CS 473: Algorithms, Fall 2016
HW 5 (due Tuesday, October 11th at 8pm)

This homework contains three problems. **Read the instructions for submitting homework on the course webpage**.

Collaboration Policy: For this home work, each student can work in a group with upto three members. Only one solution for each group needs to be submitted. Follow the submission instructions carefully.

1. In lecture we discussed the Karp-Rabin randomized algorithm for pattern matching. The power of randomization is seen by considering the *two-dimensional* pattern matching problem. The input consists of a $n \times n$ binary matrix T and a $m \times m$ binary matrix P. Our goal is to check if P occurs as a (contiguous) submatrix of T. Describe an algorithm that runs in $O(n^2)$ time assuming that arithmetic operation in $O(\log n)$-bit integers can be performed in constant time. This can be done via a modification of the Karp-Rabin algorithm. To achieve this, you will have to apply some ingenuity in figuring out how to update the fingerprint in only constant time for most positions in the array. *Hint:* we can view an $m \times m$ matrix as an m^2-bit integer. Rather than computing its fingerprint directly, compute instead a fingerprint for each row first, and maintain these fingerprints as you move around.

2. **Reservoir sampling** is a method for choosing an item uniformly at random from an arbitrarily long stream of data whose length is not known apriori.

```
UniformSample:
s ← null
m ← 0
While (stream is not done)
   m ← m + 1
   x_m is current item
   Toss a biased coin that is heads with probability 1/m
   If (coin turns up heads)
         s ← x_m

Output s as the sample
```

   (a) **Not to submit but useful to solve:** Prove that the above algorithm outputs a uniformly random sample from the stream.

   (b) To obtain k samples *with* replacement, the procedure for $k = 1$ can be done in parallel with independent randomness. Now we consider obtaining k samples from the stream *without* replacement. The output will be stored in an array of S of size k.

```
SAMPLE-WITHOUT-REPLACEMENT(k):

S[1..k] ← null
m ← 0
While (stream is not done)
   m ← m + 1
   x_m is current item
   If (m ≤ k)
         S[m] ← x_m
   Else
         r ← uniform random number in range [1..m]
         If (r ≤ k)
               S[r] ← x_m

Output S
```

Prove that the preceding algorithm generates a uniform sample of size k without replacement from the stream of size m. Assume that $m \geq k$.

3. An important and fundamental problem in streaming is the following. Suppose the stream consists of m elements each of which is an integer between 1 and n. Here we assume that n is known but the stream can be arbitrarily long. We would like to estimate the number of *distinct* numbers in the stream. For instance if the stream is $1, 1, 10, 2, 2, 2, 1, 1, 10$ the answer shoule be 3. Of course we can do this by maintaining n counters but this would require a huge amount of space. Efficient randomized algorithms are known that output a $(1+\epsilon)$-approximate estimate for the number of distinct numbers in the stream by usingly $O(\log n/\epsilon^2)$ space. Here we describe a simple seed idea for this problem. Let the stream of numbers be $a_1, a_2, \ldots, a_m$. We want to estimate d, the number of distinct numbers in the stream.

   To estimate d to within a constant factor[1] consider a balls and bins experiment of throwing d identical balls into n bins. Let Z be the index of the smallest non-empty bin. Suppose $d \in [2^i, 2^{i+1})$. Prove that $\Pr[Z \in [n/2^{i+2}, n/2^{i-1})] \geq c$ for some fixed constant c. Thus, n/Z gives a constant factor estimate for d with probability at least c.

   In order to make this into an algorithm we use a random hash function $h : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ and keep track of $Z = \min\{h(a_1), h(a_2), \ldots, h(a_m)\}$ which is only one number to store. Hashing collapses all copies of the same number into one "ball" and and also mimics the process of throwing a ball uniformly into a bin. Of course $h(a_1), h(a_2), \ldots, h(a_m)$ don't behave independently as in the balls and bins experiment unless we choose h from the set of all hash functions. However, one can show that even if h is chosen from a 2-universal family the analysis goes through. More on this can be found in the following lecture notes `https://courses.engr.illinois.edu/cs598csc/fa2014/Lectures/lecture_2.pdf`.

[1] d' is a constant factor estimate for d if there are fixed constants $c_1, c_2 \geq 1$ such that $d/c_1 \leq d' \leq c_2 d$.

The remaining problems are for self study. Do *NOT* submit for grading.

- Jeff's Spring 16 Homework 4 and 5 available at links below. `https://courses.engr.illinois.edu/cs473/sp2016/hw/hw4.pdf`, `https://courses.engr.illinois.edu/cs473/sp2016/hw/hw5.pdf`
- Consider the CountMin sketch to estimate the frequencies of the items in a stream. Suppose $\epsilon = 0.2$ and $\delta = 0.5$. Give an example of an input stream σ such that the probability is very high that for at least one of the items $j \in \sigma$, the estimate of its frequency is much larger than its actual frequency. More precisely, give an example such that (for m large enough) the probability that there is an item j with the $\tilde{F}[j] - F[j] > m/2$ is at least 0.99. Here $\tilde{F}[j]$ is the estimated frequencey of j from the sketch and $F[j]$ is the true frequency.
- There is another very useful sketch called the Count sketch. You can read about it, if you are interested. `https://courses.engr.illinois.edu/cs598csc/fa2014/Lectures/lecture_6.pdf`