7. cvičení

Datové struktury I, 20. 11. 2025 https://iuuk.mff.cuni.cz/~chmel/2526/ds1/

Věta. Tabulkové hashování je 2-nezávislé.

Důkaz. Na tabuli. ⊞

Věta. Tabulkové hashování je dokonce 3-nezávislé.

Úloha 1 (Tuhle větu si dokážeme)
Dokažte předcházející větu s následujícím postupem. Mějme $a, b, c \in \mathbb{Z}_2^\ell, x \neq y \neq z \neq x \in \mathbb{Z}_2^w$, a používejme tabulkové hashování s d částmi. Pak chceme ukázat, že $\Pr_{h\in\mathcal{H}}[h(x) = a \wedge h(y) = b \wedge h(z) = c] \leq \frac{1}{m^3}$.

a) Prvně si uvědomme, že pokud máme jen jednu část, a tedy jednu tabulku, tvrzení je triviální.

   Dále mějme alespoň dvě části. Protože x, y, z jsou různé, musí se (po dvou) lišit alespoň v jedné části.

b) Začneme s případem, kdy existuje část i, že x^i, y^i, z^i jsou všechny různé. Mějme jakkoliv zvolené ostatní tabulky, kromě tabulky T_i. S jakou pravděpodobností můžeme zvolit funkci pro tabulku T_i tak, že $h(x) = a, h(y) = b, h(z) = c$?

c) Jinak existují (BÚNO) části i, j takové, že $z^i = x^i \neq y^i$ a $y^j = x^j \neq z^j$. Potom máme následující soustavu rovnic, kde v_x, v_y, v_z jsou vyXORované výsledky z ostatních tabulek:

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

   Opět si představme, že v_x, v_y, v_z už známe. S jakou pravděpodobností budou náhodně volené tabulky T_i, T_j splňovat tuto soustavu rovnic?

d) Uvědomte si, že toto stačí.

Úloha 2 (4-nezávislost tabulkového hashování)
Ukažte, že tabulkové hashování není 4-nezávislé (pokud používáme aspoň dvě tabulky).
Hint: *Zkuste najít nějakou čtveřici vstupů takových, že hashe prvních tří jedenzačné určují hash toho čtvrtého.*

Úloha 3 (Špatná verze kukačky)
Proč je následující implementace insertu pro kukačkové hashování problematická? (Implementaci a podmínky pro rehashování pro tento příklad meteme pod koberec.)

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

Úloha 4 (Rehashujeme)
Jednoduchá implementace rehashe u kukačkového hashování je, že si všechny hodnoty vložíme do pomocného pole, a potom je po jednom insertujeme. Vymyslete implementaci rehashe, která pomocné pole nepotřebuje. (Pozor na to, že během rehashe můžeme znovu začít s rehashem.)

Úloha 5 (Vyloženě praktické systémy)
Uvažme systém funkcí $\mathcal{H}_1 = \{\mathrm{id}\}$, který obsahuje jedinou funkci, jež zobrazí x na x. Je $\mathcal{H}_1$ c-univerzální pro nějaké c? Je $\mathcal{H}_1$ (k, c)-nezávislý pro nějaká k a c?
Dále uvažme systém $\mathcal{H}_2 = \{h_a(x) = a \colon a \in [m]\}$. Dokažte, že tento systém je (1,1)-nezávislý. Dále ukažte, že $\mathcal{H}_2$ není $(2, c)$-nezávislý ani c-univerzální pro žádné c.

Úloha 6 (Modulo univerzálního systému nemusí být univerzální)
Ukažte, že pokud máme univerzální systém hashovacích funkcí $\mathcal{H}$, pak systém $\mathcal{H}'$, kde ke každé fci navíc přidáme modulo m, už nemusí být univerzální. Formálně: Dokažte, že pro každé c a $m > c$ existuje univerzum $\mathcal{U}$ a systém $\mathcal{H}$ z $\mathcal{U}$ do $\mathcal{U}$ tak, že $\mathcal{H}$ je univerzální, ale $\mathcal{H}'$ už není c-univerzální.

Užitečné pojmy

Definice (c-univerzální systém fcí)**.** Systém $\mathcal{H}$ funkcí $h : \mathcal{U} \to [m]$ je c-univerzální pro $c > 0$, pokud pro všechna $x \neq y$ platí $\Pr_{h\in\mathcal{H}}[h(x) = h(y)] \leq \frac{c}{m}$.
Systém $\mathcal{H}$ je univerzální, pokud je c-univerzální pro nějaké $c > 0$.

Definice (k-nezávislý systém fcí)**.** Systém $\mathcal{H}$ funkcí $h : \mathcal{U} \to [m]$ je (k, c)-nezávislý pro nějaká $k \geq 1, c > 0$, pokud $\Pr_{h\in\mathcal{H}}[h(x_1) = a_1 \wedge \ldots \wedge h(x_k) = a_k] \leq \frac{c}{m^k}$ pro libovolná $x_1, \ldots, x_k$ různá, $a_1, \ldots, a_k$ ne nutně různá.
Systém $\mathcal{H}$ je k-nezávislý, pokud je (k, c)-nezávislý pro nějakou nezávislou konstantu c.

Definice (Tabulkové hashování)**.** Představme si, že chceme zahashovat n-bitové řetízky do m-bitových řetízků, kde $n = k \cdot \ell$. Řetízek $x \in \{0, 1\}^n$ pak rozložíme do k částí délky ℓ, které značíme x^i. Můžeme tedy psát $x = x^1x^2 \ldots x^k$. Pak generování naší hashovací funkce $h : \{0, 1\}^n \to \{0, 1\}^m$ vypadá tak, že vybereme uniformně náhodně k funkcí $T_i : \{0, 1\}^\ell \to \{0, 1\}^m$ (tyto reprezentujeme tabulkou, proto tabulkové hashování). Vyhodnocujeme pak $h(x) = \bigoplus_{i=1}^{k} T_i(x^i) = T_1(x^1) \oplus T_2(x^2) \oplus \cdots \oplus T_k(x^k)$, kde $\oplus$ značí XOR (po jednotlivých bitech).