Asmt 1: Hash Functions and PAC Algorithms

Turn in (**a pdf**) through GradeScope by 1pm:
Wednesday, August 27

Overview

In this assignment you will experiment with random variation over discrete events.

Beyond experimenting with concepts covered in the lecture, the point of this assignment is guidance on how to work with randomization within algorithms (and data). There are no longer clear unit tests you can write, and it is not always easy to verify by eye if your code is working correctly. Making it more challenging, if you do not fix a random seed, then each time you run the code it should produce different results. So what should you do? There are two main parts:

1. run the algorithm multiple times (with different random seeds), and look at the distribution of outputs (not just one output run)

2. understand the mathematical/probabilistic analysis of the phenomena, and compare that to the distributional output.

You will do both in this assignment.

It is recommended that you use LaTeX for your assignment reports. If the math and presentation is not clear, you may lose points if your assignment is difficult to read or hard to follow. There are other ways to do this well, but latex is what I recommend. With overleaf, the barrier to start is now, I believe, quite reasonable. Find a sample form in this directory: https://www.overleaf.com/read/gvhwtfkfvvrk#39dae7

Click here for an example template specifically created for this assignment.

1 Birthday Paradox (35 points)

Consider a domain of size $n = 5000$.

A: (5 points) Generate random numbers in the domain $[n]$ until two have the same value. How many random trials did this take? We will use k to represent this value.

B: (10 points) Repeat the experiment $m = 300$ times, and record for each time how many random trials this took. Plot this data as a *cumulative density plot* where the x-axis records the number of trials required k, and the y-axis records the fraction of experiments that succeeded (a collision) after k trials. The plot should show a curve that starts at a y value of 0, and increases as k increases, and eventually reaches a y value of 1.

Below is an example of how to create a CDF plot in Python:

```
import matplotlib.pyplot as plt
def cdf(data):
    """Calculates the empirical CDF of a given dataset."""
    # sorted the data
    sorted_data = sorted(data)
    # Calculate the length of the data
    n = len(sorted_data)
    # Create a list of CDF values
    cdf_values = []
    for i, x in enumerate(sorted_data):
        cdf_values.append((i+1) / n)
```

```
        return sorted_data, cdf_values
    # Example usage
    data = [1, 3, 2, 5, 4]
    x, y = cdf(data)
    plt.step(x, y, where="post")
    plt.xlabel("Data")
    plt.ylabel("CDF")
    plt.title("Empirical CDF")
    plt.show()
```

Listing 1: Question 2D

C: (10 points) Empirically estimate the expected number of k random trials in order to have a collision. That is, add up all values k, and divide by m.

D: (10 points)

i. **(2 points)** Describe how you implemented this experiment and how long it took for $n = 5000$ and $m = 300$ trials.

ii. **(8 points)** Show a plot of the run time as you gradually increase the parameters n and m. (For at least 3 fixed values of m between 300 and 10,000, plot the time as a function of n.) You should be able to reach values of $n = 1{,}000{,}000$ and $m = 10{,}000$. *(Hint: use the timings of the smaller sizes to estimate the runtime of the largest n, m setting, and make sure it will finished before the assignment is due.)*

2 Coupon Collectors (35 points)

Consider a domain $[n]$ of size $n = 300$.

A: (5 points) Generate random numbers in the domain $[n]$ until every value $i \in [n]$ has had one random number equal to i. How many random trials did this take? We will use k to represent this value.

B: (10 points) Repeat step **A** for $m = 400$ times, and for each repetition record the value k of how many random trials we required to collect all values $i \in [n]$. Make a cumulative density plot as in **1.B**.

C: (10 points) Use the above results to calculate the empirical expected value of k.

D: (10 points)

i. **(2 points)** Describe how you implemented this experiment and how long it took for $n = 300$ and $m = 400$ trials.

ii. **(8 points)** Show a plot of the run time as you gradually increase the parameters n and m. (For at least 3 fixed values of m between 400 and 5,000, plot the time as a function of n.) You should be able to reach $n = 20{,}000$ and $m = 5{,}000$. *(Hint: use the timings of the smaller sizes to estimate the runtime of the largest n, m setting, and make sure it will finished before the assignment is due.)*

3 Comparing Experiments to Analysis (30 points)

With randomized algorithms, an important tool for understanding if they are working correctly is to leverage mathematical analysis. If the output is in the range suggested by the mathematics, then you have more confidence everything is working as intended. Apply that here (as described below) for your experiments in the first two questions.

A: (15 points) Calculate analytically (using formulas from the notes in **L2** or elsewhere, but then cite your source) the number of random trials needed so there is a collision with probability at least 0.5 when the domain size is $n = 5000$. There are a few formulas stated with varying degree of accuracy, you may use any of these – the more precise the formula, the more sure you may be that your experimental part is verified, or is not (and thus you need to fix something).
[Show your work, including describing which formula you used.]

How does this compare to your results from **1.C**?

B: (15 points) Calculate analytically (using formulas from the notes in **L2** or elsewhere, but then cite your source) the expected number of random trials before all elements are witnessed in a domain of size $n = 300$? Again, there are a few formulas you may use – the more precise, the more confidence you might have in your experimental part.
[Show your work, including describing which formula you used.]

How does this compare to your results from **2.C**?

4 BONUS : Mod k Coupons (4 points)

Extend the coupon collector analysis to when you draw coupons from two bins, and each draw must alternate between bin 1 and bin 2. However, bin 1 only contains odd values in $[n]$ and bin 2 only even values in $[n]$.

Then solve two variants.

- How does the analysis change if you can choose which bin to draw from each draw?
- What happens if there are k bins, and the jth bin (out of k) only contains numbers from $[n]$ that are $(j \mod k)$? This is in the setting, where on the ith draw where $i = j \mod k$, then you must draw from bin j.

[You must show work. If you use AI tools, please document you prompts – you may not be eligible for full credit however with answers that draw mostly on AI responses.]