CS331: Algorithms and Complexity
Homework V

Trung Dang Ryan Park Kevin Tian

Due date: November 20, 2024, end of day (11:59 PM), uploaded to Canvas.

Late policy: 15% off if submitted late, and 15% off for every further 24 hours before submission.

Please list all collaborators on the first page of your solutions.

When runtimes are unspecified, slower runtimes than the intended solution receive partial credit.

1 Problem 1

You have access to a fair coin, $\mathsf{Coin}()$, which outputs 0 and 1 with probability $\frac{1}{2}$ each. You want to implement an algorithm $\mathsf{BiasedCoin}(p)$, which takes as input a parameter $p \in (0, 1)$, and outputs 0 and 1 with probability $1 - p$ and p respectively. This seems challenging to do for an arbitrary value p, but one day, the following pseudocode appears to you in a dream.

Algorithm 1: $\mathsf{BiasedCoin}(p)$

```
Input: $p \in (0, 1)$
$(f, i) \leftarrow (\mathsf{Coin}(), 1)$
while $f \neq 1$ do
    $(f, i) \leftarrow (\mathsf{Coin}(), i + 1)$
end
return $p_i$, the $i^{\text{th}}$ digit of $p$ written out as a decimal $0.p_1p_2p_3\ldots$ in binary, so $p = \sum_{j \geq 1} p_j 2^{-j}$
```

(i) **(5 points)** What is $\mathbb{E}[i]$ when the algorithm terminates?

(ii) **(15 points)** Prove that the implementation of $\mathsf{BiasedCoin}(p)$ is correct.

2 Problem 2

You have access to a mysterious subroutine $\mathsf{MysteryCoin}()$, which for some unknown $p^\star \in (0, 1)$, outputs 0 and 1 with probability $1 - p^\star$ and $p^\star$ respectively. Your goal is to estimate $p^\star$.

(i) **(10 points)** Let X be the number of ones observed upon calling $\mathsf{MysteryCoin}()$ n times, for some $n \in \mathbb{N}$. What are $\mathbb{E}[X]$ and $\mathrm{Var}[X]$?

(ii) **(10 points)** For a target failure probability $\delta \in (0, 1)$, give an algorithm that outputs p satisfying $|p - p^\star| \leq 0.1$ with probability $\geq 1 - \delta$. You may use $O(\log(\frac{1}{\delta}))$ calls to $\mathsf{MysteryCoin}()$.

3 Problem 3

There is a universe Ω of objects that you want to store efficiently. You have access to a hash table with m buckets, as well as a function $\mathsf{Hash}(\omega)$, that for each object $\omega \in \Omega$, independently outputs a uniformly random number in $[m]$. You want to hash n distinct objects $\omega_1, \omega_2, \ldots, \omega_n \in \Omega$. The hope is that not too many of the objects end up in any one bucket (on average, we expect $\approx \frac{n}{m}$ per bucket). More specifically, we want to bound the *maximum load* of any bucket:

$$L := \max_{i \in [m]} L_i, \text{ where } L_i := |\{\omega_j \mid j \in [n], \mathsf{Hash}(\omega_j) = i\}| \text{ for all } i \in [m].$$

In other words, L_i is the number of objects that are randomly hashed to bucket $i \in [m]$.

(i) **(10 points)** Suppose $m = 10$. Give a constant n such that with probability ≥ 0.9,

$$L \leq \frac{n}{5}.$$

(ii) **(10 points)** Suppose $m \in \mathbb{N}$ is arbitrary. Give a value of n (which may depend on m) such that with probability ≥ 0.9,

$$L \leq \frac{2n}{m}.$$

Any correct answer suffices for both parts, as long as you provide a proof of correctness.

4 Problem 4

You are interested in generating a geometric random variable $X \sim \mathsf{Geom}(p)$, where $p = \frac{1}{n}$ for some $n \in \mathbb{N}$ (recall this means $\Pr[X = i] = p(1-p)^{i-1}$ for all $i \in \mathbb{N}$). The usual way of doing this is to call $\mathsf{BiasedCoin}(p)$ over and over again, until you see a 1. However, if n is large, then you would rather not have to flip $\approx n$ coins. You instead have the idea to sample (Y, Z), where

$$Y \leftarrow \left\lfloor \frac{X-1}{n} \right\rfloor + 1, \; Z \leftarrow 1 + ((X-1) \pmod{n}).$$

This is spurred by the observation that $X = n(Y-1) + Z$. For example, if $n = 20$ and $X = 75$, then this split is given by $X = 20 \cdot (4-1) + 15$, so $Z = 15$ is capturing the "remainder" and $Y = 4$ is capturing the "bucket." In this example, $X \in [1, 20]$ has $Y = 1$, $X \in [21, 40]$ has $Y = 2$, etc.

One day, the following pseudocode appears to you in a dream.

Algorithm 2: $\mathsf{FastGeom}(n)$

```
1  Input: $n \in \mathbb{N}$ for $n \geq 2$
2  $p \leftarrow \frac{1}{n}$
3  $q \leftarrow 1 - (1-p)^n$
4  $(f, Y) \leftarrow (\mathsf{BiasedCoin}(q), 1)$
5  while $f \neq 1$ do
6      $(f, Y) \leftarrow (\mathsf{BiasedCoin}(q), Y + 1)$
7  end
8  $Z \leftarrow \mathsf{Uniform}(n)$
9  $f \leftarrow \mathsf{BiasedCoin}((1-p)^{Z-1})$
10 while $f \neq 1$ do
11     $Z \leftarrow \mathsf{Uniform}(n)$
12     $f \leftarrow \mathsf{BiasedCoin}((1-p)^{Z-1})$
13 end
14 return $n(Y-1) + Z$
```

On Lines 8 and 11, $\mathsf{Uniform}(n)$ returns a uniformly random number in $[n]$.

(i) **(8 points)** Prove that Lines 4 to 7 correctly sample Y (Lemma 5, Part I may be helpful).

(ii) **(8 points)** Prove that Lines 8 to 13 correctly sample Z.

(iii) **(4 points)** Prove that $\mathsf{FastGeom}(n)$ uses $O(1)$ calls to $\mathsf{BiasedCoin}$ and $\mathsf{Uniform}$ in expectation.

5 Problem 5

(20 points) Complete the assignment at this link. This link is only accessible on your UT email.