Note: Your TA may not get to all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. The discussion worksheet is also a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

1 Comparing Traffic

Alice and Bob want to estimate the following: *Among all vehicles that pass through Hearst Ave and Bancroft Ave on a given day, what fraction of them go through both the streets?*

Alice will observe traffic on Hearst Ave all day, while Bob will observe traffic on Bancroft Ave. The two of them will take notes on small pieces of paper. At the end of the day, they would like to compare their notes and estimate the fraction of vehicles common between the two streets.

Formally, Alice observes a stream $A = \{a_1, \dots, a_n\}$ and Bob observes a stream $B = \{b_1, \dots, b_m\}$ where each $a_i, b_j \in \{1, \dots, N\}$. The goal is to estimate $\frac{|A \cap B|}{|A \cup B|}$.

1. At first, Alice and Bob decide to each store a small random sample of vehicles $S \subset A$ and $T \subset B$ (using say reservoir sampling). At the end of the day, Alice and Bob would compute $\frac{|S \cap T|}{|S \cup T|}$. Briefly argue why this algorithm fails.

   Solution: For concreteness, consider even the extreme case where $A = B$ (the streams contain the same elements possibly in different order). Even in this extreme case, Alice and Bob will end up sampling mostly disjoint subsets S and T. Note that each of them is only picking a small sample out of a large stream, so there is little chance that their samples overlap. So the estimate $\frac{|S \cap T|}{|S \cup T|}$ will likely be equal to 0, although $\frac{|A \cap B|}{|A \cup B|} = 1$.

2. Using the hashing idea from the streaming algorithm for computing distinct elements, devise an algorithm that Alice and Bob can use.

   Solution: Alice and Bob agree on a random hash function $h : \{1, \dots, N\} \to [0, 1]$. Alice and Bob both compute the stream element with the smallest hash value in their stream. i.e.,

   $$\mathit{smallest}(A) = \text{ element with smallest hash value in } A$$

   $$\mathit{smallest}(B) = \text{ element with smallest hash value in } B$$

   Claim:

   $$\Pr[\mathit{smallest}(A) = \mathit{smallest}(B)] = \frac{|A \cap B|}{|A \cup B|}$$

   Let

   $$\mathit{smallest}(A \cup B) = \text{ element with smallest hash value in} A \cup B$$

   Notice that if $\mathit{smallest}(A \cup B) \in A \cap B$, then we will have $\mathit{smallest}(A) = \mathit{smallest}(B) = \mathit{smallest}(A \cup B)$. Conversely, if $\mathit{smallest}(A) = \mathit{smallest}(B)$ then indeed $\mathit{smallest}(A \cup B) \in A \cap B$. Therefore we have that,

   $$\Pr[\mathit{smallest}(A) = \mathit{smallest}(B)] = \Pr[\mathit{smallest}(A \cup B) \in A \cap B]$$

   However, for a random hash function h, every element in $A \cup B$ is equally likely to be the smallest. Hence the probability that

   $$\Pr[\mathit{smallest}(A \cup B) \in A \cap B] = \frac{|A \cap B|}{|A \cup B|}$$

   .

The above argument shows that by picking a hash function $h : \{1, \ldots, N\} \to [0, 1]$, Alice and Bob can compute a bit b

$$b = \begin{cases} 1 & \text{if } \mathit{smallest}(A) = \mathit{smallest}(B) \\ 0 & \text{otherwise} \end{cases}$$

such that $\mathbb{E}[b] = \frac{|A \cap B|}{|A \cup B|}$.

To estimate $\frac{|A \cap B|}{|A \cup B|}$, Alice and Bob pick several independent hash functions $h_1, \ldots, h_t$ and average the corresponding bits, $\frac{1}{t}\sum_{i=1}^{t} b_i$.

2 Estimating Votes

Suppose we have a stream of votes of form (Id, Yes) or (Id, No) which has person's Id and whether they voted Yes/No. We would like to estimate the fraction of Yes votes. Unfortunately, many people have voted multiple times. People who voted multiple times voted for the same option each time. Provide an algorithm for estimating the fraction of "Yes" votes and state its space complexity. You should count the vote given by each Id only once.
Solution:

Main Idea. We want to pick t distinct elements from the stream without duplicates. By running the *distinct elements* algorithm, we can sample a single element with uniform probability, irregardless of duplicates of the corresponding vote. Hence we run the distinct elements algorithm in parallel t times, using a distinct hash function each time to sample t distinct elements. For each run, output the element with the minimum hash value. Output as estimate $\widehat{Z}$ the number of "Yes" votes in the t votes, divided by t (i.e. the fraction of "Yes" votes in the t sampled elements). Set $t = \frac{1}{2\epsilon^2} \log\left(\frac{2}{\delta}\right)$.

Analysis. Since each sample is an independent sample from the set of *distinct* voters, their average is an unbiased estimate for the true fraction of Yes votes. Using a Hoeffding bound, we get that for $t = \frac{1}{2\epsilon^2} \log\left(\frac{2}{\delta}\right)$, $|\widehat{Z} - \mu| < \epsilon$ with probability at least $1 - \delta$, where μ is the true underlying fraction of Yes votes.