1. Background

In the previous project, we considered membership dictionaries that support insertions and membership-queries. In this project, we consider an application of a dictionary for supporting approximate membership queries. What does that mean?

Consider a universe U and a dataset $D \subseteq U$. Let $\varepsilon \in [0, 1)$. A data structure supports approximate membership queries with parameter ε if, for every $x \in U$, the response exists for query(x) satisfies:

$$exists = \begin{cases} 1 & if\ \mathcal{X} \in \mathcal{D} \\ b \in \{0,1\} & if\ \mathcal{X} \notin \mathcal{D} \end{cases}$$

Now, the answer b (when $x \notin D$) is 1 with probability at most ε. We refer to the event that exists = 1 for $x \notin D$ as a *false-positive*. ‡

Where does the probability come from? The data structure uses a random function. Say, it randomly chooses a function h from a family H of functions (that contains H functions). Then, for every $D \subset U$ (such that $|D| \leq n$) and every $x \in U$ the following holds: at most εH functions from H will cause a false-positive for query(x). Functions $h \in H$ are called *hash* functions.

1.1. **Hashing.** We would like the hash function to be a random function. That is, of course, not possible, because we would need to store its "truth-table" which is too big. Instead, we resort to functions that "look" random. What does that mean? Let H denote a family of functions $h : U \to A$.

(1) Uniform distribution of x. The first thing we want is that for every $x \in U$ and every $a \in A$, that

$$|h \in H : h(x) = a| = \frac{H}{|A|}$$

(2) Pairwise independence. The second thing that we want is that every $x_1 \neq x_2 \in U$ and every $a_1, a_2 \in A$, that

$$|\{h \in H : h(x_1) = a_1 \wedge h(x_2) = a_2\}| = \frac{H}{|A|^2}$$

Namely, all pairs $(h(x_1), h(x_2))$ are equally likely.

Such a family of hash functions is called 2-independent. Do such families of functions exist? Are they easy to compute? The answer is yes. In Section 1.3, we describe such a family that is very easy to implement by a digital circuit.

1.2. **Dictionary + 2-Independent Hashing ⇒ Approximate Membership.** Assume that we have family of 2-independent hash functions. How can we use it for approximate membership?

The idea is very simple. Randomly pick a function $h \in H$. Use a dictionary that stores the dataset $\{h(y) \mid y \in D\}$. (We do not store y, instead we store $h(y) \in A$.) Now consider an element $x \notin D$. How can we bound the probability of a false-positive for query(x)?

A false-positive occurs (for query(x) when $x \notin D$) if and only if there exists an element $y \in D$ such that $h(x) = h(y)$. For a specific y, 2-independenceimplies that the probability that $h(x) = h(y)$ equals $1/|A|$. Summing up over all the elements $y \in D$, we conclude that a false positive for x occurs with probability at most $n/|A|$. Hence, we can bound the false-positive probability by ε if $\varepsilon \leq n/|A|$. In other words, we need A (the range of the hash functions) to satisfy

$$|A| \geq \frac{n}{\varepsilon}.$$

If we think of A as a set of strings, i.e., $A = \{0,1\}^a$, then $a \geq \log n + \log(1/\varepsilon)$ (logarithms base 2). §

1.3. **Tabulation Hashing.** Consider an n-bit string $key \in \{0,1\}^n$ split into $£$-bit blocks such that

$$block_i = key \quad (i+1)£ - 1 : i£$$