CS 473: Algorithms, Spring 2021
HW 5 (due Wednesday, March 17th 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. *Tabulated hashing* uses tables of random numbers to compute hash values. Suppose $|\mathcal{U}| = 2^w \times 2^w$ and $m = 2^l$, so that the items being hashed are pairs (x, y) where x and y are w-bit strings (or $2w$-bit strings broken in half), and hash values are l-bit strings.

   Let $A[0 \cdots 2^w - 1]$ and $B[0 \cdots 2^w - 1]$ be arrays of l-bit strings (A and B can be though of as $2^w \times l$ dimensional array of bits). Define the has function $h_{A,B} : \mathcal{U} \to [m]$ by setting

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

   where $\oplus$ denotes bit-wise exclusive-or. Let $\mathcal{H}'$ denote the set of all possible functions $h_{A,B}$. Note that sampling an $h_{A,B} \in \mathcal{H}'$ uniformly at random is equivalent to setting every bit of the arrays A and B to 0 or 1 uniformly at random.

   For an integer $k > 0$, we say that a family of hash functions $\mathcal{H}$ mapping $\mathcal{U}$ to $\{0, 1, \cdots, (m-1)\}$ is *k-uniform* if for any sequence of k disjoint keys and any sequence of k hash values, the probability that each key maps to the corresponding hash value is $\frac{1}{m^k}$

   $$\Pr_{h \sim \mathcal{H}} \left[\bigwedge_{j=1}^{k} h(x_j, y_j) = i_j \right] = \frac{1}{m^k} \text{ for all disjoint } \{(x_i, y_i)\}_{i \in [k]} \in \mathcal{U}, \text{ and all } i_1, \cdots i_k \in \{0, \ldots, (m-1)\}$$

   In the above, $h \sim \mathcal{H}$ means function h is picked uniformly at random from family $\mathcal{H}$. (For more details on k-uniform family of hash functions, see Jeff's notes (page 3): `https://courses.engr.illinois.edu/cs473/sp2016/notes/12-hashing.pdf`.)

   (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.

   Yes, "see part (b)" is worth full credit for part (a), but only if your solution to part (b) is correct.

2. 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 an *arbitrary* $n \times n$ binary matrix T and an *arbitrary* $m \times m$ binary matrix

P, where $m < n$. 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.]

3. **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 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 $m \geq k$.