CS 124 Homework 5: Spring 2024

Anusha Murali

Collaborators: None

No. of late days used on previous psets: 4
No. of late days used after including this pset: 5

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 p 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 collison 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.

Solution: We define $\mathbb{1}_{i,j}$ to be an indicator (or binary random variable), which is 1 when positions i and j have a collision, otherwise it is 0. Now, $\mathbb{1}_{i,j} = 1$ if positions i and j have 1 in each or 2 in each, and so on. This means that,

$$\tau := \mathbb{P}\left[\mathbb{1}_{i,j} = 1\right] = \sum_{k=1}^{m} p_k^2 \implies \mathbb{E}\left[\mathbb{1}_{i,j}\right] = \sum_{k=1}^{m} p_k^2.$$

Let the number of collisions be X. Therefore, $X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{1}_{i,j}$. This gives us the following.

$$\begin{aligned}
\mathbb{E}[X] &= \mathbb{E}\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{1}_{i,j}\right] \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{E}\left[\mathbb{1}_{i,j}\right] && \text{(Linearity of expectations.)} \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \tau \\
&= \frac{n(n-1)}{2} \sum_{k=1}^{m} p_k^2
\end{aligned}$$

This quantity is maximized when one of p_i's is equal to 1 and the rest are equal to 0. In this case, $\mathbb{E}[X] = \frac{n(n-1)}{2}$. This is the case because all the n points are the same, hence, every point is colliding with one another, which is the most we could have.

On the other hand, this quantity is minimized when all the p_i's are the same, that is, equal to $\frac{1}{m}$ each. Again, this makes sense intuitively because no element in the domain have a higher probability of producing samples than any other point this way, so the sum of the squares gets minimized when the distribution is uniform.

Another way to look at it informally is the following. Suppose $m = 2$. We want to maximize $p_1^2 + p_2^2$ subject to the constraint $p_1 + p_2 = 1$, where $p_1, p_2 \geq 0$. The former is just a circle around the origin, once we fix its radius. The latter just represents the line segment $x + y = 1$ in the quadrant where both $x, y \geq 0$. The radius of the largest circle that can intersect with this line segment is 1, which happens only when either $x = 0, y = 1$ or $x = 1, y = 0$. On the other hand, the radius of the smallest circle that intersects with this line segment is $\frac{1}{\sqrt{2}}$, which can only happen when $x = y = \frac{1}{2}$.

(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 happens 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.

Solution: We disprove this statement for $m = 2$ and $n = 2$. Let $\mathbb{P}[x_i = 1] = p$. Then,

$$\mathbb{P}[A] = 2p - p^2 \quad \text{and} \quad \mathbb{P}[B] = p^2.$$

We have the following.

$$\begin{aligned}\mathbb{P}[A\cap B] &= \mathbb{P}[B]\mathbb{P}[A|B]\\ &= p^2\cdot 1\\ &= p^2\end{aligned}$$

In the above, the second equality holds because if B happens, then it must be the case that both the samples are 1, which satisfies A. Next, we have

$$\begin{aligned}\mathbb{P}[A]\cdot\mathbb{P}[B] &= (2-p)p^3\\ &\neq \mathbb{P}[A\cap B].\end{aligned}$$

Therefore, A and B are not independent for $m=2$ and $n=2$, which disproves the claim for general $m,n\geq 2$.

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

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

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

Solution: Let $\mathscr{H}$ be the family of all hash functions from $[n]$ to $[n]$. We make the following two assumptions on the geometry of $\mathscr{H}$.

i. For any $x,v\in[n]$, the volume of the set of all hash functions h, such that $h(x)=v$, is some $\tau>0$. In other words, for any $x_1,x_2,v_1,v_2\in[n]$, the volume of the set of all hash functions h, such that $h(x_1)=v_1$, equals the volume of all hash functions h, such that $h(x_2)=v_2$.

ii. For any $x_1,x_2,v_1,v_2\in[n]$, the volume of the set of all hash functions h, such that $h(x_1)=v_1$ and $h(x_2)=v_2$, is equal to some $\gamma>0$. In other words, for any $x_1,x_2,x_3,x_4,v_1,v_2,v_3,v_4\in[n]$, the volume of the set of all hash functions h, such that $h(x_1)=v_1$ and $h(x_2)=v_2$, equals the volume of the set of all hash functions, such that $h(x_3)=v_3$ and $h(x_4)=v_4$.

