Datové struktury 2 – LS 2015/2016 – Martin Mareš

- Poprvé podle nové akreditace – dost jiná přednáška než dříve (snad je to změna k lepšímu)
- Kontakty: mj@ucw.cz, http://mj.ucw.cz/vyuka/ds2/ ↳ repríza 2017/2018 (jen mírné úpravy)
- Cvičení nejsou, ale můžete si domluvit konsultaci.
- Přibližný plán:
  - statické slovníky
  - celočíselné DS
  - dolní odhady
  - cache-oblivious DS
  - DS pro stromy a obecné grafy
  - geometrické DS
  - úsporné DS
  - streaming algoritmy

  literatura dosti kusá; budeme přidávat odkazy na web + scanované poznámky + videozáznamy z r. 2016
- 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 $\log n / \log\log n / O(1) / \ldots \Rightarrow$ model musíme specifikovat

- Budeme používat <u>Word-RAM</u> (neřekneme-li jinak)
  - w-bitová celá čísla – <u>slova</u>
  - na slovech umíme počítat v konstantním čase (jako v Céčku …)
    - aritmetika: $+, -, *, /, \%$
    - logické operace: $\&, |, \wedge, \ll, \gg, \sim$
    - porovnávání: $=, <, >$

    BÚNO dokážeme počítat i s $O(w)$-bit. slovy
  - paměť je pole slov indexované slovy $\rightarrow$ potřebujeme $w \geq \log_2 n$

    všechny logaritmy budou nadále implicitně dvojkové
  - vstup a výstup předáváme v paměti
  - čas = # provedených instrukcí
  - prostor = rozpětí mezi min. a max. adresou použité paměťové buňky

STATICKÉ MNOŽINY

- Chceme pro n-prvkovou $S \subseteq \mathcal{U}$ (universum, třeba slova RAMu) vybudovat DS, který bude umět rychle odpovídat na dotazy "$x \in S$?"

	Build	Member	
• Co už umíme:	$O(n \log n)$	$O(\log n)$	vyhledávací strom [v porovnávacím modelu nelze lépe]
	$O(n)$ průměrně	$O(1)$ w.c.	kukaččí hešování (potřebuje $\log n$-nezávislou rodinu fcí)
• Ukážeme:	$O(n)$ průměrně	$O(1)$ w.c.	perfektní hešování FKS (stačí 2-nezávislost)
	$O(n)$ průměrně	$O(1)$ w.c.	… jiný přístup …
	$O(n \log n)$ w.c.	$O(1)$ w.c.	derandomizace

PERFEKTNÍ HEŠOVÁNÍ

FKS = Fredman, Komlós, Szemerédi 1984

Opakování: **Df:** Systém $\mathcal{H}$ hešovacích funkcí $\mathcal{U} \to [m]$ je <u>c-universální</u> ($c>0$)
$\equiv \forall x,y \in \mathcal{U},\ x \neq y:\ \Pr_{h\in\mathcal{H}}[h(x)=h(y)] \le c/m$.

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

Pro $S \in \binom{\mathcal{U}}{n}$ a funkci $h: \mathcal{U} \to [m]$ počítáme <u>kolize</u>: $\{x,y\} \in \binom{S}{2}$ t.ž. $\underbrace{h(x)=h(y)}_{\text{nastane s pstí } \le \frac{c}{m}}$.

<u>Lemma:</u> $$\mathbb{E}_{h\in\mathcal{H}}[\#\text{kolizí}] = \sum_{\{x,y\}} \mathbb{E}[\underset{\text{indikátor kolize}}{C_{xy}}] \le \binom{n}{2}\cdot\frac{c}{m} \le \frac{n^2 \cdot c}{2m}$$

Pro $m = \lceil n^2 c \rceil$ je $\mathbb{E}_{h\in\mathcal{H}}[\#\text{kolizí}] < \frac{1}{2}$, takže podle Markovovy nerovnosti je
$\Pr_h[h \text{ koliduje na } S] = \Pr[\#\text{kolizí} > 2\cdot\mathbb{E}[\#\text{kolizí}]] < \frac{1}{2}$.

