CS 124 Homework 5: Spring 2024

Collaborators: Collin Fan, Sophie Zhu, Mohammad Khan, Gianfranco Randazzo, Dries Rooryk, Davin Jeong

No. of late days used on previous psets: 6
No. of late days used after including this pset: 6

Homework is due Wednesday Apr 3 at 11:59pm ET. Please remember to select pages when you submit on Gradescope. Each problem with incorrectly selected pages will lose 5 points.

Try to make your answers as clear and concise as possible; style may count in your grades. Assignments must be submitted in pdf format on Gradescope. If you do assignments by hand, you will need to scan your papers to turn them in.

Collaboration Policy: You may collaborate on this (and all problem sets) only with other students currently enrolled in the class, and of course you may talk to the Teaching Staff or use Ed. You may also consult the recommended books for the course and course notes linked from the timetable. You may not use generative AI or large language models, or search the web for solutions, or post the questions on chat forums. Furthermore, you must follow the "one-hour rule" on collaboration. You may not write anything that you will submit within one hour of collaborating with other students or using notes from such sources. That is, whatever you submit must first have been in your head alone, or notes produced by you alone, for an hour. Subject to that, you can collaborate with other students, e.g. in brainstorming and thinking through approaches to problem-solving.

For all homework problems where you are asked to give an algorithm, you must prove the correctness of your algorithm and establish the best upper bound that you can give for the running time. Generally better running times will get better credit; generally exponential-time algorithms (unless specifically asked for) will receive no or little credit. You should always write a clear informal description of your algorithm in English. You may also write pseudocode if you feel your informal explanation requires more precision and detail, but keep in mind pseudocode does NOT substitute for an explanation. Answers that consist solely of pseudocode will receive little or no credit. Again, try to make your answers clear and concise.

Problems

1. Let D be some probability distribution over the elements $\{1,\ldots,m\}$, and for any $i \in \{1,\ldots,m\}$, let p_i denote the probability that a random sample from D is equal to i.

   Suppose we draw n independent samples $(x_1, x_2, \ldots, x_n)$ from D.

   (a) **(10 points)** Given any two indices $1 \le i < j \le n$, we say that the i-th sample and the j-th sample *collide* if they are equal. (For example, in the sequence of samples $(1,1,1,5,1,2,4,5)$, there is a collision with $i = 1$ and $j = 2$, and one with $i = 2$ and $j = 3$ and one with $i = 1$ and $j = 3$ and so on for a total of 7 collisions.)
   Write down a formula in terms of $p_1,\ldots,p_m$ for the expected number of collisions in the sequence $(x_1, x_2, \ldots, x_n)$. Determine a choice of $p_1,\ldots,p_m$ that maximizes this quantity. Similarly, determine a choice of $p_1,\ldots,p_m$ that minimizes this quantity.

Let random variable $I_{i,j}$ be defined such that (i, j) is a pair of indices where $i < j$. We define $I_{i,j}$ to be:

$$I_{i,j} = \begin{cases} 1 \text{ if } x_i = x_j \\ 0 \text{ otherwise} \end{cases}$$

Thus, by the linearity of expectation, the expected number of collisions in a sequence $(x_1, x_2, ..., x_n)$ is the sum of the expected values of indicators $I_{i,j}$ for all possible pairs (i, j) where $i < j$. Each indicator intuitively has an expected value of $E(I_{i,j}) = 0+1*p_i*p_i$ since $p_i = p_j$ follows from $x_i = x_j$. Therefore, since we have $\frac{n(n-1)}{2} = \binom{n}{2}$ possible pairs of (i, j) where $i < j$, the expected number of collisions is

$$\binom{n}{2} \sum_{i=1}^{m} p_i^2$$

Based on the formula, we can have at most $\binom{n}{2}$ collisions in the sequence because $p_i \leq 1$ for all $1 \leq i \leq m$ and $p_1 + p_2 + ... + p_m = 1$. We can also verify this intuitively: the maximum number of collisions is achieved when all randomly sampled elements are equal which has $\binom{n}{2}$ collisions. Thus, consider when $p_1 = 1$. Then, by the definition of a probability, every random sample x_i from D will equal the element corresponding with p_1. It follows that $x_1 = x_2 = ... = x_n$ and there are $\binom{n}{2}$ collisions. Therefore, a choice of $p_1, ..., p_m$ that maximizes this quantity is where $p_1 = 1, p_2 = ...p_m = 0$.

To find a choice of $p_1, ..., p_m$ that minimizes the expected number of collisions, we can apply the Cauchy Schwarz Inequality. Since $\binom{n}{2}$ is a constant factor which is not affected by $p_1, ..., p_m$, we instead focus on minimizing $\sum_{i=1}^{m} p_i^2$.

Proved by Augustin-Louis Cauchy, Hermann Schwarz, and Viktor Bunyakovsky, the Cauchy Schwarz inequality dictates that $(a_1^2 + a_2^2 + ... + a_n^2)(b_1^2 + b_2^2 + ... + b_n^2) \geq (a_1 b_1 + a_2 b_2 + ... + a_n b_n)^2$.