We make a volume-based argument to prove the 2-wise independence of the uniform distribution. For a fixed $h\in\mathscr{H}$, $\mathbb{P}_H[H(x_1)=v_1\wedge H(x_2)=v_2]$ is either 0 or 1 because h itself is deterministic. Then we have the following.

$$\begin{aligned}\mathbb{P}_H[H(x_1)=v_1\wedge H(x_2)=v_2] &= \int\limits_{h\in\mathscr{H}}\mathbb{P}[h(x_1)=v_1\wedge h(x_2)=v_2]f(h)dh\\ &= \frac{1}{n^2}\end{aligned}$$

The second equality in the above holds because of the fact about $\mathbb{P}_H[H(x_1)=v_1\wedge H(x_2)=v_2]$, which implies that the integral is simply measuring the (normalized) volume of the set of all hash functions such that $h(x_1)=v_1$ and $h(x_2)=v_2$, which should be $\frac{1}{n^2}$ due to our assumption on the geometry of $\mathscr{H}$.

Following is an example of a 2-wise independent distribution over hash functions, which is not the uniform distribution. A family of such hash functions can be constructed over any prime number, $p > n$, where n is the input to our hash function.

Let us select a prime $p > n$ and then choose a, b independently and uniformly at random such that $a, b \in \{0, \ldots, p-1\}$. We now construct our hash function using a and b such that $h(n) = m = an + b \pmod{p}$, which forms our family of hash functions $\mathscr{H} = \{h | h : [n] \to [m]\}$ for any arbitrary prime $p > n$.

(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.

Solution: Let us construct a Bloom filter with two hash functions and a 26-bit array, A, which indexes each letter in the English alphabet. In other words, our Bloom filter has $k = 2$ and $m = 26$. The first hash function, H_1, turns on the index in the array that corresponds to the first letter of the incoming word, while the second hash function, H_2, turns on the index in the array that corresponds to the last letter of the incoming word. Assume that the 26-bit array is 0-indexed.

Initially all entries of array A are set to 0.

Consider the following sequence of operations:

i. Add "BLOOM": H_1("BLOOM") turns on bit 1 (corresponding to the first letter "B") and H_2("BLOOM") turns on bit 12 (corresponding to the last letter "M"). So $A[1] = 1$ and $A[12] = 1$.
ii. Add "BOOK": H_1("BOOK") turns on bit 1 (corresponding to the first letter "B") and H_2("BOOK") turns on bit 10 (corresponding to the last letter "K"). So $A[1] = 1$ and $A[10] = 1$.
iii. Add "HASH": H_1("HASH") turns on bit 7 (corresponding to the first letter "H") and H_2("HASH") turns on bit 7 (corresponding to the last letter "H"). So $A[7] = 1$.
iv. Delete "BLOOM": Since H_1("BLOOM") = 1 and H_2("BLOOM") = 12, both bits 1 and 12 are now set to 0. So, $A[1] = 0$ and $A[12] = 0$.
v. Lookup "HACK": Since $A[H_1(\text{"HACK"})] = A[7] = 1$, and $A[H_2(\text{"HACK"})] = A[10] = 1$, we report "FOUND". This is a false-positive as "HACK" was never added.
vi. Lookup "BOOK": Since $A[H_1(\text{"BOOK"})] = A[1] = 0$, we report "NOT FOUND". This is a false-negative as "BOOK" was never removed.

(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.

Solution: If there are a total of n elements currently, and we use k distinct hash functions, the total count of increments is nk, as each of the k hash functions will increment 1 bit for each of the n elements.