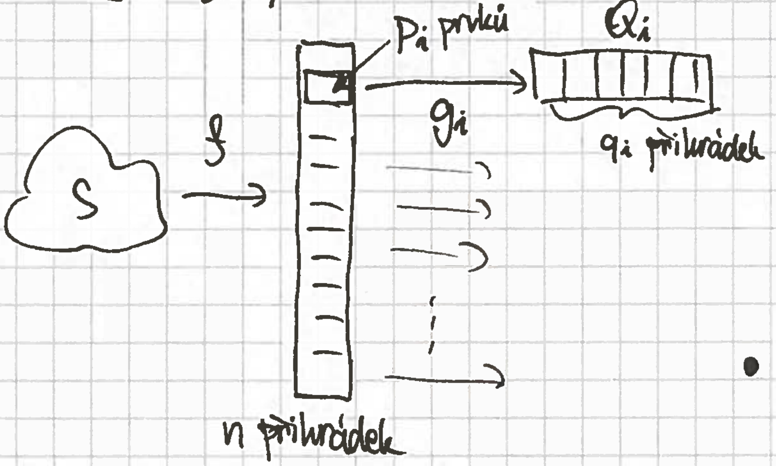
$\Rightarrow$ proto budeme-li volit h náhodně z $\mathcal{H}$, tak dlouho, než objevíme nějakou perfektní, bude to trvat průměrně ≤ 2 pokusy:

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

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

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

<u>Konstrukce</u> dvojstupňového hešování:



- $f \in \mathcal{H}$ hešuje do n přihrádek $\Rightarrow \mathbb{E}[\#\text{kolizí}] \le \frac{cn}{2}$
  ... umím efektivně najít f, pro níž $\#\text{kolizí} \le cn$.
- g_i hešuje do $[q_i]$ pro $q_i = \lceil P_i^2 \cdot c \rceil$, takže umíme najít perfektní fci

Dokážeme, že celková velikost všech Q_i je $\mathcal{O}(n)$:

$$\sum_{i=1}^{n} q_i \le n + c\cdot\sum_i \underset{P_i \cdot 2\cdot\binom{P_i}{2}}{P_i^2} \le n + c\Big(\underbrace{\sum_i P_i}_{n}\Big) + c\Big(\underbrace{\sum_i \binom{P_i}{2}}_{=\,\#\text{všech kolizí fce } f \,\le\, \frac{cn}{2}}\Big) \in \mathcal{O}(n).$$

$\binom{P_i}{2}$ = # kolizí v i-té přihrádce fce f

Spotřeba paměti:
- parametry f … $O(1)$
- parametry g … $O(n)$
- tabulka indexovaná f (pointery na Q_i) … $O(n)$
- tabulky Q_i … $O(n)$

} 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 amortizovaně na Ins/Del a $O(1)$ w.c. na dotaz.

Odbočka: Třídění reálných čísel vybíraných rovnoměrně náhodně z $[0,1]$.

- Rozdělíme $[0,1]$ na n přihrádek
- v $O(n)$ rozmístíme čísla do přihrádek … $E[\#\text{kolizí}] < \frac{n^2}{2}$
- v každé přihrádce dotřídíme bublinkově
  … trvá to $O((\text{velikost přihrádky})^2)$, což se sečte na

} průměrně $O(n)$

DETERMINISTICKÉ SLOVNÍKY

Hagerup, Miltersen, Pagh 2000

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

obecné universum

↓ … nebudeme ukazovat (myšlenka: samoopravný kód $\Rightarrow$ $x \neq y$ se liší na dost místech $\Rightarrow$ $\exists$ malá množina bitů, kde se liší všechna $x \in S$)

[pozor, toto je neuniformní: potřebujeme konstanty závislé na n, které neumíme rychle spočítat]