Thus, applying the Cauchy Schawarz Inequality, we have:

$$(p_1^2 + p_2^2 + ... + p_m^2)(1^2 * m) \geq (p_1 * 1 + p_2 * 1 + ... + p_m * 1)^2$$

$$(p_1^2 + p_2^2 + ... + p_m^2)(m) \geq (p_1 + p_2 + ... + p_m)^2$$

By the definition of a probability distribution, we know $p_1 + ... + p_m = 1$. Therefore,

$$(p_1^2 + p_2^2 + ... + p_m^2)(m) \geq 1$$

$$\sum_{i=1}^{m} p_i^2 \geq \frac{1}{m}$$

To achieve the lower bound where $\sum_{i=1}^{m} p_i^2 = \frac{1}{m}$ which minimizes our expected number of collisions formula, we can set each $p_i = \frac{1}{m}$ because $\sum_{i=1}^{m} \frac{1}{m^2} = m * \frac{1}{m^2} = \frac{1}{m}$. As such, a choice of $p_1, ..., p_m$ which minimizes the expected number of collisions is $p_1 = p_2 = ... = p_m = \frac{1}{m}$.

(b) **(10 points)** Suppose that D is the (discrete) uniform distribution and let A be the event that there is at least one index i for which $x_i = 1$ (i.e., A does *not* happen if and only if for all i, $x_i \neq 1$). Let B be the event that there zero indices i for which $x_i = 2$. Prove or disprove that A and B are independent.

Proof. To disprove that A and B are independent events, we will prove that $P(A) \neq P(A|B)$. Recall that A is the probability that at least one index i has $x_i = 1$. This is equivalent calculating the complement, which is 1 minus the probability that there exist zero indices i where $x_i = 1$. Since D is the discrete uniform distribution, we can express the probability of that no indices i exist such that $x_i = 1$ as $(\frac{m-1}{m})^n$. Thus, the probability of event A occuring is:

$$P(A) = 1 - (\frac{m-1}{m})^n$$

Then, to calculate the probability of $P(A|B)$, we will calculate the probability of $P(A)$ given that event B occurs as well. Thus, by the definition of event B, we know that there exist no indices i such that $x_i = 2$. Therefore, when we calculate the probability of event A given event B, we again calculate the complement. However, instead of calculating the probability that there exist zero indices i where $x_i = 1$, we must also factor in the assumption that there are zero indices i where $x_i = 2$. Thus,

$$P(A|B) = 1 - (\frac{m-2}{m-1})^n$$

because each x_i has $m-2$ valid choices ($x_i \neq 1$ and $x_i \neq 2$) out of $m-1$ total options ($x_1 \neq 2$).

Therefore, since $1 - (\frac{m-2}{m-1})^n \neq 1 - (\frac{m-1}{m})^n$, it follows that $P(A|B) \neq P(A)$. This implies that event A and event B are not independent for any $m > 1$. □

2. A distribution over hash functions $H : \{1,\ldots,n\} \to \{1,\ldots,n\}$ is said to be 2-*wise independent* if for any $x_1, x_2, v_1, v_2 \in \{1,\ldots,n\}$ for which $x_1 \neq x_2$, we have that

$$\Pr_H[H(x_1) = v_1 \text{ and } H(x_2) = v_2] = 1/n^2. \quad (1)$$

(a) **(10 points)** Verify that the uniform distribution over all functions $H : \{1,\ldots,n\} \to \{1,\ldots,n\}$ is 2-wise independent. Then give an example of a 2-wise independent distribution over hash functions which is not the uniform distribution.

Proof. We will first verify that the uniform distribution over all functions $H : \{1,\ldots,n\} \to \{1,\ldots,n\}$ is 2-wise independent. Since a hash function maps $\{1,\ldots,n\} \to \{1,\ldots,n\}$ and allows repeats, then it follows that there are n^n different hash functions. This is derived from the fundamental principle of counting: each of the n elements in the domain can map to any of the n elements in the codomain, so in total there are n^n total possible hash functions. Now, let us consider the total number of hash functions such that $H(x_1) = v_1$ and $H(x_2) = v_2$. Since x_1 must map to v_1 and x_2

must map to v_2, then there are $n-2$ other elements in the domain of the hash function which need to be mapped to any one of the n elements in the codomain. Thus, by the principle of counting, this is n^{n-2}. Finally, since the uniform distribution over all functions implies that each function has a $\frac{1}{n^n}$ chance of being chosen, it follows that the probability that for a $H(x_1) = v_1$ and $H(x_2) = v_2$ for a chosen hash function is $\frac{n^{n-2}}{n^n} = \frac{1}{n^2}$. Therefore, we have verified that $\Pr_H[H(x_1) = v_1 \text{ and } H(x_2) = v_2] = 1/n^2$ for the uniform distribution over all functions $H : \{1,\ldots,n\} \to \{1,\ldots,n\}$.

Now, we will consider another 2-wise independent distribution over hash functions which is not the uniform distribution. Let $n = p$, where p is some prime number so our functions are $H : \{1,\ldots,p\} \to \{1,\ldots,p\}$. Let us uniformly choose constants i and j from the domain $\{1,\ldots,p\}$. Finally, let us construct hash function H such that $H(x) = ix + j \mod p$, where we include the caveat that $p \mod p = p$ instead of 0 (so that our function does not hash an element to 0 which is not in the codomain).

