Problem 1: (LSH for Angular Similarity)

For any vectors \( x, y \in \mathbb{R}^d \), the angular distance is the angle (in radians) between the two vectors – formally, \( d_\theta(x, y) = \cos^{-1}\left(\frac{x \cdot y}{||x||_2 ||y||_2}\right) \) (where \( \cos^{-1}(\cdot) \) returns the principle angle, i.e., angles in \([0, \pi]\)). The (normalized) angular similarity is given by \( s_\theta(x, y) = 1 - d_\theta(x, y)/\pi \).

We now want to construct a LSH for the angular similarity metric. Consider the following family of hash functions: we first choose a random unit vector \( \sigma \) (i.e., \( \sigma \in \mathbb{R}^d \) with \( ||\sigma||_2 = 1 \)), and for any vector \( x \), define \( h_\sigma(x) = sgn(x.\sigma) \) (i.e., the sign of the dot product of \( x \) and \( \sigma \)). Argue that for any \( x, y \in \mathbb{R}^d \), we have:

\[
\mathbb{P}[h_\sigma(x) = h_\sigma(y)] = s_\theta(x, y)
\]

Hint: For any pair \( x \) and \( y \) in \( \mathbb{R}^d \), there is a unique plane passing through the origin containing \( x \) and \( y \) – convince yourself that \( d_\theta(x, y) \) is precisely the angle between \( x \) and \( y \) in this plane. Also, given any vector \( \sigma \), its dot product with \( x \) and \( y \) only depends on the projection of \( \sigma \) on this plane. Now what can you say about the signs of the dot products of \( x \) and \( y \) with a random unit vector?

Problem 2: (Choosing LSH Parameters for Nearest Neighbors)

An important routine in many clustering/machine learning algorithms is the \((c, R)\)-Nearest-Neighbors (or \((c, R)\)-NN) problem: given a set of \( n \) points \( V \) and a distance metric \( d \), we want to store \( V \) in order to support the following query:

Given a query point \( q \), if there exists \( x \in V \) such that \( d(x, q) \leq R \) then, with probability at least \( 1 - \delta \), we must output a point \( x' \in V \), such that \( d(x', q) \leq cR \).

We now show how to solve this problem using LSH. Assume that we are given a \((R, cR, p_1, p_2)\)-sensitive hash family \( H \). As in class, we can amplify the probabilities by first taking the AND of \( r \) such hash functions to get a new family \( H_{\text{and}} \); next, we can take the OR of \( b \) hash functions from \( H_{\text{AND}} \) to get another family \( H_{\text{OR-AND}} \).

Given the set \( V \), we hash each element using a single hash function \( g \) from \( H_{\text{OR-AND}} \) (which corresponds to \( b \times r \) hash functions from \( H \)). Now given a query point \( q \), we hash \( q \) using our cascaded hash-function \( g \), and find all \( y \in V \) such that \( g(y) = g(q) \) – let this set be denoted \( Y_q \). Finally, we can check \( d(q, y) \) for each \( y \) in \( Y_q \), and return those \( y \) for whom \( d(q, y) < cR \).

Part (a)

If there exists \( x \in V \) such that \( d(x, q) \leq R \) then, argue that we output \( x \) with probability \( 1 - (1 - p_1^b)^h \). On the other hand, also show that the expected number of false positives (i.e., points \( x' \in V \) such that \( d(x', q) > cR \)) that we consider is \( np_2^b \).

Part (b)

Note that since we are checking explicitly for false positives, we never output one – however, we have \( O(1) \) runtime cost for each false positive (to check its distance). Choose \( r \) to ensure that the
expected number of false-positives is 1. Using this choice of \( r \), show that for guarantee we desire for the \((c,R)\)-NN problem, we need to choose \( b = n^\rho \ln(1/\delta) \), where \( \rho = \frac{\ln(1/p_1)}{\ln(1/p_2)} \).

**Problem 3: (More on the Morris’ Counter)**

Recall in class we saw the basic Morris counter, wherein we initiated the counter to 1 when one item arrived, and upon each subsequent arrival, incremented the counter with probability \( 1/2^X \). We also showed that after \( n \) items have arrived, \( \mathbb{E}[2^X] = n + 1 \).

**Part (a)**

Prove that the variance of the counter is given by:

\[
\text{Var}(2^X_n) = \frac{n^2 - n}{2}
\]

Using this, find the probability that the average of \( k \) Morris counters is less than \( n + 1 - \epsilon n \) after \( n \) items have passed.

*Hint: Use induction for \( \mathbb{E}[2^X] \).*

**Part (b)**

Next, suppose we modify the counter as follows: we still initialize counter \( Y \) to 1 when the first item arrives, but on every subsequent arrival, we increment the counter by 1 with probability \( 1/(1+a)^Y \), for some \( a > 0 \). Let \( Y_n \) be the counter-state after \( n \) items have arrived – choose constants \( b, c \) such that \( b \cdot (1+a)^Y_n + c \) is an unbiased estimator for the number of items (i.e., \( \mathbb{E}[b \cdot (1+a)^Y_n + c] = n \)).

