CS170 Discussion Section 13: *4/26*

Commitment Scheme

A casino would like to allow people to play the game of roulette by calling in over the phone. In a game of roulette, a ball is spun around a wheel and randomly falls into one of 38 holes. The player wins if they guess which hole the ball will fall in (there are other winning conditions, we pick just this one). If the casino is not honest, they can lie to the player and tell them they always lose. Show how a commitment scheme can help.

The casino rolls the ball around the wheel. It lands in a pocket, call this pocket x. The casino then sends a commitment of x to the player.

Now, the player can safely send the casino a guess of the number they think has been selected. The casino can now reveal the commitment to the user along with the value x. The player can verify that the commitment was done correctly, and will know if they have won or not.

Mathematically, the casino begins by picking a prime p and generator g. The casino spins the wheel and notes the value x the ball lands in. Then, the casino chooses a random number $0 \leq r < \frac{p}{38}$ and sends $c = g^{x+38r}$ to the player. Because computing the discrete log is hard, the player cannot feasibly cheat and determine the value of x before guessing.

After the player picks his selected choice y and sends it to the casino, the casino reveals the values of x and r to the player. The player can now check that c was computed correctly, and if $x = y$ then the player has won. The casino can not change the value x it committed to, because otherwise the players verification would not be correct.

Notice that the process here is inverted from the actual version of roulette. In a real game, the player will place his bet on a board *before* the wheel is spun. By doing this, the player **commits** to his choice. Then, when the ball lands in a pocket, the player can see this and can not change where his bet is placed.

Streaming: Reservoir Sampling

(a) You have a length n stream of numbers. You want to randomly select one number fairly (all numbers should have equal probability of being selected). However, you don't know what n is beforehand, and you can only parse through the stream once! Design a $O(\log n)$ space algorithm that will randomly choose one element fairly.

- Keep the first item in memory
- When the i^{th} item arrives ($i > 1$),
  1) With probability $\frac{1}{i}$, discard the old item, and keep the new one in memory
  2) With probability $1 - \frac{1}{i}$, keep the old item and ignore the new one

This will give each item a probability $\frac{1}{n}$ of being chosen. This can be seen intuitively for $n = 1, 2$. And this can be shown in general using induction (if this property holds for size n, if we were to increase the size to $n+1$, the property will hold by definition of what to do on the $(n+1)^{th}$ element).

(b) Same as part (a), except now you want to randomly select k numbers, not just one.

We can take a similar approach as the solution for part (a). However, we keep track of k items instead of 1, and start with the first k items of the stream. For the i^{th} item, we will keep it with probability $\frac{k}{i}$. If we decide to keep an item, randomly choose one of the k spots to replace. Then this will fairly sample k elements from the stream. This can be shown using induction.

Chernitto's Educated Guess

Chernitto gets one try to guess a very important value. All Chernitto has at hand is an algorithm that yields a ϵ-accurate answer with probability 0.6, and an inaccurate answer with probability 0.4. How can Chernitto give a ϵ-accurate answer with a probability of at least 0.999?

Chernitto should run his algorithm k times, and then return the median. The higher k is, the higher chance of getting a median that is within ϵ of the true value. We can use the Chernoff bound to give us a lower bound for k.

Let's attempt to calculate the probability that our median of k answers is *not* ϵ-accurate. Assuming we have an odd k, to output a final ϵ-accurate answer, we just need half or more of our iterations to give ϵ-accurate outputs. This is analagous to flipping a biased, heads-favored coin a total of k flips, and hoping we get more heads than tails.

Let X_i be a Bernoulli random variable that is 1 if the i^{th}output is ϵ-accurate, and 0 if not. We know these probabilities to be 0.6 and 0.4, respectively. We attempt to count the number of times that the answer is ϵ-accurate:

Let $X = X_1 + X_2 + ... + X_n$. Then the exact probability is:

$$\Pr\left[X > \frac{n}{2}\right] = \sum_{\lfloor \frac{n}{2} \rfloor + 1}^{n} \binom{n}{i} p^i (1-p)^{n-i}$$

Using the Chernoff inequality, this can be bounded:

$$\Pr\left[X > \frac{n}{2}\right] \geq 1 - e^{-\frac{n}{2p}(p-\frac{1}{2})^2}$$

$$\Pr\left[X \leq \frac{n}{2}\right] = 1 - \Pr\left[X > \frac{n}{2}\right] = e^{-\frac{n}{2p}(p-\frac{1}{2})^2}$$

In our case, p is 0.6, so we want this quantity to be smaller than 0.001:

$$\Pr\left[X \leq \frac{n}{2}\right] = e^{-\frac{n}{2\times 0.6}(0.6-\frac{1}{2})^2}$$

Then we need to solve the following:

$$e^{-\frac{n}{1.2}0.01} = 0.001$$

$$n = -\ln(0.001)\frac{1.2}{0.01}$$

Solving this yields $n = 828.9$. So if we let $k = 829$, then we will certainly output a ϵ-accurate with probability greater than 0.999.

Note: There are other variants of the Chernoff Bound that you can work with, and they could give you varying results. All such results will still be valid. Most variants derive from the Moment-Generating-Function form, and give some leeway so that the formula looks nice.

Project Check-In

If logistically reasonable, you are encouraged to check in with your TA on your project phase II progress so far.