A: Views and Rewriting [15%]

1. [15%] Find an equivalent rewriting of the query

\[ Q() : -R_1(X_1, X_2), R_2(X_2, X_3), R_3(X_3, X_4), R_4(X_4, X_5), R_5(X_5, X_1), X_1 < X_2 \]

using the following views:

\[
V_1(X_1, X_3) : -R(X_1, X_2), R(X_2, X_3) \\
V_2(X_1, X_3) : -R(X_3, X_4), R(X_4, X_5), R(X_5, X_1)
\]

Solution. The rewriting is \( Q() : -V_1(X, Y), V_2(Y, X), X < Y \).

B: Parallel Query Processing [25%]

1. The task is to compute the query \( Q(x_1, x_2, x_3) : -R_1(x_1), R_2(x_2), R_3(x_3) \) in parallel using \( p \) machines. The query is a cartesian product of three unary relations. Let the sizes of the relations be \( N_1, N_2, N_3 \) respectively.

(a) [10%] Suppose that we compute \( Q \) in a single round by distributing the data once (using the HyperCube technique). What is the minimum amount of data that each machine can receive? What is the total communication?

Solution. The minimum amount of data received is \( L = 3(N_1 N_2 N_3 / p)^{1/3} \). The total communication is \( L \cdot p \).

(b) [10%] Suppose that we compute \( Q \) in two rounds. In the first round we compute the cartesian product of \( R_1, R_2 \), and the second round the cartesian product of the intermediate result with \( R_3 \). What is the minimum amount of data that each machine can receive in each round? What is the total communication across both rounds?

Solution. For the first round, the minimum amount of data is \( L_1 = 2(N_1 N_2 / p)^{1/2} \). For the second round, it is \( L_2 = 2(N_1 N_2 N_3 / p)^{1/2} \). The total communication is \( (L_1 + L_2) \cdot p \).
Which of the above two strategies achieves the best load per machine? Which one the best total communication?

Solution. The one-round strategy is almost always better, unless the sizes \( N_1, N_2, N_3 \) are very very small (smaller than \( p \)).

C: Data Streaming [30%]

1. [15%] In reservoir sampling, we want to produce a uniform sample of size \( k \) from a stream of unknown size. The algorithm works by placing the \( i \)-th item in the reservoir (of size \( k \)) with probability \( k/i \) (all the first \( k \) items go in the reservoir initially). Show that at any point where we have seen \( n \) total items, the probability of each item being in the reservoir is \( k/n \).

Solution. The proof is by induction. For \( k = n \), the first \( n \) items are in the reservoir with probability \( k/n = 1 \). Consider the state after we have seen the \( n \)-th item, and let us compute the probability of an element \( a \) seen at the \( i \)-th position being still in the reservoir. Initially it is included with probability \( k/i \). After we see the \((i + 1)\)-th element, it is removed with probability \( 1/k \cdot \frac{k}{i+1} = \frac{1}{i+1} \), so it remains with probability \( \frac{i}{i+1} \). After the \((i + 2)\)-th element, it remains with probability \( \frac{i+1}{i+2} \), and so on. Multiplying these probabilities gives us \( \frac{k}{i} \cdot \frac{i}{i+1} \cdot \frac{i+1}{i+2} \cdots = \frac{k}{n} \).

2. [15%] Recall the application of the Misra-Gries algorithm for the case where we want to find the top-\( k \) most frequent elements. For any element \( j \), let \( f_j \) be its actual frequency in the stream, and \( \hat{f}_j \) its estimated frequency. We showed in class that \( f_j - m/k \leq \hat{f}_j \leq f_j \), where \( m \) is the total length of the stream.

Show that this bound can be improved to \( f_j - m - \hat{m} \leq \hat{f}_j \leq f_j \), where \( \hat{m} \) is the sum of the estimated frequencies \( \hat{f}_j \).

Solution. The argument for \( \hat{f}_j \leq f_j \) is the same as before. To show that \( f_j - m - \hat{m} \leq \hat{f}_j \), it suffices to see that every time we are reducing the counter (thus underestimating \( f_j \)), we reduce a total of \( k \) counters. The total reduction can be at most \( m - \hat{m} \).
1. [15%] Consider a tuple-independent probabilistic database with three relations: \( R(A, B), S(A, C), T(A, D) \). Suppose we want to answer the boolean query

\[
Q() : \neg R(x, y), S(x, 'a'), T(x, w)
\]

Describe a safe plan for computing the probability of \( Q \) if one exists; otherwise, explain why it is not possible to obtain one.

**Solution.** A safe plan is as follows: First we compute \( \Pi_A(R) \), \( \Pi_A(T) \), and then \( \Pi_A(\sigma_{C='a'}(S)) \). Then we join the 3 intermediate results in any order.

2. [15%] Consider an inconsistent database, where the integrity constraints are primary keys. The database consists of two relations \( R(A, B) \) and \( S(B, C, D) \). We want to obtain the consistent answers for the query \( Q(x) : \neg R(x, y), S(y, z, 'a'), \) which can be computed through a first-order rewriting. Write a query in SQL that computes the consistent answers for \( Q \).

**Solution.** The query must output the \( x \)'s for which for every corresponding \( y \), there exists a tuple in \( S \) with \( S.D = 'a' \).

---

**Deliverables**

Submit a single PDF document using the Dropbox in `learn@uw` (Homework 3). It is strongly suggested to use LaTeX to write your solution.