

Data structure 2 - LS 2015/2016 - Martin Mareš

- Poprvé podle nové akreditace - dost jiná přednáška než dříve (snad je to zúčtená k přednášce?)
- Kontakty: mj@ucw.cz, <https://mj.ucw.cz/vyuka/ds2/>
- Cvičení nejsou, ale můžete si domluvit konzultaci.
- Přibližný plán:
 - statické sloupky
 - celočíslné DS - haldy (bin., Fib)
 - ~~lokalní~~ ~~odhad~~ - více o Splay stromech
 - cache-oblivious DS
 - DS pro stromy a obecné grafy
 - ~~geometrické~~ DS - transformace
 - úsporné DS
 - streaming algoritmy

→ repríza 2017/2018
(jen mírné úpravy)
2019/2020
2021/2022
2023/2024

Literatura dosti kusá,
budeme přidávat odkazy
na web
+ scenované poznámky
+ videozáznamy z r. 2016
+ rozepsaná kniha a 2020

- Požadavky ke zkoušce: znát odpřednesené, umět to aplikovat a upravovat

VÝPOČETNÍ MODEL

- kdybychom studovali poly. vs. exp., na modelu nezáleží
- u DS ale potřebujeme rozlišovat $\log n / \log \log n / O(1) / \dots \Rightarrow$ model musíme specifikovat
- Budeme používat Word-RAM (neřekneme-li jinak)
 - w-bitová celá čísla - slova
 - na slovech umíme počítat v konstantním čase (jako v Cětku ...)
 - aritmetika: +, -, *, /, %
 - logické operace: &, |, ^, <<, >>, ~
 - porovnávání: =, <, >
 - paměť je pole slov indexované slovy $\rightarrow$ potřebujeme $w \geq \log_2 n$
 - vstup a výstup předáváme v paměti
 - čas = # provedených instrukcí
 - prostor = rozepí mezi min. a max. adresou použitých paměťových buněk

Bůho dokážeme
počítat
i s O(w)-bit. slovy

všechny logaritmy
budou nadále implicitně
dvojkové

STATICKE MAPOVANI

universum
(třeba slova RAMu)

- Chceme pro n-prvkovou $S \subseteq U$ vybudovat DS, která bude umět rychle odpovídat na dotazy "x ∈ S?"

	Build	Member	
• Co už umíme:	$O(n \log n)$	$O(\log n)$	vyhledávací strom [v porovnávacím modelu ještě lépe]
	$O(n)$ průměrně	$O(1)$ w.c.	kukací heslování (potřebuje $\log n$ -nezávislou radinu fci)
• Ukážeme:	$O(n)$ průměrně	$O(1)$ w.c.	perfektní heslování FKS (stačí 2-nezávislost)
	$O(n)$ průměrně	$O(1)$ w.c.	... jiný přístup...
	$O(n \log n)$ w.c.	$O(1)$ w.c.	demondomizace

Opakování: Def: Systém $\mathcal{H}$ hesovacích funkcí $U \rightarrow [m]$ je c-universální ($c > 0$)
 $\equiv \forall x, y \in U, x \neq y: \Pr_{h \in \mathcal{H}}[h(x) = h(y)] \leq c/m$.

Navíc obvykle chceme "hezkou parametrizaci" - tedy aby náhodný výběr $h \in \mathcal{H}$ šlo provést rovnoměrně náhodným výběrem $O(1)$ parametrů a pomocí nich pak $h(x)$ vyhodnotit v čase $O(1)$.

