 and at most 100.
- You are told that you can pick an envelope, look inside, and then you will be given a one-time option to switch envelopes (without looking inside the new envelope). You will then be allowed to keep the money in envelope you end up with.

Your strategy is the following:

1. You pick an envelope uniformly at random.
2. You open it and count the amount of money inside. Say the result is x.
3. You then select an integer y between 1 and 100 uniformly at random.
4. If $y > x$, you switch envelopes, otherwise you stay with the envelope you picked in step (a)

Show that you have a better than 50-50 chance of taking home the envelope with the larger amount of money in it. More specifically, suppose the two envelopes have i and j dollars in them respectively, where $i < j$. Calculate the probability that you take home the envelope with the larger amount of money.