Now, consider an analogy of m bins and a total of kn balls. If we throw kn balls randomly into m bins, the probability that the number of balls in one bin being i is,

$$\operatorname{Bin}\left(i, kn, \frac{1}{m}\right) = \binom{kn}{i}\left(\frac{1}{m}\right)^{i}\left(1-\frac{1}{m}\right)^{kn-i}, \tag{3}$$

where Bin denotes binomial distribution.
Then the probability that there are more than j balls in this bin is,

$$P = 1-\sum_{i=0}^{j}\operatorname{Bin}\left(i, kn, \frac{1}{m}\right) = 1-\sum_{i=0}^{j}\binom{kn}{i}\left(\frac{1}{m}\right)^{i}\left(1-\frac{1}{m}\right)^{kn-i} \tag{4}$$

In the Counting Bloom filter with k distinct hash functions, n elements and m total locations in the array using a 4-bit counter, we are interested in finding the probability that a specific counter overflows. In a 4-bit counter, overflow occurs when the counter exceeds a count of 15. Therefore, the 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 is,

$$\boxed{P = 1-\sum_{i=0}^{15}\binom{kn}{i}\left(\frac{1}{m}\right)^{i}\left(1-\frac{1}{m}\right)^{kn-i}}.$$

Let P 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, and let $E(X)$ the expected number of counters that overflow for a given n. The expected number of counters, $E(X)$, which overflow is computed by multiplying the probability P that a specific counter overflows by m, which is the total number of counters.
The following table shows the above probability P and the expected number of counters that overflow for the given values of n.

n	m	P	$E(X)$
1000	8000	2.089×10^{-13}	1.672×10^{-9}
10000	80000	≈ 0	≈ 0
100000	800000	≈ 0	≈ 0

Note: For $n = 10000$ and $n = 100000$, I couldn't correctly compute the probability due to loss of precision using Python. Other students have also reported this issue on Ed.
In order to avoid the above round off errors, I have provided a loose upper bound below.
From Equation 3 above, the probability that the count of one location being 16 is,

$$P[\text{count} = 16] = \operatorname{Bin}\left(16, kn, \frac{1}{m}\right) = \binom{kn}{16}\left(\frac{1}{m}\right)^{16}\left(1-\frac{1}{m}\right)^{kn-16}.$$

Therefore, we find a loose upper bound to be,

$$P[\text{count} \geq 16] \;\leq\; m\binom{nk}{16}\frac{1}{m^{16}} \;\leq\; m\left(\frac{enk}{16m}\right)^{16}.$$

Using the above loose upper bound expression for the probability that a specific counter overflows, following is the updated table showing the probabilities and the expected number of counters that overflow for the given values of n:

n	m	P	$E(X)$
1000	8000	$\leq 2.089 \times 10^{-12}$	$\leq 1.6721 \times 10^{-8}$
10000	80000	$\leq 2.089 \times 10^{-11}$	$\leq 1.6721 \times 10^{-6}$
100000	800000	$\leq 2.089 \times 10^{-10}$	$\leq 1.6721 \times 10^{-4}$

(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?

Solution: The Equation 4 derived above gives the probability that there are more than j balls in the bin. Therefore, using the bins and balls analogy, the false positive probability of the Counting Bloom filter, which is equal to the probability of an element exceeding a count of j in the Counting Bloom filter is,

$$P_{\text{false positive}} = \left(1 - \sum_{i=0}^{j} \binom{kn}{i} \left(\frac{1}{m}\right)^{i} \left(1 - \frac{1}{m}\right)^{kn-i}\right)^{k}$$

When we substitute $j = 0$ in the above expression, the false positive probability becomes equal to the false positive probability of a regular Bloom filter. □

4. Let $A, B \subset \{1, \dots, N\}$. In class we saw how to estimate the resemblance $R(A,B) = |A \cap B|/|A \cup B|$ by sampling random permutations $\pi_1, \dots, \pi_n$ of $\{1, \dots, N\}$ 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 $\pi_1, \dots, \pi_n$ to be random *cyclic* permutations (that is, to sample each π_i, sample a random number y from $\{0, \dots, N-1\}$ and take π_i to be the permutation which maps every element $x \in \{1, \dots, N\}$ 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 $\min(\pi(A)) = \min(\pi(B))$.

Solution: The claim is false. We will prove this with the following counter example.