Pro $S \subseteq U$ a funkci $h \in \mathcal{H}$ počítáme kolize: $\{x, y\} \in \binom{S}{2}$ t.j. $h(x) = h(y)$.

nastane s jstí $\leq \frac{c}{m}$

$$\mathbb{E}[\# \text{kolizí}] = \sum_{\{x, y\}} \mathbb{E}[C_{xy}] \leq \binom{n}{2} \cdot \frac{c}{m} \leq \frac{n^2 \cdot c}{2m}$$

↑
indikátor kolize

Pro $m = n^2$ je $\mathbb{E}_{h \in \mathcal{H}}[\# \text{kolizí}] < \frac{1}{2}$, takže podle Markovovy nerovnosti je

$$\Pr_h[h \text{ koliduje na } S] = \Pr[\# \text{kolizí} \geq 2 \cdot \mathbb{E}[\# \text{kolizí}]] \leq \frac{1}{2}.$$

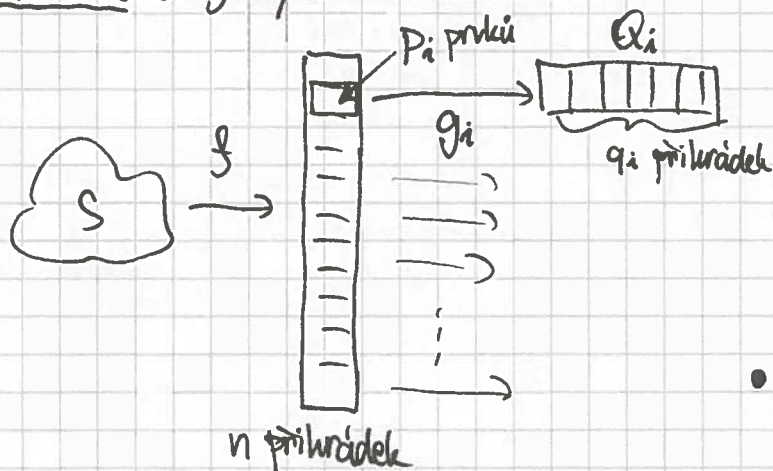
$\Rightarrow$ proto budeme-li volit h náhodně z $\mathcal{H}$, tak dlouho, než objevíme nějakou perfektní, bude to trvat průměrně ≤ 2 pokusy:

Lemma (O čekání): Čekáme-li na událost, která nastane s jstí p , pak $\mathbb{E}[\# \text{pokusy, než se dočkáme}] = 1/p$.

... jeden pokus trvá $O(n)$ [pokud kolize detekujeme přírůdkovým tříděním], takže v čase průměrně $O(n)$ perfektní fci najdeme.

... jenže potřebujeme kvadraticky velkou tabulku $\Rightarrow$ inicializace stojí $O(n^2)$.

Konstrukce dvojstupňového hešování:



$f \in \mathcal{H}$ hešuje do n přírůdek

$$\mathbb{E}[\# \text{kolizí}] \leq \frac{cn}{2}$$

... umím efektivně najít f , pro něj $\# \text{kolizí} < cn$.

g_i hešuje do $[q_i]$ pro $q_i = \lceil P_i^2 \cdot c \rceil$, takže umíme najít perfektní g_i

Dokážeme, že celková velikost všech Q_i je $O(n)$:

$$\sum_{i=1}^n q_i \leq n + c \cdot \sum_i P_i^2 \leq n + c \left(\sum_i P_i \right) + c \left(\sum_i \binom{P_i}{2} \right) \in O(n).$$

$\uparrow$
 $P_i + 2 \cdot \binom{P_i}{2}$
 $\# \text{kolizí v } i\text{-té přírůdce}$

$\# = \# \text{všech kolizí pro } f \leq \frac{cn}{2}$

Spotřeba paměti:

- parametry $f \dots O(1)$
- parametry $g \dots O(1)$
- tabulka indexovaná f (pointery na Q_i) $\dots O(n)$
- tabulky $Q_i \dots O(n)$

} celkem $O(n)$

Čas na konstrukci:

- průměrně $O(n)$ na volbu f
- průměrně $O(q_i)$ na volbu g_i

} celkem průměrně $O(n)$

Čas na dotaz:

- výpočet f
- nahlédnutí do hlavní tabulky pro pointer a param. g_i
- výpočet g_i
- nahlédnutí do Q_i

} $O(1)$ w.c.

