CS 630 – Fall 2024 – Lab 8
Nov 6, 2024

Problem 1 *Multiple Bloom filters*

1. Suppose we have two Bloom filters A and B with the same number of bits and using the exact same hash functions. A is representing the set X (the elements $x \in X$ are stored in A) and B is representing Y. Answer the following questions

   (a) Let $C = A \wedge B$ be the Bloom filter formed by computing the bitwise Boolean *and* operation between A and B. Prove that C is not the same as the Bloom filter that would be constructed by adding the elements of the set $X \cap Y$ one at a time.

   > **Solution:**
   >
   > Suppose that X and Y have no element in common, $X \cap Y = \emptyset$. Then the Bloom filter we get by adding elements in the intersection one at a time would be empty. However, if there are elements $x \in X$ and $y \in Y$ that share at least one of their hash values, i.e. there is such hash function h_i that $h_i(x) = h_i(y)$, then the corresponding bit in C would be 1.

   (b) Does C correctly represent the set $X \cap Y$, in the sense that it gives a positive answer for membership queries of all elements in this set? Explain why or why not.

   > **Solution:**
   > This is True. If there is an element $z \in X \cap Y$ then the corresponding bits are set to 1 both in A and B, hence those bits are also ones in the bitwise and of the two.

   (c) Let $D = A \vee B$ be the Bloom filter formed by computing the bitwise Boolean *or* (inclusive) operation between A and B. Show that D does represent the union of the sets X and Y.

   > **Solution:**
   > Let $z \in X \cup Y$. wlog we may assume that z is in X. Every hash bit corresponding to z is one in A. Since we use the Boolean or this implies that the corresponding bits are also one in D.

Problem 2 *Verifying Polynomial Identities*

Let $P(x)$ and $Q(x)$ be polynomials over a finite field F of size $|F|$, each with total degree at most d. You have access to P and Q only as black boxes; that is, you can evaluate them at any point but do not know their explicit forms.

1. Describe a randomized algorithm to test whether $P \equiv Q$ (i.e., whether P and Q represent the same polynomial).
2. Prove that if $P \not\equiv Q$, the probability that the algorithm incorrectly concludes $P \equiv Q$ is at most $\frac{d}{|F|}$.

Solution:

1. **Algorithm Description:**

   To test whether $P \equiv Q$, we can use the following randomized algorithm:

   (a) Define $R(x) = P(x) - Q(x)$.

   (b) For $i = 1$ to k:

      i. Randomly select elements $a^{(i)}$ independently and uniformly from F.

      ii. Evaluate $R(a^{(i)})$.

      iii. If $R(a^{(i)}) \neq 0$, conclude that $P \not\equiv Q$ and terminate the algorithm.

   (c) If $R(a^{(i)}) = 0$ for all $i = 1$ to k, conclude that $P \equiv Q$.

2. **Error Probability Analysis:**

   When $P \not\equiv Q$, the polynomial R is a non-zero polynomial of total degree at most d.

   We aim to bound the probability that R evaluates to zero at a randomly chosen point $a \in F$.

   Fundamental theorem of algebra (Gauss): A single variable non-zero polynomial of degree d over F cannot have more than d zeros in F.

   Probability Calculation:

   The total number of possible inputs is $|F|$. Therefore, the probability that a randomly chosen $\mathbf{a} \in F$ is a root of R is at most:

   $$\Pr[R(\mathbf{a}) = 0] \leq \frac{d}{|F|}$$

   Error Probability over k Trials:

Since each trial is independent, the probability that R evaluates to zero in all k trials when $R \not\equiv 0$ is at most:

$$\left(\frac{d}{|F|}\right)^k.$$

Thus, the probability that the algorithm incorrectly concludes $P \equiv Q$ is at most $\left(\frac{d}{|F|}\right)^k$.

Problem 3 *Universal Hashing*

For hashing, we first define the universe U from which all keys would come from (e.g. U could be the set of strings from ascii characters of length at most 10). Then a hash function $h : U \to \{1, 2, \ldots, M\}$ is a functions that maps the keys to a specific index of an M-sized array.

1. Prove the following claim, for any hash function h if $|U| \geq (N-1)M + 1$ there exists a set S of N elements that all hash to the same location

   Solution: We focus on the contrapositive. If every location has at most $N - 1$ elements of U hashing to it, then U could only have size at most $M(N - 1)$. (Uses pigeon hole principle)

2. A hash function $h : U \to \{1, \ldots, M\}$ is universal if for all $x \neq y$ we have

   $$P(h(x) = h(y)) \leq 1/M$$

   Show that for any set $S \subset U$ of size N and for any x the expected number of collisions between x and any element in S is at most N/M

   Solution: We fix some random x, then for any $y \in S$ we define a Bernoulli indicator random variable C_y where $C_y = 1$ if x and y collides and 0 otherwise.

   Because h is universal then we have

   $$E[C_y] = Pr(C_y = 1) = Pr(h(x) = h(y)) \leq 1/M$$

   So the random variable $C = \sum_{y \in S} C_y$ denote the total number of collisions between x and any element in S.

So by linearity of expectation we get

$$E[C] = \sum_{y \in S} E[C_y] \leq N/M$$

3. If $h : U \to \{1, 2, \ldots, M\}$ is universal and assume that it has a constant time complexity. Show that when working with a specific set S of size M, any sequence of L insert, lookup and delete operations has an expected total cost of $O(L)$

Solution: Since S of size M then the expected number of collision is $E[C] \leq M/M = 1$. Let X_i be a random variable denoting the cost of the $i - th$ operation. The cost of the operation would be proportional with the number of collisions. Which means

$$E[X_i] = O(E[C]) = O(1)$$

Thus by linearity of expectation the expected total cost is x

$$E[X] = \sum_{i=1}^{L} E[X_i] = O(L)$$