CS 473 ✧ Fall 2022

Homework 5

Due Tuesday, October 11 2022 at 9pm

Unless a problem specifically states otherwise, you may assume a function RANDOM that takes a positive integer k as input and returns an integer chosen uniformly and independently at random from $\{1, 2, \ldots, k\}$ in $O(1)$ time. For example, to flip a fair coin, you could call RANDOM(2).

---

1. Consider a random walk on a path with vertices numbered $1, 2, \ldots, n$ from left to right. At each step, we flip a coin to decide which direction to walk, moving one step left or one step right with equal probability. The random walk ends when we fall off one end of the path, either by moving left from vertex 1 or by moving right from vertex n.

   (a) Prove that if we start at vertex 1, the probability that the walk ends by falling off the *right* end of the path is exactly $1/(n+1)$.

   (b) Prove that if we start at vertex k, the probability that the walk ends by falling off the *right* end of the path is exactly $k/(n+1)$.

   (c) Prove that if we start at vertex 1, the expected number of steps before the random walk ends is exactly n.

   (d) What is the *exact* expected length of the random walk if we start at vertex k, as a function of n and k? Prove your result is correct. (For partial credit, give a tight Θ-bound for the case $k = (n+1)/2$, assuming n is odd.)

   [Hint: Trust the recursion fairy. Yes, "see part (b)" is worth full credit for part (a), but only if your solution to part (b) is correct. Same for parts (c) and (d).]

2. ***Tabulation hashing*** uses tables of random numbers to compute hash values. Suppose $|\mathcal{U}| = 2^w \times 2^w$ and $m = 2^\ell$, so the items being hashed are pairs of w-bit strings (or $2w$-bit strings broken in half) and hash values are ℓ-bit strings.

   Let $A[0..2^w - 1]$ and $B[0..2^w - 1]$ be arrays of independent random ℓ-bit strings, and define the hash function $h_{A,B} : \mathcal{U} \to [m]$ by setting

   $$h_{A,B}(x, y) := A[x] \oplus B[y]$$

   where $\oplus$ denotes bit-wise exclusive-or. Let $\mathcal{H}$ denote the set of all possible functions $h_{A,B}$. Filling the arrays A and B with independent random bits is equivalent to choosing a hash function $h_{A,B} \in \mathcal{H}$ uniformly at random.

   (a) Prove that $\mathcal{H}$ is 2-uniform.

   (b) Prove that $\mathcal{H}$ is 3-uniform. *[Hint: Solve part (a) first.]*

   (c) Prove that $\mathcal{H}$ is ***not*** 4-uniform.

   [Hint: Yes, "see part (b)" is worth full credit for (a), if your part (b) solution is correct.]

3. Suppose we are given a coin that may or may not be biased, and we would like to compute an accurate *estimate* of the probability of heads. Specifically, if the actual unknown probability of heads is p, we would like to compute an estimate $\tilde{p}$ such that

$$\Pr[|\tilde{p} - p| > \varepsilon] < \delta$$

where ε is a given ***accuracy*** or ***error*** parameter, and δ is a given ***confidence*** parameter.

The following algorithm is a natural first attempt; here FLIP( ) returns the result of an independent flip of the unknown coin.

```
MEANESTIMATE(ε):
  count ← 0
  for i ← 1 to N
      if FLIP( ) = HEADS
          count ← count + 1
  return count/N
```

(a) Let $\tilde{p}$ denote the estimate returned by MEANESTIMATE(ε). Prove that $\mathrm{E}[\tilde{p}] = p$.

(b) Prove that if we set $N = \lceil \alpha/\varepsilon^2 \rceil$ for some appropriate constant α, then we have $\Pr[|\tilde{p} - p| > \varepsilon] < 1/4$. *[Hint: Use Chebyshev's inequality.]*

(c) We can increase the previous estimator's confidence by running it multiple times, independently, and returning the *median* of the resulting estimates.

```
MEDIANOFMEANSESTIMATE(δ, ε):
  for j ← 1 to K
      estimate[j] ← MEANESTIMATE(ε)
  return MEDIAN(estimate[1..K])
```

Let p^* denote the estimate returned by MEDIANOFMEANSESTIMATE(δ, ε). Prove that if we set $N = \lceil \alpha/\varepsilon^2 \rceil$ (inside MEANESTIMATE) and $K = \lceil \beta \ln(1/\delta) \rceil$, for some appropriate constants α and β, then $\Pr[|p^* - p| > \varepsilon] < \delta$. *[Hint: Use Chernoff bounds.]*