**Part (c) (OPTIONAL)**

Now suppose you are restricted to use a single Morris counter, but can choose \( a \) as above. Find the variance of the estimator, and using Chebyshev, find the required \( a \) to ensure that the estimate is within \( n \pm \epsilon n \) with probability at least \( 1 - \delta \). What is the expected storage required by this counter?

**Problem 4: (Dyadic Partitions and the Count-Min Sketch)**

In this problem, we modify the Count-Min sketch to give estimates for range queries and heavy-hitters. For this, we first need an additional definition. For convenience, assume \( n = 2^k \); the dyadic partitions of the set \([n]\) are defined as follows:

\[
\mathcal{I}_0 = \{\{1\}, \{2\}, \ldots, \{n\}\}
\]

\[
\mathcal{I}_1 = \{\{1, 2\}, \{3, 4\}, \ldots, \{n-1, n\}\}
\]

\[
\mathcal{I}_2 = \{\{1, 2, 3, 4\}, \{5, 6, 7, 8\}, \ldots, \{n-3, n-2, n-1, n\}\}
\]

\[
\vdots
\]

\[
\mathcal{I}_k = \{\{1, 2, \ldots, n\}\}
\]
Part (a)

Let \( \mathcal{I} = I_0 \cup I_1 \cup \ldots \cup I_k \) be the set of all dyadic intervals. Show that \( |\mathcal{I}| \leq 2n \). Moreover, show that any interval \([a, b] = \{a, a+1, \ldots, b\}\) can be written as a disjoint union of at most \( 2 \log_2 n \) sets from \( \mathcal{I} \). (For example, for \( n = 16 = 2^4 \), the set \([6, 15]\) can be written as \( \{6\} \cup \{7, 8\} \cup \{9, 10, 11, 12\} \cup \{13, 14\} \cup \{15\} \), which is less than \( 2 \times 4 = 8 \) sets.)

Part (b)

In class, given a stream of \( m \) elements, we saw how to construct a Count-Min sketch for the frequencies of items \( i \in [n] \), and how to use it for point queries (i.e., to estimate \( f_i \) for some \( i \in [n] \)).

We now extend this to range queries – estimating \( F_{[a, b]} = \sum_{i=a}^{b} f_i \) for given \( a, b \).

Note first that the basic Count-Min sketch can be interpreted as constructing a sketch for frequencies of set-membership for the sets in \( I_0 \). We have also seen how to make hash functions for general set-membership (for example, the Bloom filter!) – we can thus extend the Count-Min sketch to include an estimate for the frequencies of all the dyadic intervals. Using this new sketch, show that for a given range query \([a, b]\) , we can use a Count-Min sketch with \( R = \log(1/\delta) \) rows and \( B = 2/\epsilon \) columns to get an estimate \( F_{[a, b]} \) satisfying:

\[
P \left[ F_{[a, b]} < \sum_{i \in [a, b]} f_i + 2m \epsilon \log n \right] \geq 1 - \delta
\]

Part (c)

The \( \phi \)-heavy-hitters (or \( \phi \)-HH) query is defined as follows:

Given stream \( \{x_1, x_2, \ldots, x_m\} \) with \( x_i \in [n] \), and some constant \( \phi \in [0,1] \), we want to output a subset \( L \subset [n] \) such that, with probability at least \( 1 - \delta \), \( L \) contains all \( i \in [n] \) such that \( f_i \geq \phi m \), and moreover, every \( i \in L \) satisfies \( f_i \geq \phi m/2 \).

We now adapt the above sketch for the \( \phi \)-HH problem. First, using the union bound, argue that if we choose \( \delta = \gamma/n \), then we have that for all dyadic intervals \( I \in \mathcal{I} \), we have that the frequency estimate \( F_I \) obeys: \( P \left[ F_I < \sum_{i \in I} f_i + m \epsilon \right] \geq 1 - \delta \). Thus, argue that if we use \( \epsilon < \phi/2 \), then the set of all \( i \in [n] \) such that \( F_{\{i\}} > \phi m \) is a solution to the \( \phi \)-HH problem.

Part (d)

Note though that the brute force way to find all \( i \in [n] \) such that \( F_{\{i\}} > \phi m \) requires \( n \) point queries. Briefly argue how you can use the frequency estimates \( F_I \) for the dyadic intervals to find the same using \( O(\log n/\phi) \) queries.

**Hint:** Consider a binary tree defined by the dyadic intervals, with the root as \( I_{\log n} = \{[n]\} \), and the leaves as \( I_0 = \{\{1\}, \{2\}, \ldots, \{n\}\} \). Argue that for every heavy-hitter node \( i \), every parent node in the tree has \( F_I > \phi m \). Also, at any level \( j \), how many sets \( I \in \mathcal{I}_j \) can have \( F_I > \phi m \)?