Let us consider the two sets $A, B \subset \{1,2,3\}$, where $N = 3$, $A = \{2\}$ and $B = \{1,2\}$. The set resemblance (or Jaccard coefficient) is,

$$R(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{1}{2}.$$

Representing the elements in the set as rows and the sets A and B as columns, we have the following characteristic matrix:

	A	B
1	0	1
2	1	1
3	0	0

Since there are 3 elements in our example, namely 1, 2, and 3, using a cyclic permutation, we can permute the rows in the above characteristic matrix in only three ways, which are shown below along with the MinHash values for $\pi(A)$ and $\pi(B)$:

i. <u>Cyclic Permutation 1</u>

A	B
0	1
1	1
0	0

$\text{MinHash}(\pi_1(A)) = 2$
$\text{MinHash}(\pi_1(B)) = 1$

Therefore, $\text{MinHash}(\pi_1(A)) \neq \text{MinHash}(\pi_1(B))$.

ii. <u>Cyclic Permutation 2</u>

A	B
1	1
0	0
0	1

$\text{MinHash}(\pi_2(A)) = 1$
$\text{MinHash}(\pi_2(B)) = 1$

Therefore, $\text{MinHash}(\pi_1(A)) = \text{MinHash}(\pi_1(B))$.

iii. Cyclic Permutation 3

A	B
0	0
0	1
1	1

MinHash($\pi_3(A)$) = 3
MinHash($\pi_3(B)$) = 2

Therefore, MinHash($\pi_1(A)$) $\neq$ MinHash($\pi_1(B)$).

Hence, out of the three possible cyclic permutations, each corresponding to modulo 3, MinHash match occurs only on one of them. Hence the average MinHash match for the above cyclic permutations, which is equal to the probability that MinHash($\pi_1(A)$) = MinHash($\pi_1(B)$) is $\frac{1}{3}$, while the actual set resemblance is $\frac{1}{2}$. Therefore, the algorithm will not work if we use random cyclic permutations.

(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 $R(A,B)$.

Solution: In this variant of the set resemblance problem, we use only a single random permutation π, where $\min_n(\pi(A))$ denotes the smallest n elements of $\pi(A)$, $\min_n(\pi(B))$ denotes the smallest n elements of $\pi(B)$ and A, and B are two random subsets selected from the given universal set.
Let $\min_n(\pi(A))$ and $\min_n(\pi(B))$ denote the n smallest elements returned by our hash functions from sets A and B respectively. Therefore, $\min_n(\pi(A)) \cup \min_n(\pi(B))$ is the union of these elements. We note that the elements returned by $\min_n(\pi(\min_n(\pi(A)) \cup \min_n(\pi(A)))$ are the same as the n elements in the set $\min_n(\pi((A \cup B))$, as they come from the n smallest elements from each set returned by the hash function. Let $S = \min_n(\pi(\min_n(\pi(A)) \cup \min_n(\pi(A)))$. Hence $|S| = n$. Since MinHash function is applied on the given random permutation, the set $S = \min_n(\pi(\min_n(\pi(A)) \cup \min_n(\pi(A)))$ is a simple random sample of $A \cup B$.
Now, consider the intersection of the above set S and the intersection of both $\min_n(\pi(A))$ and $\min_n(\pi(B))$. Let this set be T, so $T = S \cap \min_n(\pi(A)) \cap \min_n(\pi(B))$. Since 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, it follows that $m = |T|$. Therefore,

$$\frac{|T|}{|S|} = \frac{|S \cap \min_n(\pi(A)) \cap \min_n(\pi(B))|}{|\min_n(\pi(\min_n(\pi(A)) \cup \min_n(\pi(A)))|} = \frac{m}{n},$$

is an unbiased estimator of the set resemblance $R(A,B)$ as desired. □

(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

$$\sum_{i=1}^{N}\left(\binom{N-i}{t-1}\Big/\binom{N}{t}\right)^2. \qquad (5)$$

We shall use the guarantee for MinHash to prove the expected value of $R(A,B)$ as follows.

Let us consider two uniformly random subsets A and B of $\{1,\ldots,N\}$. Let their sizes be $|A| = |B| = t$.

Consider an instance of their following characteristic matrix, where $N = 5$ and $t = 2$. As can be seen, the random subset A contains elements 3 and 5, while the random subset B contains elements 3 and 4.

Element	A	B
1	0	0
2	0	0
3	1	1
4	0	1
5	1	0

We distinguish three types of rows in the above matrix.

- Type X rows have 1 in both columns,
- Type Y rows have 1 in one of the columns and 0 in the other,
- Type Z rows have 0 in both columns.

The ratio of the number of type X and type Y rows determine both the set resemblance $R(A,B)$ and the probability that $\min_N(\pi(A)) = \min_N(\pi(B))$.

