Tutorial 10

Exercise 1 (4-independence of tabulation hashing)
Show that tabulation hashing is not 4-independent (if we use at least two tables).
Hint: *Try to find four inputs such that the hashes of the first three uniquely determine the hash of the fourth.*

Theorem. Tabulation hashing is 3-independent

Exercise 2 (We will prove this theorem)
Prove the previous theorem using the following procedure. Let $a, b, c \in \mathbb{Z}_2^\ell, x \neq y \neq z \neq x \in \mathbb{Z}_2^w$, and use tabulation hashing with d parts. Then we want to show that $\Pr_{h\in\mathcal{H}}[h(x) = a \wedge h(y) = b \wedge h(z) = c] \leq \frac{1}{m^3}$.

a) First, realize that if we have only one part, and thus one table, the claim is trivial.

   Now assume we have at least two parts. Since x, y, z are distinct, they must differ (pairwise) in at least one part.

b) Start with the case where there exists a part i such that x^i, y^i, z^i are all different. Let the other tables, except for table T_i, be arbitrarily chosen. With what probability can we choose the function for table T_i so that $h(x) = a, h(y) = b, h(z) = c$?

c) Otherwise, there exist (WLOG) parts i, j such that $z^i = x^i \neq y^i$ and $y^j = x^j \neq z^j$. Then we have the following system of equations, where v_x, v_y, v_z are the XORed results from the other tables:

$$T_i[x^i] \oplus T_j[x^j] \oplus v_x = a$$
$$T_i[y^i] \oplus T_j[y^j] \oplus v_y = b$$
$$T_i[z^i] \oplus T_j[z^j] \oplus v_z = c$$

   Again, suppose that v_x, v_y, v_z are already known. With what probability will the randomly chosen tables T_i, T_j satisfy this system of equations?

d) Realize that this is sufficient.

Exercise 3 (Rolling hash is d-universal)
For a prime p and vector length d, we define the hash function family $\mathcal{R} = \{h_a \colon a \in \mathbb{Z}_p\}$, where $h_a(x) = \sum_{i=0}^{d-1} x_{i+1}a^i$ and everything is computed modulo p (so we are using the field $\mathbb{Z}_p$). (We consider $x \in \mathbb{Z}_p^d$, vectors are indexed starting with one.)
Prove that this family is $(d-1)$-universal.
Hint: *The Fundamental Theorem of Algebra: every polynomial of degree d has at most d roots – try to find one.*

Exercise 4 (Finding a needle in a text)
Design an algorithm for finding all occurrences of a substring x of length n in a text T of length m using hashing, which runs in expected time $\mathcal{O}(n + m + k \cdot n)$, where k is the number of occurrences of x in T.

Bonus exercises

Exercise 5 (FKS (Fredman, Komlós, Szemerédi))
We will demonstrate the construction of a (static) collision-free hash table for a subset S of size n of a universe $\mathcal{U}$. You might have encountered a construction that required $\Omega(n^2)$ memory (more precisely, memory cells). We will manage this with a linear number of memory cells (assuming we can have a truly random hash function, which we can construct and sample in constant time, and store in constant space)[1].
The process of building the table will proceed as follows: we will build two levels. In the first level, we use a truly random hash function f to divide the elements of S into buckets $B_1, \ldots, B_n$ (and denote $b_i := |B_i|$). In the second level, we build a collision-free table using a construction where for each bucket B_i, we create a table of size $2b_i^2$ for b_i elements, and we randomly choose a suitable hash function until there are no collisions.

[1] The same can be done with a reasonably universal function; this is just for simplicity.

First level: In constant time, we choose a random hash function $f : \mathcal{U} \to [n]$, and use it to divide S into buckets. We repeat this until the condition $\sum_{i=1}^{n} b_i^2 \leq \beta n$ holds, with $\beta = 4$.
We want to show that this step will, on average, be repeated at most twice. Let C denote the number of collisions.

a) Determine $\mathbb{E}[C]$.

b) Determine C in terms of b_i.

c) Based on the two previous values, determine $\mathbb{E}[\sum_{i=1}^{n} b_i^2]$.

d) Apply Markov's inequality to the random variable $X = \sum_{i=1}^{n} b_i^2$ with a suitable value to get the desired result. (Also, the expected value of a geometric distribution will come in handy.)

Second level: In the second level, for each $i \in [n]$, we choose a universal hash function $g_i : \mathcal{U} \to [\alpha b_i^2]$ for $\alpha = 2$. We repeat this until it is injective (collision-free) for the elements in bucket B_i.
Let C_x denote the number of collisions of key $x \in B_i$ at the second level.

a) Formulate an upper bound on $\mathbb{E}[C_x]$.

b) Use Markov's inequality and the union bound to upper bound the probability that there exists an element with at least one collision.

c) How many times will we have to repeat the process? ([Insert your favorite note about the expected value of a geometric distribution here.])

Useful notions

Proposition (Union bound)**.** For elements A_1, A_2, we have $\Pr[A_1 \cup A_2] \leq \Pr[A_1] + \Pr[A_2]$.

Definition (c-universal function system)**.** A system $\mathcal{H}$ of functions $h : \mathcal{U} \to [m]$ is c-universal for $c > 0$, if for all $x \neq y$, it holds that $\Pr_{h \in \mathcal{H}}[h(x) = h(y)] \leq \frac{c}{m}$.
A system $\mathcal{H}$ is universal, if it is c-universal for some $c > 0$.

Definition (Tabulation hashing)**.** Suppose we want to hash n-bit strings into m-bit strings, where $n = k \cdot \ell$. We decompose a string $x \in 0,1^n$ into k parts of length ℓ, which we denote by x^i. Thus, we can write $x = x^1 x^2 \dots x^k$. Then, the generation of our hash function $h : 0,1^n \to 0,1^m$ proceeds by selecting k functions $T_i : 0,1^\ell \to 0,1^m$ uniformly at random (these are represented by a table – hence the name *tabulation* hashing). We then evaluate $h(x) = \bigoplus_{i=1}^{k} T_i(x^i) = T_1(x^1) \oplus T_2(x^2) \oplus \dots \oplus T_k(x^k)$ where $\oplus$ denotes bitwise XOR.

Theorem (Markov inequality)**.** Let X be a nonnegative random variable. Then $\forall \varepsilon > 0$ we have $P[X \geq \varepsilon] \leq \frac{\mathbb{E}[X]}{\varepsilon}$.
Equivalently, for any $d \geq 1$, $P[X \geq d \cdot \mathbb{E}[X]] \leq \frac{1}{d}$.