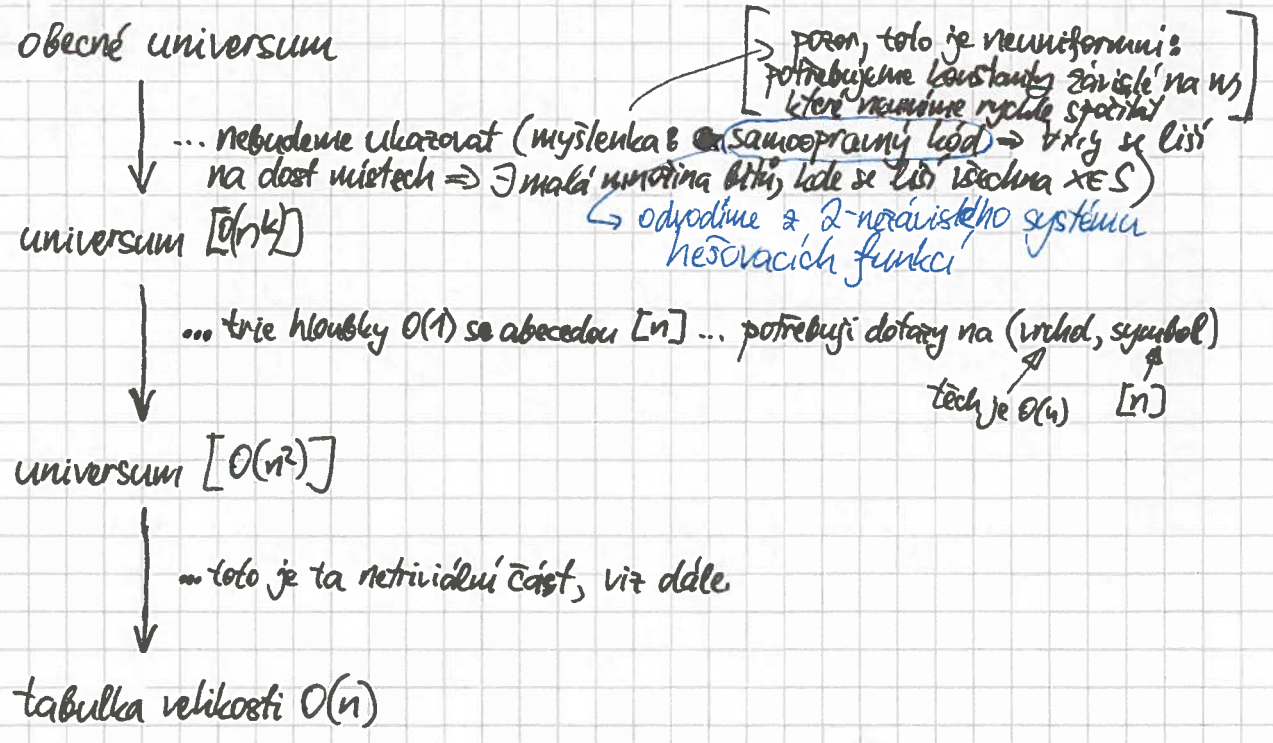
Poznámka: $\exists$ dynamizace s časem $O(1)$ průměrně amortizované na Ins/Del a $O(1)$ w.c. na dotaz.

Odhad: Třídění reálných čísel vybraných rovnoměrně náhodně z $[0,1]$.

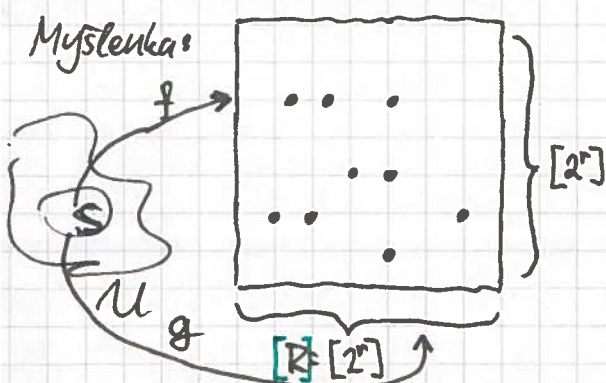
- Rozdělíme $[0,1]$ na n příhradek
 - v $O(n)$ rozmístíme čísla do příhradek $\dots \in [\# \text{kolizí}] < \frac{n^2}{2}$
 - v každé příhradce dotřídíme bublinkově
- $\dots$ trvá to $O(\text{velikost příhrádky}^2)$, což se seče na $O(n)$