Let x be the number of rows of type X and y be the number of rows of type Y. Then the set resemblance is,

$$R(A,B) = \frac{x}{x+y}.$$

If we imagine the rows of the above characteristic matrix permuted randomly and we proceed from the top, the probability that we encounter a type X row before we encounter a type Y row is precisely the the probability that $\min_N(\pi(A)) = \min_N(\pi(B))$. Therefore,

$$P\left(\min_N(\pi(A)) = \min_N(\pi(B))\right) = \frac{x}{x+y}.$$

Therefore, we find that both the set resemblance $R(A,B)$ and the probability that $\min_N(\pi(A)) = \min_N(\pi(B))$ are equal and,

$$R(A,B) = P\left(\min_N(\pi(A)) = \min_N(\pi(B))\right) = \frac{x}{x+y}.$$

We shall prove that the above ratio is equal to the expression given in the problem as follows.

Let us consider a scenario where there are i number of type X rows. So the remaining $t-i$ rows must be that of type Y.

In order to find the MinHash of the sets A and B, we are only looking for the very first type X row from the top. Let us say that it occurs on the i-th row from the top. Therefore, the remaining $t-1$ rows (where we have 1's) can occupy the $N-i$ rows that are found below the first X type row in any order. The number of ways this can happen is,

$$\binom{N-i}{t-1}.$$

The number of ways to select t rows randomly from the universal set of N rows is,

$$\binom{N}{t}.$$

Hence, the probability that the first type X row of set A appearing on row i is,

$$\frac{\binom{N-i}{t-1}}{\binom{N}{t}}.$$

Therefore, the expected value that the first type X row of set A and set B appearing on row i simultaneously is

$$\left(\frac{\binom{N-i}{t-1}}{\binom{N}{t}}\right)^2.$$

Since the first type X row can appear anywhere from $i = 1$ to $i = N$, the total expected value that we are interested in is,

$$R(A,B) = \frac{x}{x+y} = \sum_{i=1}^{N} \left(\frac{\binom{N-i}{t-1}}{\binom{N}{t}}\right)^2.$$

□

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?

   Solution: The 11 comparisons between the patten P and the document D are highlighted below in red.

P	D_i	Number of comparisons
[6,2,9,5,1,4,1,3]	$D_{21} = [1,3,8,4,3,8,6,4]$	1
[6,2,9,5,1,4,1,3]	$D_{18} = [1,4,9,1,3,8,4,3]$	2
[6,2,9,5,1,4,1,3]	$D_{14} = [3,8,7,6,1,4,9,1]$	1
[6,2,9,5,1,4,1,3]	$D_9 = [4,1,3,3,9,3,8,7]$	1
[6,2,9,5,1,4,1,3]	$D_4 = [5,7,1,7,1,4,1,3]$	5
[6,2,9,5,1,4,1,3]	$D_0 = [2,3,1,1,5,7,1,7]$	1
		11

   (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 $k \le n$ 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.

   Solution:

   The algorithm can do at most $O(nk)$ comparisons in the worst case trivially because it will do that many comparisons to find all matches.

   (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.

   Solution: Let k be a multiple of the ceiling of $\log_{10}(p)$ (let's call that quantity ℓ). Then we claim that the family we propose involves $O\left(\frac{nk}{\log_{10}(p)}\right)$ comparisons. Let x_p be the pattern such that it is a repeat of the prime number (p) k/ℓ times, x'_p be the pattern such that it is a repeat of the prime number (p) $k/\ell - 1$ times, and s_{2p} be the string to represent the number $2p$. The document d is as follows: $x'_p s_{2p} x'_p s_{2p} \ldots$, such that the length of d is at least $k + \ell$. The above algorithm will hash x_p to 0, and will hash D_0 to 0, as well. It will force the algorithm to perform $k - \ell + 1$ comparisons. Then the next hash of D_1 may not match the hash of x_p. However, the hash of D_ℓ will evaluate to 0, and match that of x_p. This way, after the next ℓ hashes, the algorithm would be forced to make $k - 2\ell + 1$ comparisons. After that, $k - 3\ell + 1$ comparisons, and so on until 1. After this, the whole sequence repeats. In total, we end up having to start the comparing n/ℓ times approximately. When we sum all these numbers (using the sum of a geometric series), we have $O\left(\frac{nk}{\log_{10}(p)}\right)$ comparisons. □

(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.