9. cvičení

Datové struktury I, 2. 12. 2024 https://iuuk.mff.cuni.cz/~chmel/2425/ds1/

Úloha 1 (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 2 (4-nezávislost tabulkového hashování)
Ukažte, že tabulkové hashování není 4-nezávislé (pokud používáme aspoň dvě tabulky).
Hint: *Zkuste najít nějakou čtveřici vstupů takovýchž že hashe prvních tří jednoznačně určují hash toho čtvrtého.*

Věta. Tabulkové hashování je 3-nezávislé.

Úloha 3 (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 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.)

Užitečné definice

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 $x = x^1 x^2 \dots x^k$. 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).

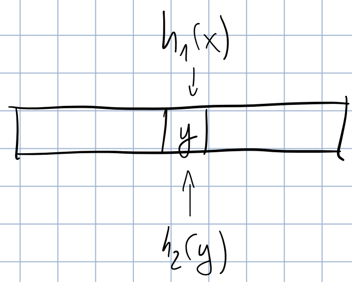
Definice (Kukačkové hashování)**.** V každém okamžiku máme dvě hashovací funkce $f, g : \mathcal{U} \to [m]$ volené uniformně náhodně z nějakého systému hashovacích funkcí a jedno pole velikosti m. Naším cílem je, že každý prvek x, který je zahashovaný, se vyskytuje v jednom ze dvou „hnízd“ $f(x)$ nebo $g(x)$.
Lookup se podívá na tato dvě místa, a podle výsledku buď řekne, zda se tam prvek nachází, nebo ne.
Insert probíhá následovně: pokud je jedno z hnízd $f(x)$ nebo $g(x)$ volné, usadíme x do volného místa. Jinak vybereme jedno z plných míst (řekněme $f(x)$), x do něj vložíme, a vyjmeme prvek x_1, který byl v tomto hnízdě původně uložený. Teď musíme uložit x_1, a to vložíme do toho hnízda $f(x_1), g(x_1)$, ze kterého jsme jej *ne*vyjmuli – takže jej dáme do druhého hnízda než bylo $f(x)$. Takhle můžeme nějakou dobu pokračovat, dokud nenajdeme prázdné místečko, nebo dokud nedojde k tomu, že už takhle přesouváme prvky příliš dlouho (řekněme $6 \log(m)$ nebo $6 \log(n)$, kde m je počet hnízd/přihrádek a n je počet uskladňovaných prvků). Potom se na tento pokus o vložení vykašleme, a začneme znovu s tím, že si vygenerujeme nové funkce f a g, a všechny prvky v našem poli přehashujeme.

