Recap of this week’s lectures:

- Hashing: Universal hashing, and constructions.
- Perfect hashing: dictionary lookup in constant worst-case time.
- The Data Streaming model
- Heavy hitters, both without and with deletions.

**Hashing:** A *universal* hash family $\mathcal{H}$ from $U$ to $[m] := \{0, 1, \ldots, m - 1\}$ is a set of hash functions $\mathcal{H} = \{h_1, h_2, \ldots, h_k\}$ each mapping $U$ to $[m]$, such that for any $a \neq b \in U$, when you pick a random function from $\mathcal{H}$,

$$\Pr[h(a) = h(b)] \leq \frac{1}{m}.$$ 

Also, a $\ell$-*universal* hash family $\mathcal{H}$ from $U$ to $[m] := \{0, 1, \ldots, m - 1\}$ is a set of hash functions $\mathcal{H} = \{h_1, h_2, \ldots, h_k\}$ each mapping $U$ to $[m]$, such that for any distinct $a_1, \ldots, a_\ell \in U$, and for any $\alpha_1, \ldots, \alpha_\ell \in [m]$, when you pick a random function from $\mathcal{H}$,

$$\Pr[h(a_1) = \alpha_1 \text{ and } \ldots \text{ and } h(a_\ell) = \alpha_\ell] = \frac{1}{m^\ell}.$$ 

1. Show that a 2-universal hash family is a universal hash family.
2. Show that a $k$-universal hash family is a $\ell$-universal hash family for any $\ell \leq k$.

3. Is this hash family from $U = \{a, b\}$ to $\{0, 1\}$ (i.e., $m = 2$) universal? 1-universal? 2-universal?

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_1$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$h_2$</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$h_2$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$h_3$</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$h_4$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
5. Suppose that an adversary knows the hash family \( \mathcal{H} \) and controls the keys we hash, and the adversary wants to force a collision. In this problem part, suppose that \( \mathcal{H} \) is universal. The following scenario takes place: we choose a hash function \( h \) randomly from \( \mathcal{H} \), keeping it secret from the adversary, and then the adversary chooses a key \( x \) and learns the value \( h(x) \). Can the adversary now force a collision? In other words, can it find a \( y \neq x \) such that \( h(x) = h(y) \) with probability greater than \( \frac{1}{m} \)? If so, write down a particular universal hash family in the same format as in part (b), and describe how an adversary can force a collision in this scenario. If not, prove that the adversary cannot force a collision with probability greater than \( \frac{1}{m} \).

\[ \begin{array}{c|ccc} & x & y & z \\ \hline h_1 & 0 & 0 & 1 \\ h_2 & 1 & 0 & 1 \end{array} \]

6. Answer the question from \#5, but supposing that \( \mathcal{H} \) is 2-universal, not just universal.
Streaming.

**Sampling:** Given a number $k$, you want to maintain a random sample of size $k$ from the stream. I.e., for each $n \geq k$, the set you have at time $n$ should be a random subset of the prefix $a[1:n]$, each of the $\binom{n}{k}$ subsets of size $k$ from this prefix should be equally likely.

1. For $k = 1$, show that the algorithm: pick the first element. When faced with the $n^{th}$ element, with prob. $1/n$ discard the element in your hand and pick the new element, and with prob. $1 - 1/n$ keep the element in hand.

2. Give an algorithm for general $k$. (What would you do when faced with the $n^{th}$ element? With what probability should you pick this element? Which element should you drop?)
**Missing Numbers:** Suppose I give you a stream of \( n - 1 \) elements, which contains all the numbers from 1 thru \( n \) except one of them. (The numbers do not appear in sorted order.) Clearly you can figure out the missing number by storing all \( n - 1 \) numbers and looking for the missing number. How can you output the missing number with only \( O(\log n) \) space? What if there are two missing numbers: can you again use only \( O(\log n) \) space?

---

**Solution:**
For one missing number, you can store the sum of all numbers seen so far. Then finally subtract that from \( \frac{n(n+1)}{2} \) to get the missing number. For two, you can store, e.g., the sum, and the sum of their squares. Then you’ll know \( a + b \) and \( a^2 + b^2 \), and can solve for the answer.

You could also have stored the sum and product of the numbers seen, but that requires more space. The product of the numbers could be as large as \( \Omega(n!) \) which requires \( \Omega(n \log n) \) bits to store.
Min-Hashing.

In min-hashing we created an estimator with “one-sided error”: our estimate was always an overestimate. I.e., for the target value $v$ we created a random variable $X$ such that $\Pr[X \geq v] = 1$. Suppose $E[X] = \mu$.

1. Show that $\Pr[X \geq 2\mu] \leq \frac{1}{2}$. (“The probability of one estimate being too large is at most 50%.”)

2. Use this to show that if we take $k$ independent copies $X_1, X_2, \ldots, X_k$ of the r.v. $X$, then $\Pr[\min_{i=1}^{k}(X_i) \geq 2\mu] \leq 2^{-k}$.

3. Show that $k = \lg(1/\delta)$ gives $2^{-k} = \delta$. (If we want error probability $2^{-100}$, take the minimum of 100 independent estimates.)