↳ odvodíme z 2-nezávislého systému hešovacích funkcí

universum $[O(n^k)]$

↓ … trie hloubky $O(1)$ se abecedou $[n]$ … potřebuji dotazy na (vrchol, symbol)

těch je $O(n)$ … $[n]$

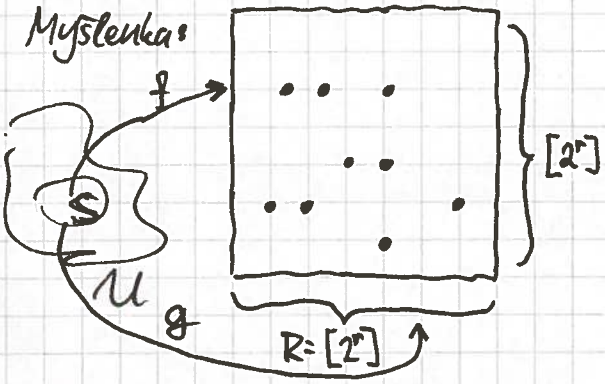
universum $[O(n^2)]$

↓ … toto je ta netriviální část, viz dále.

tabulka velikosti $O(n)$

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

Myšlenka:



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

Krok: $(i,j) \to (i, j \oplus a_i)$ } po každé řádce n-krát, sníží #kolizí

... a pak totéž ~~znovu~~ ve sloupcích

[omezíme #kolizí ve sloupcích funkcí #kolizí 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: Nechť (f,g) je q-dobrá a $r \geq \log n + 1$.
Pak $\exists a_0 \ldots a_{2^r-1} \in R$ t.ž. $\underbrace{(x \mapsto g(x) \oplus a_{f(x)},\ f(x))}_{\text{posunu řádky a transponuji mřížku}}$ je q'-dobrá pro $q' = \cancel{\lfloor 2^{3-r} q \cdot n \rfloor} \cdot n$

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

Navíc všechna a_i lze pro dané S, f, g spočítat randomizovaně v očekávaném čase $O(n + 2^r)$ a worst-case prostoru $O(n + 2^r)$. } za předpokladu, že f,g umíme vyhodnocovat v konst. čase

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

(jelikož $\mathcal{U}$ je velké $O(n^2)$, je $r \leq \log n + O(1)$)

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

- (f,g) je jistě 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 tvrzení lemmatu)

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

} čas $O(1)$

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

Dk lemmatu: Nejprve očíslujeme řádky od nejplnějšího:

$$S_i := \{ x \in S \mid f(x) = v_i \},$$

kde $v_1 \ldots v_{2^r}$ je permutace na $R = [2^r]$
taková, že $|S_1| \geq |S_2| \geq \ldots \geq S_{2^r}$
V tomto pořadí řádkům přidělujeme jejich ~~...~~ a_{v_i}.

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

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

dvojic $x \in S_{<i}$, $y \in S_i$ t.ž. $g(x) \oplus \underbrace{a_{g(x)}}_{a_{v_j} \text{ pro nějaké } j<i} = g(y) \oplus \underbrace{a_{g(y)}}_{= a_{v_i}}$

$\Updownarrow$

$a_{v_i} = g(x) \oplus g(y) \oplus a_{g(x)}$ ... to nastane s pstí $\frac{1}{2^r}$

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

$\downarrow$

za $O(1)$ pokusů najdu a_{v_i}, pro které $\#NK \le \lfloor 2 \cdot \mathbb{E}[\#NK] \rfloor$ ← je to celé číslo

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

- Jak to udělat efektivně? • Udržujeme m_t := kolik bodů jsme už umístili do t-tého sloupce

... tedy $\left|\{x \in S_{<i} \mid g(x) \oplus a_{g(x)} = t\}\right|$.

