CSC265 F23: Assignment 1

Due: November 3, by midnight

Guidelines: (read fully!!)

- Your assignment solution must be submitted as a *typed* PDF document. Scanned handwritten solutions, solutions in any other format, or unreadable solutions will **not** be accepted or marked. You are encouraged to learn the LaTeX typesetting system and use it to type your solution. See the course website for LaTeX resources.
- Your submission should be no more than 7 pages long, in a single-column US Letter or A4 page format, using at least 10 pt font and 1 inch margins.
- To submit this assignment, use the MarkUs system, at URL `https://markus.teach.cs.toronto.edu/2023-09`
- This is a *group assignment.* This means that you can work on this assignment with *at most one other* student. You are *strongly encouraged* to work with a partner. Both partners in the group should work on and arrive at the solution together. Both partners receive the same mark on this assignment.
- Work on all problems together. For each problem, one of you should write the solution, and one should proof-read and revise it. The first page of your submission must list the *name*, *student ID*, and *UTOR email address* of both group members. It should also list, for each problem, which group member wrote the problem, and which group member proof-read and revised it.
- You **may not** consult any other resources except: your partner; your class notes; your textbook and assigned readings. *Consulting any other resource, or collaborating with students other than your group partner, is a violation of the academic integrity policy!*
- You may use any data structure, algorithm, or theorem previously studied in class, or in one of the prerequisites of this course, by just referring to it, and without describing it. This includes any data structure, algorithm, or theorem we covered in lecture, in a tutorial, or in any of the assigned readings. Be sure to give a *precise reference* for the data structure/algorithm/result you are using.
- Unless stated otherwise, you should justify all your answers using rigorous arguments. Your solution will be marked based both on its completeness and correctness, and also on the clarity and precision of your explanation.

Question 1. (12 marks)

Your goal in this question is to design a data structure which allows you to very quickly sample from a discrete distribution. Suppose you are given non-negative rational numbers $p_1, \dots, p_n \in [0,1]$ such that $\sum_{i=1}^n p_i = 1$. They define a probability distribution on $[n] = \{1, \dots, n\}$ by $\mathbb{P}(i) = p_i$. In the following subquestions you will design a data structure which takes linear space, can be constructed in linear time, and allows sampling from this distribution in constant time.

Part a. (8 marks)

Suppose that $p_1, \dots, p_n$ are as above. Your goal in this subquestion is to split the probabilities p_i across $2n$ buckets arranged in n pairs, so that the total mass of each pair is $1/n$, and each bucket contains some portion of the probability mass of some $i \in [n]$. More formally, describe an algorithm which computes two $n \times 2$ two-dimensional arrays A and B such that:

- The entries of A are non-negative rational numbers, and the entries of B are integers in $\{0\} \cup [n]$.
- If $B[j,k] = 0$ for some $j \in [n]$ and $k \in \{1,2\}$, then $A[j,k] = 0$, too.
- For any $j \in [n]$ we have $A[j,1] + A[j,2] = \frac{1}{n}$.
- For any $i \in [n]$ we have

$$\sum_{j:B[j,1]=i} A[j,1] + \sum_{j:B[j,2]=i} A[j,2] = p_i. \tag{1}$$

With respect to the informal description above, the values in A denote how much probability mass is stored in a bucket, while the values in B denote which p_i this mass belongs to. For example, $A[3,2] = 1/9$, $B[3,2] = 10$ means that the second bucket in the third pair of buckets stores mass $1/9$, coming from p_{10}.1 $B[j,k] = 0$ is allowed so that we can have empty buckets.

As an example, say we are given $p_1 = 3/4, p_2 = 1/8, p_3 = 1/16, p_4 = 1/16$. Then one possible output of your algorithm is

$$A = \begin{bmatrix} 1/16 & 3/16 \\ 1/16 & 3/16 \\ 1/8 & 1/8 \\ 1/4 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 3 & 1 \\ 4 & 1 \\ 1 & 2 \\ 1 & 0 \end{bmatrix}.$$

On input $p_1, \dots, p_n$, your algorithm should compute A and B in time $O(n)$. (Partial credit is given for $O(n \log n)$.) Justify why your algorithm correctly computes A and B and works in the required time.

Part b. (4 marks)

Suppose you are given two functions:

- Bernoulli(p) which takes a rational number $p \in [0,1]$ and returns a random bit B such that $\mathbb{P}(B = 1) = p$ and $\mathbb{P}(B = 0) = 1 - p$;
- RandInt(n) which takes a positive integer n and returns a uniformly random number in $[n]$.

Assume that both functions work in constant time. Without using any other randomness except by calling the above two functions, show how to use the arrays A and B from the previous subquestion to sample in worst-case constant time a random integer $I \in [n]$ such that $\mathbb{P}(I = i) = p_i$. Justify your answer.

Question 2. (12 marks) This question is about a method to compress an array of integers into a much smaller array using hashing, so that we can still approximate the entries of the original array.

Part a. (4 marks)

Suppose that $A[0\,..\,n-1]$ is an array of non-negative integers, and consider the following algorithm which compresses A into a smaller array $S[0\,..\,k-1]$. In this algorithm, $\mathcal{H}$ is a universal family of hash functions that map $\{0,\dots,n-1\}$ to $\{0,\dots,k-1\}$.

```
Compress(A)
1  Sample $h \in \mathcal{H}$ uniformly at random.
2  Initialize $S[0\,..\,k-1]$ s.t. $S[i] = 0$ for all $i = 0\,..\,k-1$.
3  for $i = 0\,..\,n-1$
4          $S[h(i)] = S[h(i)] + A[i]$
5  return $(S, h)$
```

We can then query S using the following simple procedure, which takes an array $S[0\,..\,k-1]$, a hash function $h \in \mathcal{H}$, and an index $i \in \{0,\dots,n-1\}$.

```
Query(S, h, i)
1  return $S[h(i)]$
```

Show that, for any $i \in \{0,\dots,n-1\}$, the value $\tilde{a}_i$ returned by Query(S,h,i) satisfies $\tilde{a}_i \geq A[i]$, and, moreover,

$$\mathbb{E}[\tilde{a}_i - A[i]] \leq \varepsilon(k)\sum_{j\neq i} A[j],$$

where the expected value is taken over the random choice of h, and $\varepsilon(k)$ is some monotonically decreasing function of k. In your answer, give a precise formula for $\varepsilon(k)$, and make it as small as you can. Justify your answer.

Part b. (8 marks)

Let $\delta \in (0,1)$ be a parameter. Describe a modification of the Compress and Query procedures above so that Compress(A) outputs a data structure consisting of $O(k\log(1/\delta))$ integers rather than k integers, and Query takes the data structure output by Compress(A) and an integer $i \in \{0,\dots,n-1\}$, and outputs $\tilde{a}_i$ such that $\tilde{a}_i \geq A[i]$, and

$$\mathbb{P}\Big(\tilde{a}_i - A[i] \geq 2\varepsilon(k)\sum_{j\neq i} A[j]\Big) \leq \delta.$$

Here $\varepsilon(k)$ is the same as in the previous subquestion. Moreover, Compress should take time $O(n\log(1/\delta))$ and Query should take time $O(\log(1/\delta))$.

You can use the following fact from probability theory.

Markov's Inequality. For any random variable Z such that $Z \geq 0$, and any $t > 0$, we have

$$\mathbb{P}(Z \geq t\mathbb{E}[Z]) \leq \frac{1}{t}.$$

Give pseudocode for your modifications of Compress and Query, and justify your answer.

Hint: Consider using multiple hash functions.