1. (6 points) Consider the problem of counting the number of distinct elements in a data stream. Let \( 1 \leq a_1, \ldots, a_n \leq m \) denote the first \( n \) elements in the data stream. Let \( d \) denote the number of distinct elements in \( (a_1, \ldots, a_n) \).

Consider the following algorithm. Algorithm 1 hashes every element in the stream, maintains the \( t \) smallest distinct hash values, and then uses the \( t \)-th smallest hash value to estimate \( d \).

We write \([n]\) for \( \{1, \ldots, n\}\).

Algorithm 1: Estimating the number of distinct elements.

\[
\text{Input : } m \geq 10, n \geq 1, 0 < \epsilon < 1, \text{ and a stream of } n \text{ numbers } 1 \leq a_1, \ldots, a_n \leq m.
\]

\[
\text{Output: an estimation of the number of distinct elements in } (a_1, \ldots, a_n).
\]

Let \( M = m^3 \) and \( t = \frac{40}{\epsilon^2} \).

Suppose \( h \) is a hash function that maps \([m]\) to \([M]\) uniformly at random.

Initialize \( S \leftarrow \emptyset \).

\[
\text{for } i = 1 \text{ to } n \text{ do}
\]

\[
\text{if } h(a_i) \notin S \text{ then}
\]

\[
\begin{align*}
&\text{if } |S| < t \text{ then} \\
&S \leftarrow S \cup \{h(a_i)\}.
&\text{else} \\
&\text{if } h(a_i) \text{ is smaller than the largest number in } S, \text{ replace that number with } h(a_i).
\end{align*}
\]

\[
\text{if } |S| < t \text{ then}
\]

\[
\text{return } |S|.
\]

\[
\text{else}
\]

\[
\text{Let } v \text{ be the largest number in } S.
\]

\[
\text{return } \tilde{d} = \frac{tM}{v}.
\]

(1) Assume that \( h \) can be stored for free and \( h(x) \) can be evaluated in \( O(1) \) time. Show that implementing Algorithm 1 using (balanced) binary search trees requires \( O\left(\frac{\log m}{\epsilon^2}\right) \) bits of space, and each iteration of the for loop runs in time \( O(m \log(1/\epsilon)) \).

Next, we will prove one side of the correctness of Algorithm 1: \( \Pr[\tilde{d} > (1+\epsilon)d] \leq \frac{1}{10} \) (over the randomness in \( h \)).

The event \( \frac{tM}{v} = \tilde{d} > (1+\epsilon)d \) happens iff \( v < \frac{tM}{(1+\epsilon)d} \). In other words, for \( \tilde{d} > (1+\epsilon)d \) to happen, there must be at least \( t \) hash values that are less than \( \frac{tM}{(1+\epsilon)d} \leq (1 - \frac{\epsilon}{2}) \frac{tM}{d} \).
2. (6 points) In this question, we will study the uniqueness of PageRank and how to compute it via iterative methods. Consider an unweighted directed graph \( G = (V, E) \) with \( |V| = n \). We define the transition matrix \( M \in \mathbb{R}^{n \times n} \) of \( G \) as

\[
M_{i,j} = \begin{cases} 
\frac{1}{d(j)} & \text{if there is an edge from } j \text{ to } i, \\
0 & \text{otherwise},
\end{cases}
\]

where \( d(j) \) is the outgoing degree of node \( j \).

Let \( 0 < \alpha < 1 \) be the teleport probability. Let \( \mathbf{1} \in \mathbb{R}^n \) be the all ones vector.

(1) Prove that \( (I - (1 - \alpha)M) \) is a strictly column diagonally dominant matrix. (\( A \) is strictly column diagonally dominant iff \( |A_{i,j}| > \sum_{i \neq j} |A_{i,j}| \) for all \( j \).)

(2) Prove that there is a unique vector \( \mathbf{r}^* \in \mathbb{R}^n \) such that \( \mathbf{r}^* = \alpha \frac{1}{n} \mathbf{1} + (1 - \alpha)M \mathbf{r}^* \).

(You can use the following facts without proving them: all strictly column diagonally dominant matrices are invertible; the inverse of an invertible real matrix is a real matrix.)

(3) For a vector \( x \), we define the \( \ell_1 \)-norm of \( x \) as \( \|x\|_1 = \sum_i |x_i| \).

For a matrix \( A \), we define the \( \ell_1 \)-norm of \( A \) as \( \|A\|_1 = \max_{x \neq 0} \frac{\|Ax\|_1}{\|x\|_1} \).

Prove that \( \|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |A_{i,j}| \).

(4) The PageRank vector \( \mathbf{r}^* \in \mathbb{R}^n \) can be approximated as follows:

- Start with any nonnegative vector \( \mathbf{r}_0 \in \mathbb{R}^n \) with \( \|\mathbf{r}_0\|_1 = 1 \).

- For \( i = 1, \ldots, t \), iteratively compute \( \mathbf{r}_i = \alpha \frac{1}{n} \mathbf{1} + (1 - \alpha)M \mathbf{r}_{i-1} \).

Prove that for some \( t = O \left( \frac{\log(1/\epsilon)}{\alpha} \right) \), after \( t \) iterations we have \( \|\mathbf{r}_t - \mathbf{r}^*\|_1 \leq \epsilon \).

(Hint: You may find the inequality \( \|Ax\|_1 \leq \|A\|_1 \|x\|_1 \) useful.)
3. (2 bonus points) These extra-credit questions are related to Q1 and Q2.

(1) Prove that any deterministic algorithm for computing a \((1 \pm 0.1)\)-approximation of the number of distinct elements in an \(n\)-element data stream must use \(\Omega(n)\) bits of space. (Hint: Suppose Alice has the first \(\frac{n}{2}\) numbers and Bob has the last \(\frac{n}{2}\) numbers. The following claim might be useful.)

\textbf{Claim 1.} For all \(n \geq 1\), there exists a set of bit strings \(S \subseteq \{0, 1\}^n\) such that, for some \(1 \leq b \leq n\):

- Every bit string in \(S\) has exactly \(b\) ones.
- Any two strings in \(S\) have at most \(\frac{b}{10}\) overlapping ones. Formally, for any \(a_1, a_2 \in S\) where \(a_1 \neq a_2\), we have \(|\{1 \leq i \leq n : a_1(i) = a_2(i) = 1\}| \leq \frac{b}{10}\).
- \(|S| \geq 2^{cn}\) for some universal constant \(c > 0\).

(2) Consider a linear system \(Ax = b\) with \(A \in \mathbb{R}^{n \times n}\) and \(b \in \mathbb{R}^n\). Suppose \(A\) is a (symmetric) positive definite matrix. Let \(0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n\) denote the eigenvalues of \(A\). Let \(x^*\) be the solution to \(Ax = b\). Note that for any \(\alpha > 0\), we have \(\alpha Ax^* = \alpha b\), which is equivalent to \(x^* = (I - \alpha A)x^* + \alpha b\).

Consider the following iterative method for solving \(Ax = b\).

- Start with \(x_0 = 0\).
- For \(i = 1, \ldots, t\), iteratively compute \(x_i = (I - \alpha A)x_{i-1} + \alpha b\).

Prove that one can choose the value of \(\alpha\) such that after \(t = O(\kappa \log(1/\epsilon))\) iterations, we have \(\|x_t - x^*\|_2 \leq \epsilon \|x^*\|_2\), where \(\kappa = \frac{\lambda_n}{\lambda_1}\) is the condition number of \(A\).