# Tutorial 9

Data Structures 1, 25. 4. 2025 https://iuuk.mff.cuni.cz/~chmel/2425/ds1en/

**Exercise 1** (Independence and universality)
Prove the following:

- if a hashing system is $(k, c)$-independent, it is also $(k-1, c)$-independent (for $k \geq 2$),
- if a hashing system is $(2, c)$-independent, it is also $c$-universal.

**Exercise 2** (Truly practical systems)
Let us consider the hashing function system $\mathcal{H}_1 = \{\mathrm{id}\}$ that contains just one function, the identity that maps $x$ to $x$. Is $\mathcal{H}_1$ $c$-universal for some $c$? Is $\mathcal{H}_1$ $(k, c)$-independent for some $k$ and $c$?
Next, consider the system $\mathcal{H}_2 = \{h_a(x) = a \colon a \in [m]\}$. Prove that this system is (1,1)-independent. Next, show that $\mathcal{H}_2$ is neither $(2, c)$-independent nor $c$-universal for any $c$.

**Exercise 3** (Modulo of a universal system does not need to be universal)
Show that, if we have a universal system of hash functions $\mathcal{H}$, then the system $\mathcal{H}'$, where each function is computed modulo $m$, does not have to be universal. Formally: Show that for every $c$ and $m > c$, there exists a universe $\mathcal{U}$ and a system $\mathcal{H}$ from $\mathcal{U}$ to $\mathcal{U}$ such that $\mathcal{H}$ is universal but $\mathcal{H}'$ is not $c$-universal.

**Exercise 4** (A bad version of cuckoo hashing)
Why is the following insert implementation for cuckoo hashing problematic? For *this exercise*, implementation and rehashing conditions are not the issue. We care mostly about the use of the hash functions.

```
for i=1 to n
  if T[h1(x)] is empty
    T[h1(x)] = x
    return
  swap(T[h1(x)], x)
  if T[h2(x)] is empty
    T[h2(x)] = x
    return
  swap(T[h2(x)], x)
```

**Exercise 5** (Rehashing)
A simple implementation of rehashing for cuckoo hashing is that we ibnsert all values into an auxiliary array and then insert them one-by-one. Create an implementation of rehashing that does not require this auxiliary array.
(Note that during a rehash, we can start rehashing recursively.)

---

**Bonus exercises**

**Exercise 6** (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** ($k$-independent function system)**.** A system $\mathcal{H}$ of functions $h : \mathcal{U} \to [m]$ is $(k, c)$-independent for some $k \geq 1, c > 0$, if $\Pr_{h \in \mathcal{H}}[h(x_1) = a_1 \wedge \ldots \wedge h(x_k) = a_k] \leq \frac{c}{m^k}$ for any pairwise distinct $x_1, \ldots, x_k$ and any not necessarily distinct $a_1, \ldots, a_k$.
A system $\mathcal{H}$ is $k$-independent, if it is $(k, c)$-independent for some fixed constant $c$.

**Definition** (Cuckoo hashing)**.** We have two hashing functions $f, g : \mathcal{U} \to [m]$ chosen uniformly at random from a hash function system, and one array $T$ of size $m$. Our goal is to maintain the invariant, that if $x$ is stored in the table, it is in one of the two "nests" $f(x)$ or $g(x)$.
Lookup simply looks at the two cells and then says yes/no depending on what it sees.
Insert works as follows: if $f(x)$ is empty, then we insert $x$ there. Otherwise, the cell $f(x)$ is full. Then, we take the element $x_1$ stored there out, and put $x$ in instead. Now, we have to store $x_1$, and we put it into the "nest" $f(x_1), g(x_1)$ in which it was *not* originally stored – which is the nest differing from $f(x)$. If it was not empty, we might get an $x_2$. This could go on for a while, so if we cannot stop these nest changes within $\lceil 6 \log m \rceil$ or $\lceil 6 \log n \rceil$ steps, we decide it is not worth trying to push the elements around, and we rehash the whole table by generating new functions $f, g$ and rehashing all the stored elements.

**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 $d \geq 1$, $P[X \geq d \cdot \mathbb{E}[X]] \leq \frac{1}{d}$.