DETERMINISTICKÉ SLOVNÍKY Hagerup, Miltersen, Pagh 2000

- Nejprve ukážeme ~~deterministickou~~ ^{randomizovanou} verzi, pak ji derandomizujeme.
- Skládáme několik transformací: (všechno to jsou prosté funkce)



• Notace: nerozlišujeme mezi čísly z $[O(n^2)]$ a bit. řetěci z $\{0,1\}^{2 \log n + O(1)}$



Prvky z S odpovídají bodům ~~na~~ v mřížce $[2^r] \times [2^r]$.
 Chceme je transformovat tak, aby v řádce byl max. 1 bod.

Krok 1 $(i, j) \rightarrow (i, j \oplus a_i)$
 ... a pak totéž znovu ve sloupcích

počet řádek n -krát, sníží #kolizi
 omezíme #kolizi ve sloupcích funkcí #kolizi v řádcích

Df: Dvojice funkcí (f, g) z $\mathcal{U}$ do $[R]$ je q -dobrá (pro $q \geq 0$), pokud f má na S nejvýše q kolizí a $x \mapsto (f(x), g(x))$ je na S prostá.

Lemma: Necht' (f, g) je q -dobrá a $r \geq \log n + 1$.

$q' = \min(n, \lfloor 2^{3-r} \cdot q \rfloor \cdot n)$

Pak $\exists a_0, \dots, a_{2^r-1} \in [R]$ t.z. $(x \mapsto g(x) \oplus a_{f(x)}, f(x))$ je q' -dobrá pro $q' = \lfloor 2^{3-r} \cdot q \rfloor \cdot n$
 posunu řádky a transponuji mřížku

Navíc všechna a_i lze pro dané S, f, g spočítat randomizovaně v očekávaném čase $O(n)$ a worst-case prostoru $O(n)$.

Využití: Mějme nějakou $S \subseteq \{0, 1\}^w$. Zvolíme $r > \max(w/2, \log n + 3)$.

je-li $\mathcal{U}$ je velké $O(n^2)$, je $r \leq \log n + O(1)$ a $R \in O(n)$

0. krok: (f, g) rozkládají $x \in S$ na horních a dolních r bítů (s překryvem, je-li třeba)

- (f, g) je jisté prostá na S
 - f má na S nejvýše $\binom{n}{2} < n^2$ kolizí.
- } pár (f, g) je n^2 -dobrý

Lemma (zde je potřeba omezení $q' \leq n$ z též lemma) $\Downarrow$

1. krok: (f', g') ... jelikož $2^{3-r} < \frac{1}{n}$, musí tento pár být $< n$ -dobrý

Lemma (... a zde naopak druhá část ...)

2. krok: (f'', g'') ... < 1 -dobrý, takže 0-dobrý $\Rightarrow f''$ je prostá na S .

Výpočet hes. funkce

1. Rozdělíme x na ~~p, q~~ r -bitové
2. ~~$q \leftarrow q \oplus a_p$~~
3. $p \leftarrow p \oplus b_q$
4. Vyjdeme výsledek p

} čas $O(1)$

Prostor: Tabulky pro a, b + finální hesovací tabulka } $O(n)$ slov

$R = 46n$
 $\rightarrow$ about 46n words of memory (can be improved)

Dle lemmatu: Nejprve očíslovíme řádky od nejpolevnějšího:

$S_i = \{x \in S \mid f(x) = v_i\}$,
 kde $v_1, \dots, v_{2^r}$ je permutace na $[R] = [2^r]$
 taková, že $|S_1| \geq |S_2| \geq \dots \geq |S_{2^r}|$

[Dietzfelbinger et al. 2009]

for $m = 1.023n$
 we obtain perfect hash fn with 1.4 bits per key + $o(n)$ aux tables

V tomto pořadí řádkům přidělujeme jejich ~~tabulky~~ a_{v_i} .
 Build $O(n)$ expected (no randomness) Query $O(1)$ w.c.