Now, since we uniformly choose i, j from $\{1,\ldots,p\}$, we have p^2 total possibilities of i and j. Moreover, since i and j uniquely determine a hash function and we choose a hash function from our distribution by selecting i and j, then it follows that each function of form $H(x) = ix + j \mod p$ has probability $\frac{1}{p^2}$ of being chosen and all other functions have zero probability of being chosen. Then, we consider the probability that $H(x_1) = v_1$ and $H(x_2) = v_2$ for any $x_1, x_2, v_1, v_2 \in \{1,\ldots,p\}$. The statement $H(x_1) = v_1$ and $H(x_2) = v_2$ implies finding $H(x_1) = ix_1 + j \mod p = v_1$ and $H(x_2) = ix_2 + j \mod p = v_2$. Therefore, since we have two equations and two unknowns in our system of equations, there must exist a unique i, j for which the above equations hold. In addition, since each hash function contains a unique i, j, then there must only be *1* hash function for which $H(x_1) = ix_1 + j \mod p = v_1$ and $H(x_2) = ix_2 + j \mod p = v_2$.

Thus, as we have previously shown that each function $H(x) = ix + j \mod p$ has $\frac{1}{p^2}$ probability of being chosen, then it follows that $\Pr_H[H(x_1) = v_1 \text{ and } H(x_2) = v_2] = \frac{1}{p^2}$ for a given $x_1, x_2, v_1, v_2 \in \{1,\ldots,p\}$. As such, we have given another 2-wise independent distribution over hash functions which is not the uniform distribution over all hash functions. □

(b) **(0 points, optional)** Let $i \in \{1,\ldots,n\}$. For a hash function H sampled from a 2-wise independent distribution, define the random variable N_i to be the number of elements in $\{1,\ldots,n\}$ that H maps to hash value i. Compute the expectation $\mathbb{E}[N_i]$ and the variance $\mathbb{V}[N_i]$ of this random variable.

(c) **(0 points, optional)**[1] Conclude that with probability at least 2/3, $\max_i N_i \le O(\sqrt{n})$. You may use Chebyshev's inequality, which states that for any random variable Z,

$$\Pr[|Z - \mathbb{E}[Z]| > t] \le \frac{\mathbb{V}[Z]}{t^2}. \tag{2}$$

[1] We won't use this question or 2b for grades. Try it if you're interested. It may be used for recommendations/TF hiring.

3. (a) **(5 points)** In our description of Bloom filters in class, we could only add and look up items. Suppose we try to implement deletion of an item from a Bloom filter by changing the corresponding elements to 0s. Give an example sequence of operations (adds, deletes, and lookups) for which this Bloom filter has both a false positive and a false negative.

Let us consider a bloom filter with three hash functions h_1, h_2, h_3 and three elements x_1, x_2, x_3 we want to perform a sequence of operations on. Additionally, we define the hash tables t_1, t_2, t_3 corresponding to hash functions h_1, h_2, h_3 below ($k = 3$ and $m = 2$):

$$t_1 = \begin{bmatrix} t_{11} & t_{12} \end{bmatrix}, t_2 = \begin{bmatrix} t_{21} & t_{22} \end{bmatrix}, t_3 = \begin{bmatrix} t_{31} & t_{32} \end{bmatrix}$$

Where each $t_{i,j}$ reflects the j-th place in the table t_i. Finally, let us define the hash values obtained for each item x_1, x_2, x_3 as below:

	x_1	x_2	x_3
h_1	t_{11}	t_{11}	t_{11}
h_2	t_{21}	t_{22}	t_{21}
h_3	t_{31}	t_{32}	t_{32}

Then, let us illustrate the sequence of operations add(x_1), add(x_2), add(x_3), delete(x_1), lookup(x_2), which causes the Bloom filter to have both a false positive and a false negative. We begin with all bits in all tables set to 0.

Add(x_1):

$$t_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}, t_2 = \begin{bmatrix} 1 & 0 \end{bmatrix}, t_3 = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Add(x_2):

$$t_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}, t_2 = \begin{bmatrix} 1 & 1 \end{bmatrix}, t_3 = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

Add(x_3):

$$t_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}, t_2 = \begin{bmatrix} 1 & 1 \end{bmatrix}, t_3 = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

This yields a false positive as all the corresponding entries of the hashed values of x_3 all contain 1's. Therefore, the Bloom filter wrongly asserts that x_3 is already a member of the set (speaking generally) when in reality it is not.

Delete(x_1):

$$t_1 = \begin{bmatrix} 0 & 0 \end{bmatrix}, t_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}, t_3 = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

Lookup(x_2):

$$t_1 = \begin{bmatrix} 0 & 0 \end{bmatrix}, t_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}, t_3 = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

In this case, since $h_1(x_2) = t_{11} = 0$, our Bloom filter wrongly asserts that x_2 is not a member of the set (again speaking generally) when in reality we have already added x_2 to the set.

Therefore, the sequence of operations add(x_1), add(x_2), add(x_3), delete(x_1), lookup(x_2) is a sequence of operations for which the above modified Bloom Filter has a false positive and false negative.