  - Na počátku setřídíme S podle $(f(x), g(x))$ lexikograficky ... $O(n + 2^r)$

    ↳ pořadí $S_1 \dots S_{2^r}$ dalším přihrádkovým tříděním

  - Pro každou S_i zvolíme a_{v_i}, pak pro $\forall x \in S_i$ spočítáme pomocí m_* kolize $\}\ O(|S_i|)$ $\times\ O(1)$ průměrně pokusů

    ... až najdeme správné a_{v_i}, přepočítáme m_* $\}\ O(|S_i|)$

    2^r-krát

    ↳ v součtu přes všechna i: $O(n + 2^r)$ průměrně

- Jak dobrý pár jsme vyrobili?

Pro původní pár (f,g): # kolizí fce $f = \sum_i \binom{|S_i|}{2} \le q$ ← shora omezeno tímto

Pro nový pár počítáme kolize fce $x \mapsto g(x) \oplus a_{f(x)}$:

$$\#\text{kolizí} \le \sum_{i=1}^{2^r} \lfloor \underbrace{2^{1-r} \cdot |S_i| \cdot |S_{<i}|}_{\text{už víme}} \rfloor \le \sum_{i=1}^{t} \lfloor 2^{1-r} \cdot \underbrace{|S_i| \cdot |S_{<i}|} \rfloor \le \sum_{i=1}^{t} \left\lfloor 2^{3-r} \sum_{j<i} \binom{|S_j|}{2} \right\rfloor \le n \cdot \lfloor 2^{3-r} \cdot q \rfloor.$$

$|S_i| \cdot \sum_{j<i} |S_j| = \sum_{j<i} |S_i| \cdot |S_j| \le \sum_{j<i} |S_j|^2$

$|S_j|^2 \le 4 \cdot \binom{|S_j|}{2}$ jelikož $|S_j| \ge 2$

vynecháme členy, pro které $|S_i| \le 1$

... $|S_{<i}| < 2^{r-1}$ platí díky volbě r,

takže $\lfloor \dots \rfloor = 0$

pokud $|S_i| = 0$, je celý součin triviálně 0, pro $|S_i| = 1$ víme $|S_{<i}| < 2^{r-1} \Rightarrow \lfloor \dots \rfloor = 0$.

$t := \max\{i \mid |S_i| > 1\}$

$2^{1-r} \le \frac{1}{n}$, takže $(\dots) \le |S_i|$

$\rightarrow \sum_i (\dots) \le n$

(to je druhá část $\min(\dots)$ z tvrzení lemmatu)

Derandomizace

T je obecně nějaká náhodná veličina, tedy funkce jen z členu.

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

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

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

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

- Původní randomizovaný krok vypadal takto:
  - máme tabulku všech m_t a (multi)množinu $X \subseteq R$ ta hraje roli $\{g(x) \mid x \in S_i\}$
  - hledáme $a \in R$ t.ž. $\sum_{x\in X} m_{x\oplus a} \le \left\lfloor 2^{1-r}\cdot|X|\cdot \underbrace{\sum_t m_t}_{\text{to je } |S_{<i}|} \right\rfloor$

  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 $\mathbb{E}[\#\text{kolizí} \mid \pi_k(a) = A]$ prefix délky k

  ... stále tuto $\mathbb{E}[\ldots]$ udržujeme pod původní $\mathbb{E}[\ldots]$, stačí ji umět rychle spočítat.

$$\mathbb{E}\left[\sum_{x\in X} m_{x\oplus a} \,\middle|\, \pi_k(a)=A\right] = \sum_{x\in X}\mathbb{E}\left[m_{x\oplus a} \,\middle|\, \pi_k(a)=A\right] = \sum_{x\in X} \underbrace{\mathbb{E}\left[m_t \,\middle|\, \pi_k(t)=\pi_k(x)\oplus A\right]}_{\text{průměr všech } m_t \text{ pro daný prefix } \pi_k(t)}$$

Budeme udržovat intervalový strom nad všemi m_t:

vrchol si pamatuje součet listů v podstromu

Strom uložíme do pole jako haldu
- přepočítání m_i v $O(r)$
- dotaz na Σ 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 běží v $O(nr) = O(n\log n)$.