• Necht' jsme již zpracovali $S_{i-1} := S_1 \cup \dots \cup S_{i-1}$ a přidáváme S_i .

Vybereme $a_{v_i} \in [R]$ náhodně, počítáme, kolik vzniklo nových kolizí:

$$E[\#NK] = |S_{i-1}| \cdot |S_i| \cdot 2^{-r}$$

za $O(1)$ pokusů najdu a_{v_i} ,
pro které $\#NK \leq \lfloor 2 \cdot E[\#NK] \rfloor$ ← je to celé číslo
 $|S_{i-1}| \cdot |S_i| \cdot 2^{1-r}$

dvojice $x \in S_{i-1}, y \in S_i$
t.j. $g(x) \oplus a_{g(x)} = g(y) \oplus a_{g(y)}$
 $a_{g(y)}$ pro nějaké $j \in i$ $= a_{v_i}$
 $\Leftrightarrow$
 $a_{v_i} = g(x) \oplus g(y) \oplus a_{g(x)}$
... to nastane s pstí $\frac{1}{2^r}$

• Jak to udělat efektivně? • Udržujeme $m_t :=$ kolik bodů jsme už umístili do t -tého sloupce
... tedy $\{x \in S_{i-1} \mid g(x) \oplus a_{g(x)} = t\}$.

• Na počátku seřadíme S podle $(f(x), g(x))$ lexikograficky ... $O(n \log n + 2^r)$
↳ pořadí $S_1 \dots S_{2^r}$ dalším přírůdkovým tříděním

• Pro každou S_i zvolíme a_{v_i} , pak pro $x \in S_i$ spočítáme pomocí m_x kolize
... až najdeme správné a_{v_i} , přepočítáme m_x
 2^n -krát
 $\left. \begin{array}{l} O(|S_i|) \\ \times O(1) \text{ příměrná} \\ \text{kolize} \end{array} \right\} O(|S_i|)$
↳ v součtu přes všechna i $O(n) + 2^r$ průměrně

• Jak dobrý pár jsme vyrobili?

Pro původní pár (f, g) : $\# \text{kolizí fce } f = \sum_i \binom{|S_i|}{2} \leq q$

Pro nový pár počítáme kolize fce $x \mapsto g(x) \oplus a_{g(x)}$:
 $\# \text{kolizí} \leq \sum_{i=1}^{2^r} \underbrace{[2^{1-r} \cdot |S_i| \cdot |S_{i-1}|]}_{\text{už víme}} \leq \sum_{i=1}^t [2^{1-r} \cdot |S_i| \cdot |S_{i-1}|] \leq \sum_{i=1}^t [2^{3-r} \cdot \sum_{j < i} \binom{|S_j|}{2}] \leq n \cdot [2^{3-r} \cdot q]$
shora omezeno touto
vynecháme členy, pro které $|S_i| \leq 1$
... tedy $|S_i| \leq 2^{n-1}$
také díky volbě r ,
takže $\lfloor \dots \rfloor = 0$
spolu $|S_i| = 0$, je celý součin triviálně 0,
pro $|S_i| = 1$ máme $|S_{i-1}| < 2^{n-1} \Rightarrow \lfloor \dots \rfloor = 0$.
 $2^{1-r} \leq \frac{1}{n}$, takže $\dots \leq |S_i|$
 $\rightarrow \sum_i \dots \leq n$
 $t_i = \max \{i \mid |S_i| > 1\}$ (to je druhá část užití (...) z první lemmatu)
 $|S_i| \cdot \sum_{j < i} |S_j| = \sum_{j < i} |S_i| \cdot |S_j| \leq \sum_{j < i} |S_j|^2$
 $\leq 4 \cdot \binom{|S_i|}{2}$
jelikož $|S_j| \geq 2$

Derandomizace

6

- Obecný trik: Hledáme z : $T(z) \leq \mathbb{E}[T]$

T je obecně nějaká náhodná veličina, tedy funkce, jejíž výsledek je náhodný.