(b) **(15 points)** Counting Bloom filters are a variant of Bloom filters where you do not use an array of bits, but an array of small counters. Instead of changing 0s to 1s when an element hashes to a Bloom filter location, you increment the counter. To delete an element, you decrement the counter. To check if an element is in the set, you just check if all of the counter entries are bigger than 0; if they are, then you return the element is in the set. If you see a 0, you say that the element is not in the set. Deletions will work properly for a Counting Bloom filter as long as the counters never overflow; once a counter overflows, you would not have an accurate count of the number of elements currently hashed to that location. A standard choice in this setting in practice to use 4 bit counters. Find an expression for the probability that a specific counter overflows when using 4 bit counters when n elements are hashed into m total locations using k hash functions. (You may have a summation in your expression.) Calculate this probability when $m/n = 8$ and $k = 5$, for $n = 1000, 10000$, and 100000, and calculate the expected number of counters that overflow for each case.

For a 4 bit counter to overflow, we must exceed the maximum possible value stored in the 4 bit counter, which is $2^3 + 2^2 + 2^1 + 2^0 = 15$. Thus, we want to find an expression for the probability that a specific counter has a value that exceeds 15, in which case the counter would overflow.

Let n be the number of elements hashed into one table of size m using k hash functions (thus we have m total locations and nk hash values). Let X be the random variable which denotes the number of elements hashed into a given location. Thus, random variable X is the value of the counter at some specific location in a hash table which follows a binomial distribution. Intuitively, we wish to solve for $P(X > 15)$, which is the probability that X is larger than 15 (in which case the specific counter overflows). Note that the probability $P(X > 15)$ is equivalent to its complement $1 - P(X \leq 15)$. Thus, if we obtain an expression for $1 - P(X \leq 15)$, we will have found an expression for $P(X > 15)$.

Now we will find an expression for $1 - P(X \leq 15)$. This is equivalent to finding $1 - (P(X = 15) + P(X = 14) + ... + P(X = 0))$. Then, let us define a expression for $P(X = i)$. We have nk total hash values for n elements being hashed into one table with mk bins by k hash functions. Note that we have nk hash values because each element is hashed by all k hash functions (so the problem essentially becomes a balls into bins problem where we have nk balls and m bins). We assume that the hash functions are random: thus, each hash value has a probability of $\frac{1}{m}$ of being mapped to a specific

bin/location. Therefore, to find the probability that exactly i elements were hashed into any specific bin is:

$$P(X = i) = \binom{nk}{i} (\frac{1}{m})^i (1 - \frac{1}{m})^{nk-i}$$

We include the binomial coefficient $\binom{nk}{i}$ because we must account for *all* possible ways to have choose i "successful" hashed values out of nk total hashed values. Using this, we can express $P(X > 15)$ as:

$$P(X > 15) = 1 - P(X \leq 15) = 1 - \sum_{i=0}^{15} P(X = i) =$$

$$P(X > 15) = 1 - \sum_{i=0}^{15} \binom{nk}{i} (\frac{1}{m})^i (1 - \frac{1}{m})^{nk-i}$$

Therefore, an expression for the probability that a specific counter overflows when using 4 bit counters is:

$$\Pr[\text{specific counter overflows}] = 1 - \sum_{i=0}^{15} \binom{nk}{i} (\frac{1}{m})^i (1 - \frac{1}{m})^{nk-i}$$

Now, we will calculate this probability and the expected number of counters which overflow for $n = 1000, 10000$, and 100000 when $m/n = 8$ and $k = 5$. To calculate the expected number of overflowed counters, we define random variable I_x for any bit x in our bloom filter as:

$$I_x = \begin{cases} 1 \text{ if x overflows} \\ 0 \text{ otherwise} \end{cases}$$

Therefore, by the linearity of expectation, the expected number o of counters to overflow is:

$$E(o) = \sum_{\text{bit x}} E(I_x)$$

Since there are m bits in the table and we found the probability that any given bit overflows, this becomes:

$$E(o) = (m)(1 * \Pr[\text{specific counter overflows}] + 0)$$

Now, calculating these values for $n = 1000, 10000$, and 100000 when $m/n = 8$ and $k = 5$, we get the following:

For $n = 1000$, $m/n = 8$, and $k = 5$:

$$\Pr[\text{specific counter overflows}] = 1.40807676779E-17$$

Expected number of overflowed bits: 1.126461414232000E-13

For $n = 10000$, $m/n = 8$, and $k = 5$:

$$\Pr[\text{specific counter overflows}] = 1.43647159812E-17$$

Expected number of overflowed bits: 1.1491772784960000E-12

For $n = 100000$, $m/n = 8$, and $k = 5$:

$$\Pr[\text{specific counter overflows}] = 1.43933921414E-17$$

Expected number of overflowed bits: 1.15147137131200000E-11

(c) **(5 points)** Suppose that you use m counters for n elements and k hash functions for a Counting Bloom filter. What is the false positive probability (assuming there has never been an overflow), and how does it compare with the standard Bloom filter?

A false positive occurs when all of the k locations that an element is hashed to in the size m table by the k hash functions contain non-zero counters when the element has not been added to the set beforehand. Therefore, the false positive probability is the probability that all of the locations a specific element is hashed into have a counter greater than 0 before the element itself is looked up or added to the Counting Bloom Filter.