Úloha 1 (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)
```

→ tady jsem vytáhl „y"

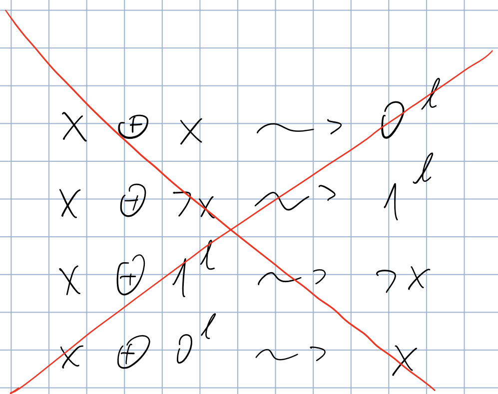
→ tady chci $T[h_2(y)]$



→ není tam $\lceil 6 \cdot \log n \rceil$, chybí tam rehash,

Nemůžu rovnou zhashovat $h_2(x)$, protože původně na stejné pozici prvek „y" byl zahashovaný pomocí právě h_2, takže bych znova prohodil y a x, tak skončím tam kde jsem začal.

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



a	0	0	1	1
b	0	1	0	1
⊕	0	1	1	0

→ Když tohle všechno zXORuju, musím dostat 0.

Pokud budu mít výsledky hashů prvních tří prvků, vím jak musí vypadat 4. hash, abych celkem dal XOR do 0.

0 0	$T_1[0] \oplus T_2[0]$
0 1	$T_1[0] \oplus T_2[1]$
1 0	$T_1[1] \oplus T_2[0]$
1 1	$T_1[1] \oplus T_2[1]$

Úloha 3 (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.

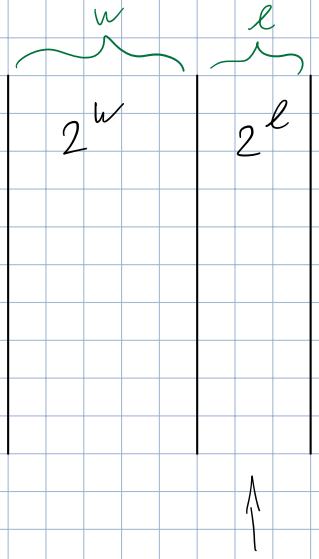
b) Začneme s případem, kdy existuje část i, že x^i, y^i, z^i jsou všechny různé. Mějme jakkoliv zvolené ostatní tabulky, kromě tabulky T_i. S jakou pravděpodobností můžeme zvolit funkci pro tabulku T_i tak, že $h(x) = a, h(y) = b, h(z) = c$?

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.



Celkem řádků: 2^w

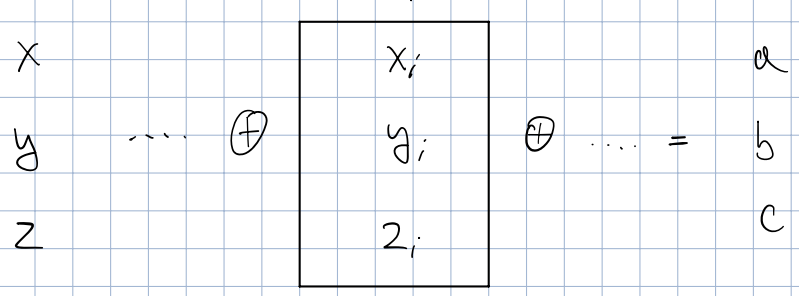
Celkem tabulek: $(2^\ell)^{2^w} = 2^{\ell 2^w}$

a) Jelikož mám jen jednu tabulku, tak rozdělení výstupu (do ℓ bitů) je uniformně náhodné.

Proto pokud $h(x) \neq h(y) \neq h(z) \neq h(x)$, tak $P(\text{kolize tří prvků}) \leq \frac{1}{m^3}$

$$= \frac{1}{2^\ell} \cdot \frac{1}{2^\ell} \cdot \frac{1}{2^\ell} = \frac{1}{2^{3\ell}} = \frac{1}{m^3}$$

b)



– můžeme XORy přesunout tak, že T_i bude poslední tabulka.

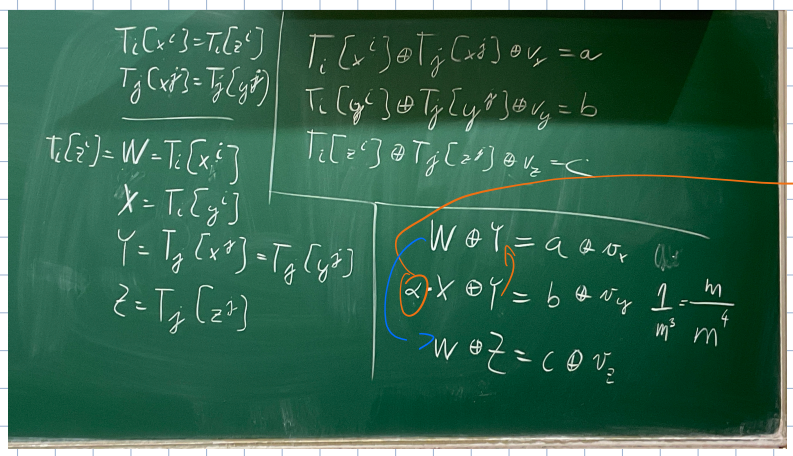
T_i hashuje do $2^{\frac{w}{d}}$ bitů. Může mít celkem $2^{\frac{w}{d}-3}$, které budou splňovat ten poslední XOR operaci.

$$P[T_i \text{ je správně}] = \frac{(2^\ell)^{2^{\frac{w}{d}-3}}}{(2^\ell)^{2^{\frac{w}{d}}}} = 2^{\ell-3} = \frac{1}{(2^\ell)^3} = \frac{1}{m^3}$$

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?



→ po dvou různé

→ Jakmile zafixuju jednu proměnnou, mám jednoznačně určené Y. Pak ale mám určené i W. Tudíž pak i Z.

Tudíž celkem mám $\frac{1}{m^4}$ ohodnocení proměnných X, Y, W, Z, ale pouze m z nich správných. Prst. je $\frac{m}{m^4} = \frac{1}{m^3}$

tedy 3-nezávislost