Na přednášce jste viděli, že pomocí zcela náhodné hashovací funkce lze sestrojit (statickou) tabulku pro podmnožinu S univerza U, která nemá žádné kolize. Nevýhodou je, že spotřebovává paměť $\Omega(n^2)$. Teď si ukážeme, jak snížit paměť[1] na $\mathcal{O}(n)$. Předpokládáme, že takovou funkci dokážeme samplovat v konstantním čase a že si ji dokážeme pamatovat v konstantním prostoru.

High-level pohled. Uděláme si dvouúrovňovou tabulku. V první úrovni si zcela náhodnou hashovací funkcí f rozhodíme prvky do *kyblíků* $B_1, \ldots, B_n$ (některé mohou být prázdné). Označíme $b_i = |B_i|$.

Ve druhé úrovni si v každém kyblíku uděláme bezkolizní tabulku pomocí konstrukce z přednášky.

První úroveň. V konstantním čase si vybereme zcela náhodnou hashovací funkci $f: U \to [n]$. Tou rozházíme S do kyblíků. To opakujeme, dokud neplatí, že $\sum_{i=1}^{n} b_i^2 \leq \beta n$ pro $\beta = 4$.

1. Chceme ukázat, že tento krok budeme opakovat nejvýše dvakrát (ve střední hodnotě). Bude se hodit znát *Markovovu nerovnost*: Nechť X náhodná veličina nabývající kladných hodnot. Pak $\Pr[X \geq a] \leq \frac{\mathbb{E}[X]}{a}$ pro $a > 0$.

   Nechť C značí počet kolizí.

   a) Určete $\mathbb{E}[C]$.

   b) Určete C v závislosti na b_i.

   c) Určete $\mathbb{E}[\sum_{i=1}^{n} b_i]$ na základě předchozích dvou hodnot.

   d) Aplikujte Markovovu nerovnost na náhodnou veličinu $X = \sum_{i=1}^{n} b_i^2$ s vhodnou konstantou, abyste dostali požadovaný výsledek (bude se opět hodit střední hodnota geometrického rozdělení).

Druhá úroveň. Pro každé $i \in [n]$ volíme v i-tém kyblíku univerzální hashovací funkci $g_i: U \to [\alpha b_i^2]$ pro $\alpha = 2$. Opakujeme, dokud není prostá pro prvky, které dopadly do B_i.

2. Nechť τ_i měří počet opakování volby g_i, dokud nedostaneme prostou funkci pro prvky, které dopadly do B_i. Chceme ukázat, že $\mathbb{E}[\tau_i] \leq 2$.

   Nechť C_x měří počet kolizí klíče $x \in B_i$.

   a) Shora odhadněte $\mathbb{E}[C_x]$.

   b) Použitím Markovovy nerovnosti a union boundu určete shora odhadněte pravděpodobnost, že existuje prvek, který má aspoň jednu kolizi.

   c) Dokončete úlohu.

3. Určete časovou složitost pro konstrukci obou úrovní. (Každá by měla být lineární)

4. Jak implementovat konstantní lookup pomocí téhle věci?

Poznámka. Tuto konstrukci můžete najít pod názvem *FKS*[2] *perfect hashing.*

5. Ukažte, že dokonce i systém $\{h_a(x) = ((ax) \bmod p) \bmod m; a \in [1, p], m \leq p, p \text{ je prvočíslo}\}$, tedy bez aditivního členu, je universální. Bude to platit, když povolíme $a = 0$?

6. Uvažme systém hashovacích funkcí $\{h_{a,b}(x) = ((ax + b) \bmod p) \bmod m; a, b \in [p], m \leq p, p \text{ je prvočíslo}\}$. Ukažte, že není 3-nezávislý.

7. Ukažte, že tabulkové hashování je 3-nezávislé.

[1] Měřenou v počtu paměťových buněk. Například předpokládáme, že každá buňka je $\mathcal{O}(\log n)$-bitová a umíme s ní všemožné konstantní bitové operace.

[2] Podle autorů Fredman, Komlós, Szemerédi.