Let us consider one of the k locations a element not in the set is hashed to by a certain hash function. The probability that a specific bit in this hash table is 0 after n elements added to the table is $(1-\frac{1}{m})^{nk}$ because there are nk hashes for n elements and the size of the table is m. Therefore, the probability that this specific bit in the hash table is *not 0* (ie. greater than 0, which is what we want to calculate) is $1-(1-\frac{1}{m})^{nk}$ by the complement property of probabilities. Finally, the probability that this occurs in all k locations which an element is hashed to is $(1-(1-\frac{1}{m})^{nk})^k$. Therefore, the probability of a false positive in the Counting Bloom filter is

$$\Pr_{\text{counting}}[fpp] = (1-(1-\frac{1}{m})^{nk})^k$$

The false positive probability of the standard Bloom filter, which is given in section notes 7, is

$$\Pr_{\text{standard}}[fpp] = (1-(1-\frac{1}{m})^{nk})^k$$

Where n,m, and k are as defined in the problem description. Thus, the probability of a false positive in a counting Bloom filter is the same as the probability of a false positive in the standard Bloom filter. This is reasonable since the only difference between a counting Bloom filter and a standard Bloom filter is the fact that a counting Bloom filter counts the number of elements hashed to a given bit while the standard instead only tracks if the number of elements hashed to that space is greater than 0 (this is a simplification). Since we only care about the probability $1-P(X=0)$, it makes sense that the false positive probability remains the same across both versions.

4. Let $A, B \subset \{1,\ldots,N\}$. In class we saw how to estimate the resemblance $R(A,B) = |A \cap B|/|A \cup B|$ by sampling random permutations $\pi_1,\ldots,\pi_n$ of $\{1,\ldots,N\}$ and computing the fraction of these for

which $\min(\pi(A)) = \min(\pi(B))$.

In the first two parts of this question, we will explore variants of this algorithm. The third part can be solved independently of the first two parts.

(a) **(7 points)** True or false: the algorithm would still work if instead of random permutations, we took $\pi_1, \ldots, \pi_n$ to be random *cyclic* permutations (that is, to sample each π_i, sample a random number y from $\{0, \ldots, N-1\}$ and take π_i to be the permutation which maps every element $x \in \{1, \ldots, N\}$ to $x + y \pmod N$). If the statement is true, provide a proof. Otherwise, you should specify (with proof) some choice of A, B, N such that for a uniformly random *cyclic* permutation, $R(A,B)$ does not equal the probability that $\min(\pi(A)) = \min(\pi(B))$.

Proof. To prove that this statement is false, let us consider the choice of A, B, N such that $A = \{1\}$, $B = \{1,2\}$, and $N = 3$. It follows that we have three possible cyclic permutations:

$$\pi_0 = x + 0 \mod 3, x \in \{1, ..., 3\}$$

$$\pi_1 = x + 1 \mod 3, x \in \{1, ..., 3\}$$

$$\pi_2 = x + 2 \mod 3, x \in \{1, ..., 3\}$$

It is also clear that the set resemblance of A and B is $R(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{1}{2}$. However, consider the probability that $\min(\pi_i(A)) = \min(\pi_i(B))$ for $i \in \{0,1,2\}$.

For $\pi_0 = x \mod 3$, we have $\min(\pi_0(A)) = \min(\pi_0(1)) = \min(1) = 1$ and $\min(\pi_0(B)) = \min(\pi_0(1,2)) = \min(1,2) = 1$. Thus, $\min(\pi_0(A)) = \min(\pi_0(B))$.

For $\pi_1 = x + 1 \mod 3$, we have $\min(\pi_1(A)) = \min(\pi_1(1)) = \min(2) = 2$ and $\min(\pi_1(B)) = \min(\pi_1(1,2)) = \min(2,0) = 0$. Thus, $\min(\pi_1(A)) \neq \min(\pi_1(B))$.

For $\pi_2 = x + 2 \mod 3$, we have $\min(\pi_2(A)) = \min(\pi_2(1)) = \min(0) = 0$ and $\min(\pi_2(B)) = \min(\pi_2(1,2)) = \min(0,1) = 0$. Thus, $\min(\pi_2(A)) = \min(\pi_2(B))$.

Therefore, we can see that the probability that for a uniformly random cyclic permutation, the probability that $\min(\pi(A)) = \min(\pi(B))$ is $\frac{2}{3}$. Since $\frac{2}{3} \neq \frac{1}{2}$, then $R(A,B)$ does not equal the probability that $\min(\pi(A)) = \min(\pi(B))$ for a uniformly random cyclic permutation. □

(b) **(7 points)** Instead of sampling n different random permutations, consider the following algorithm that only uses a single random permutation π: 1) let $\min_n(\pi(A))$ denote the smallest n elements of $\pi(A)$ and let $\min_n(\pi(B))$ denote the smallest n elements of $\pi(B)$, and 2) output m/n, where m is the number of elements that all three sets $\min_n(\pi(A))$, $\min_n(\pi(B))$, and $\min_n(\pi(A \cup B))$ simultaneously have in common.

Prove that if π is a uniformly random permutation, then the expected value of m/n is equal to $R(A,B)$.