Postupně fixujeme části z tak, aby $\mathbb{E}[T | \text{fixovaná část}]$ ^{nerostla} ~~nehlavově~~.

Využíváme toho, že $\mathbb{E}[T] = P(\alpha) \cdot \mathbb{E}[T | \alpha] + (1 - P(\alpha)) \cdot \mathbb{E}[T | \neg \alpha]$.

V našem případě bude $P(\alpha) = \frac{1}{2}$, takže stačí vybrat $z \in \{T | \alpha\}$ a $\mathbb{E}[T | \neg \alpha]$.

- Původní randomizovaný krok vypadal takto:

- máme tabulku všech m_t a množinu $X \subseteq [R]$ ^{ta hraje roli} $\{g(x) | x \in S_i\}$

- hledáme $a \in [R]$ t.j. $\sum_{x \in X} m_{x \oplus a} \leq \lfloor 2^{1-r} \cdot |X| \cdot \sum_{t \in T} m_t \rfloor$ ^{to je $|S_i|$}

randomizovaně jsme to uměli v $O(|X|)$ průměrně, ukážeme, jak deterministicky v $O(|X| \cdot r) = O(|X| \cdot \log n) \Rightarrow$ sečte se na $O(n \log n)$

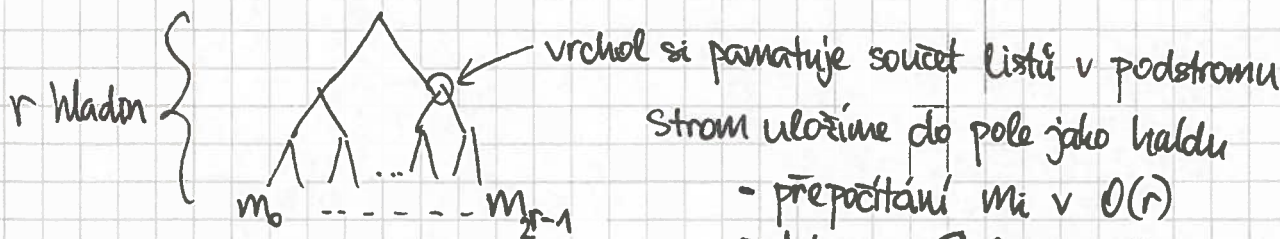
- Postupně fixujeme bity čísla a od nejvyššího a přepočítáváme $\mathbb{E}[\# \text{listů} | \pi_k(a) = A]$

... stále tuto $\mathbb{E}[\dots]$ udržujeme pod původní $\mathbb{E}[\dots]$, stačí ji umět rychle spočítat. ^{prefix délky k}

$$\mathbb{E}\left[\sum_{x \in X} m_{x \oplus a} \mid \pi_k(a) = A\right] = \sum_{x \in X} \mathbb{E}[m_{x \oplus a} \mid \pi_k(a) = A] = \sum_{x \in X} \mathbb{E}[m_t \mid \pi_k(t) = \pi_k(x) \oplus A]$$

průměr všech m_t pro daný prefix $\pi_k(t)$

Budeme udržovat intervalový strom nad všemi m_t :



Strom uložíme do pole jako haldy

- přepočítání m_i v $O(r)$
- dotaz na $\sum$ listů pro daný prefix v $O(1)$

$\Rightarrow$ 1 krok derandomizace zvládneme v $O(|X|)$... prefixy počítáme v $O(1)$ ^{bitovými operacemi}

- celou volbu a zvládneme v $O(|X| \cdot r)$
- pak v $O(|X| \cdot r)$ aktualizujeme strom

celý algoritmus
bude v $O(nr) = O(n \log n)$.