Proof. To prove that if π is a uniformly random permutation, then the expected value of m/n is equal to $R(A,B)$, we define random variable I_x on all $x \in A \cap B$:

$$I_x = \begin{cases} 1 \text{ if } \pi(x) \in \min_n(\pi(A \cup B)) \\ 0 \text{ otherwise} \end{cases}$$

Since π is a uniformly random permutation, it follows that any elements $x, y \in A \cup B$ has $\Pr[\pi(x) \in \min_n(\pi(A \cup B))] = \Pr[\pi(y) \in \min_n(\pi(A \cup B))]$. This is proved correct immediately by symmetry (see lemma 3 proof on lecture notes 16). In addition, since a permutation is a one-to-one mapping, we can then say that each element $x \in A \cup B$ has $\Pr[\pi(x) \in \min_n(\pi(A \cup B)] = \frac{n}{|A \cup B|}$.

Next, we will prove that $\pi(x)$ is a member of $\min_n(\pi(A))$, $\min_n(\pi(B))$, and $\min_n(\pi(A \cup B))$ if and only if $x \in A \cap B$ and $\pi(x) \in \min_n(\pi(A \cup B))$. The first direction is proved immediately by the fact that $\pi(x)$ belongs to $\min_n(\pi(A))$ and $\min_n(\pi(B))$. It then follows that x must be an element in both A and B and $x \in A \cap B$. To prove the reverse direction, suppose for the sake of contradiction that $x \in A \cap B$ and $\pi(x) \in \min_n(\pi(A \cup B))$ but $\pi(x) \notin \min_n(\pi(A))$. Then, it follows that there must be at least n different elements in A such that $\pi(x) > \pi(e_i)$ for $i \geq n$. However, these elements e_i would also be included in $A \cup B$ by definition of set union. Thus, there would be at least n elements in $A \cup B$ with $\pi(e_i) < \pi(x)$, implying that $\pi(x) \notin \min_n(\pi(A \cup B))$. Therefore, we have reached a contradiction so $\pi(x) \in \min_n(\pi(A))$. We can prove that $\pi(x) \in \min_n(\pi(B))$ in the exact same manner since $x \in A \cap B$ (the fact that$\pi(x) \notin \min_n(\pi(A \cup B))$ is already assumed and also follows from $\pi(x) \in \min_n(\pi(B))$ and $\pi(x) \in \min_n(\pi(B))$).

Thus, it follows from the previous proof and the linearity of expectation that the expected number of elements m that all three sets $\min_n(\pi(A))$, $\min_n(\pi(B))$, and $\min_n(\pi(A \cup B))$ simultaneously have in common is:

$$E(m) = \sum_{x \in A \cap B} E(I_x)$$

$$E(m) = |A \cap B| * (1 * \frac{n}{|A \cup B|}) + 0$$

$$E(m) = \frac{n|A \cap B|}{|A \cup B|}$$

Therefore, since n is static,

$$E(\frac{m}{n}) = \frac{E(m)}{n} = \frac{|A \cap B|}{|A \cup B|}$$

Since $R(A,B)$ is defined as $R(A,B) = \frac{|A \cap B|}{|A \cup B|}$, we have thus proved that the expected value $\frac{m}{n}$ is equal to $R(A,B)$ □

(c) **(7 points)** Let A be a uniformly random subset of $\{1, \ldots, N\}$ of size t, and let B be a uniformly

random subset of $\{1,\ldots,N\}$ of size t. Use the guarantee for MinHash to prove that the expected value of $R(A,B)$ is given by the expression

$$\sum_{i=1}^{N}\left(\binom{N-i}{t-1}\Big/\binom{N}{t}\right)^2. \tag{3}$$

Proof. The guarantee for MinHash is as follows:

$$R(A,B) = \Pr[\min(\pi(A)) = \min(\pi(B))]$$

Thus, we can use the Law of Total Probability to express:

$$\Pr[\min(\pi(A)) = \min(\pi(B))] = \sum_{i=1}^{N} \Pr[\min(\pi(A)) = \min(\pi(B)) = i]$$

However, if A and B are uniformly random subsets of $\{1,...,N\}$ of size t, this implies that they are chosen independently. We can also express the probability that $\min(\pi(A)) = \min(\pi(B)) = i$ is the probability that $\min(\pi(A)) = i$ and $\min(\pi(B)) = i$. Therefore, since A and B being chosen independently implies the previous two probabilities are independent, we have:

$$\Pr[\min(\pi(A)) = \min(\pi(B)) = i] = \Pr[\min(\pi(A)) = i] * \Pr[\min(\pi(B)) = i]$$

Therefore, by the linearity of expectation, we can express the expected value of $E(R(A,B))$ as:

$$E(R(A,B)) = \sum_{i=1}^{N} E(\Pr[\min(\pi(A)) = i]) * E(\Pr[\min(\pi(B)) = i])$$

Now, without loss of generality (since A and B are selected independently, uniformly, randomly, and of the same size), we will find the expected value $E(\Pr[\min(\pi(A)) = i])$. Note that we can assert $E(\Pr[\min(\pi(A)) = i]) = \Pr[\min(\pi(A)) = i]$ since the probability that $\min(\pi(A)) = i$ does not vary if A is always a uniformly random subset of fixed set $[1,...,N]$ and of fixed size t.

Our job now is to determine the probability that $\min(\pi(A)) = i$ for a given $i \in \{1,...,N\}$. Intuitively, the probability of this is the total number of permutations $\pi(A)$ such that $\min(\pi(A)) = i$ divided by the total possible permutations $\pi(A)$. We start by finding the total possible $\pi(A)$. Since by definition, a permutation $\pi(A)$ maps elements $[1,...,t]$ of A to elements $[1,...,N]$ in a one to one manner, it follows that the number of permutations $\pi(A)$ is $\binom{N}{t}$.

Now, we consider the total possible permutations $\pi(A)$ with $\min(\pi(A)) = i$. For i to be the minimum of $\pi(A)$, the other $t-1$ elements in $\pi(A)$ must be greater than i. Furthermore, it follows that within the broader interval $[1,...,N]$, there are $N-i$ elements larger than i (since a permutation maps $[1,...,N]$ onto $[1,...,N]$, i is guaranteed to be in $[1,...,N]$). Thus, for the other $t-1$ elements in A which do not map to i, there

are $N-i$ possible elements greater than i to choose the remaining $t-1$ elements to map to in a permutation. Therefore, there are for any i, $\binom{N-i}{t-1}$ possible permutations for which $\min(\pi(A)) = i$.

Combining the last two paragraphs, we have $\binom{N-i}{t-1}$ possible permutations $\pi(A)$ which satisfy $\min(\pi(A)) = i$ out of $\binom{N}{t}$ possible permutations $\pi(A)$. Therefore, the probability that $\min(\pi(A)) = i$ for a given $i \in \{1, ..., N\}$ is:

$$\Pr[\min(\pi(A)) = i] = \frac{\binom{N-i}{t-1}}{\binom{N}{t}}$$

Therefore, the expected value $E(\Pr[\min(\pi(A)) = i])$ is:

$$E(\Pr[\min(\pi(A)) = i]) = \frac{\binom{N-i}{t-1}}{\binom{N}{t}}$$

Since we determine $E(\Pr[\min(\pi(B)) = i])$ in the exact same manner, we get:

$$E(R(A,B)) = \sum_{i=1}^{N} \frac{\binom{N-i}{t-1}}{\binom{N}{t}} * \frac{\binom{N-i}{t-1}}{\binom{N}{t}}$$

$$E(R(A,B)) = \sum_{i=1}^{N} (\frac{\binom{N-i}{t-1}}{\binom{N}{t}})^2$$

Thus, we have proved that the expected value of $R(A,B)$ is given by the expression

$$\sum_{i=1}^{N} (\frac{\binom{N-i}{t-1}}{\binom{N}{t}})^2$$

□

5. Recall the fingerprinting algorithm presented in lecture 15 for determining whether a pattern of length k appears in a document of length n characters, each a number between 0 and 9: we hash the pattern by taking the decimal number it represents mod p; we hash each substring of length k of the document by taking the decimal number it represents mod p, and, for every hash match, we compare the corresponding place in the document to the pattern, character by character, stopping if we find an instance of the pattern. For instance, comparing integers 31415 to 31415 takes 5 comparisons of characters (and then we can stop the algorithm, having found an occurrence of the pattern), and comparing 31415 to 31428 takes 4 comparisons of characters (3, 1, 4, x) (but we must move on to checking any other hash matches, in case one of them is a match).

   Instead of choosing a random prime number for p, say that we use some fixed prime number p that is made public. Sara knows p and wants to slow down your algorithm, so she is trying to construct a document and pattern that force you to do many comparisons of characters.

   (a) **(5 points)** Suppose $p = 1249$, the pattern P is $[6,2,9,5,1,4,1,3]$ (so $k = 8$), and the document

D is [2,3,1,1,5,7,1,7,1,4,1,3,3,9,3,8,7,6,1,4,9,1,3,8,4,3,8,6,4,1] (so $n = 30$). The algorithm makes 11 comparisons of characters. What are those comparisons?

Let $p = 1249$, $P = [6,2,9,5,1,4,1,3]$, and
$D = [2,3,1,1,5,7,1,7,1,4,1,3,3,9,3,8,7,6,1,4,9,1,3,8,4,3,8,6,4,1]$.
We get the following computations.

$H(P) = H(31415926) = 31415926(\mod 1249) = 1078$
$H(D_{22}) = H(14683483) = 14683483(\mod 1249) = 239$
$H(D_{21}) = H(46834831) = 46834831(\mod 1249) = 1078$
$H(D_{20}) = H(68348319) = 68348319(\mod 1249) = 541$
$H(D_{19}) = H(83483194) = 83483194(\mod 1249) = 0$
$H(D_{18}) = H(34831941) = 34831941(\mod 1249) = 1078$
$H(D_{17}) = H(48319416) = 48319416(\mod 1249) = 602$
$H(D_{16}) = H(83194167) = 83194167(\mod 1249) = 775$
$H(D_{15}) = H(31941678) = 31941678(\mod 1249) = 1001$
$H(D_{14}) = H(19416783) = 19416783(\mod 1249) = 1078$
$H(D_{13}) = H(94167839) = 94167839(\mod 1249) = 733$
$H(D_{12}) = H(41678393) = 41678393(\mod 1249) = 512$
$H(D_{11}) = H(16783933) = 16783933(\mod 1249) = 1120$
$H(D_{10}) = H(67839331) = 67839331(\mod 1249) = 1145$
$H(D_9) = H(78393314) = 78393314(\mod 1249) = 1078$
$H(D_8) = H(83933141) = 83933141(\mod 1249) = 341$
$H(D_7) = H(39331417) = 39331417(\mod 1249) = 407$
$H(D_6) = H(93314171) = 93314171(\mod 1249) = 132$
$H(D_5) = H(33141717) = 33141717(\mod 1249) = 751$
$H(D_4) = H(31417175) = 31417175(\mod 1249) = 1078$
$H(D_3) = H(14171751) = 14171751(\mod 1249) = 597$
$H(D_2) = H(41717511) = 41717511(\mod 1249) = 911$
$H(D_1) = H(17175113) = 17175113(\mod 1249) = 114$
$H(D_0) = H(71751132) = 71751132(\mod 1249) = 1078$

The algorithm makes comparisons at $H(D_{21})$, $H(D_{18})$, $H(D_{14})$, $H(D_9)$, $H(D_4)$, and $H(D_0)$. At each of those. At $H(D_0)$, we check that 3 does not match 7 in 1 comparison and then stop (x). At $H(D_4)$, we make 5 comparisons since the first 4 characters of P and D_4 match but the fifth does not match (3,1,4,1,x). At $H(D_9)$, we check that 3 does not match 7 in 1 comparison and then stop (x). At $H(D_{14})$, we check that 3 does not match 1 in 1 comparison and then stop (x). At $H(D_{18})$, we make 2 comparisons since the first character of P and D_{18} match but the second does not match (3,x). Finally, at $H(D_{21})$, we check that 3 does not match 4 in 1 comparison and then stop (x). In total, these account for the 11 comparisons of characters.

(b) **(12 points)** Give an upper bound on the number of comparisons of characters that the algorithm performs in terms of the length n of the document, the length $k \leq n$ of the pattern, and the prime p.

For full credit, your bound should be asymptotically tight for large n: that is, there should exist a constant c such that if n is large enough (where "large enough" may depend on k and p), your answers to this and the next question should differ by at most a factor of c. Correctly-proven bounds that don't quite match are eligible for partial credit.

Proof. Let n be the length of the document, $k \leq n$ be the length of the pattern, and prime number p.

In the algorithm, we define D_i as the length k substring of D starting at index i in the array. However, since D_i ends at index $i+k$ naturally from its definition, we have $0 \leq i \leq n$ and $i+k \leq n$. Therefore, $0 \leq i \leq n-k$. This implies that we have $n-k+1$ substrings D_i which our algorithm checks. The worst case is when our hashing function using $\mod p$ is such that each of these substrings D_i has the same hashed value as the pattern we are trying to detect. Moreover, in the worst case, we have k comparisons of characters for a given substring since we must continue comparing corresponding elements in the pattern and D_i until either two characters don't match or the entire substring and pattern matches. Thus, if the last character in D_i does not match with the pattern, we still make k comparisons without terminating the algorithm early. Therefore, the upper bound of total comparisons of characters is $k*(n-k+1) = kn-k^2+k = O(kn)$ for the algorithm. □

(c) **(12 points)** Give a family of examples (each consisting of a document, a pattern, and p) that require asymptotically as many comparisons of characters as possible.

Proof. Let us define a document $D = [1,1,1,...,1]$ where D is of length n and consists of only 1s. In addition, let us define pattern P such that $P = [1,1,1,...,3]$ where P has the first $k-1$ elements as 1s and the k-th element as 3 (total length k). Finally, we choose prime number $p = 2$ so that our hashing function is $H(x) = x \mod 2$.

Since any substring D_i contains only 1s and is therefore odd, then it follows that $H(D_i) = 1 \mod 2$. Furthermore, since the last digit in the pattern P is 3, it also follows that $H(P) = 1 \mod 2$. Therefore, every substring D_i has the same hash value as the hash value of the pattern P, so our algorithm is forced to check all substrings. In addition, by the construction of each substring, it is clear that the first $k-1$ elements match the first $k-1$ elements of the pattern for any D_i, causing our algorithm to keep performing character comparisons on the next element pair. Then, at the k-th comparison between the last elements in the substring and pattern, our algorithm verifies $1 \neq 3$, leading to no pattern match and our algorithm must continue checking the next substring.

Therefore, since we have $n-k+1$ substrings D_i in our document and we perform k checks for all substrings, then it follows that this family of examples for any document length n and pattern length $k \geq 2$ yields $k(n-k+1) = O(kn)$ character comparisons. Thus, this family of examples require asymptotically as many comparisons

of characters as possible. □

(d) **(0 points, optional)**[2] Make your answers to the previous two questions asymptotically tight in terms of p, n, and k: that is, there should exist a constant c such that for all sufficiently large n, k, and p your answers are within a factor c of each other. For instance, an upper bound of $O(n)$ and examples achieving $\Omega(n-k)$ would satisfy the requirements for full credit, but not for this optional problem.

[2] We won't use this question for grades. Try it if you're interested. It may be used for recommendations/TF hiring.