Hešování

11 November, 2021 13:18

Slovníkový problém

univerzum $\mathcal{U}$, $S \subseteq \mathcal{U}$, $|S| = n$

→ chceme reprezentovat S

operace:
- Find(x) / Member(x)
- Insert(x)
- Delete(x)

Naivní řešení: pole velikosti $\mathcal{U}$, informace o $x \in \mathcal{U}$ je uložena na pozici x.

lépe: hašování do pole velikosti $m \geq n$.

$h : \mathcal{U} \to \{1, \dots, m\}$ hašovací fce.

→ prvek x se uloží na pozici $h(x)$.

- typicky h může být vybráno náhodně z nějaké množiny vhodných fcí:

např.: 1) $h(x) = ((ax + b) \bmod |\mathcal{U}|) \bmod m$ pro náhodně zvolená $a, b \in \mathcal{U}$.

2) h je zcela náhodná fce $\mathcal{U} \to \{1, \dots, m\}$

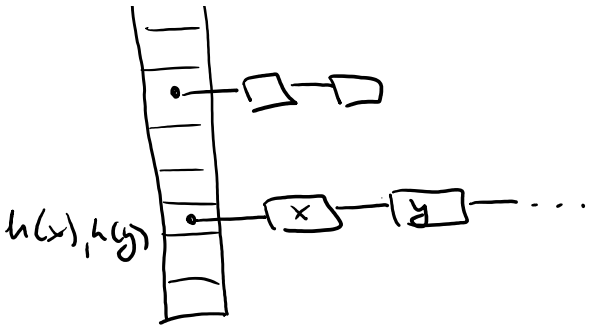
a čas od času se změní. (viz dále)

problém: kolize $x, y \in S$ $x \neq y$ $h(x) = h(y)$

způsoby řešení kolizí:

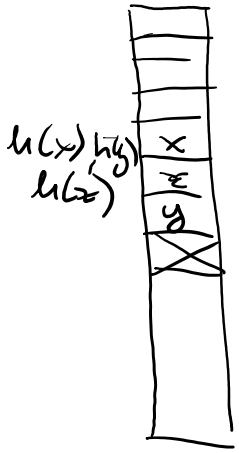
separované řetězce

řetězení



Seznam / vyhledávací strom

lineární přidávání



Insert(x) - najdu nejbližší volnou pozici za $h(x)$ a tam uložím x.

Find(x) - hledám od $h(x)$ do nejbližší volné pozice

dvojité hašování - podobně jako lineární přidávání, ale prvek x ukládám na pozici

$$h_1(x) + i\, h_2(x) \qquad i = 0, 1, 2, \ldots$$

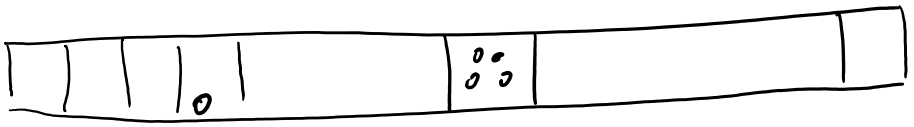
kde $h_1()$ a $h_2()$ jsou dvě různé nezávislé hašovací fce.

(je potřeba, aby $h_2(x)$ bylo nesoudělné s m, což je pravda např. pokud m je prvočíslo a $h_2(x) \in \{1, \ldots, m-1\}$.)

- řada dalších způsobů
- DELETE: vesměs problematický
  → označování smazaných prvků pro Find()

~ jejich znovupoužití při Insert().
→ pokud je smazaných prvků příliš mnoho, přehashuj vše.

• Balls & Bins: n míčků, n košů, každý míček hodím do náhodného z nich.



$$\Pr[\text{daný koš je prázdný}] = \left(1-\frac{1}{n}\right)^n \underset{n\to\infty}{\approx} \frac{1}{e}$$

$$\Pr[\text{daný koš obsahuje } k \text{ míčků}] = \binom{n}{k}\frac{1}{n^k}\left(1-\frac{1}{n}\right)^{n-k}$$

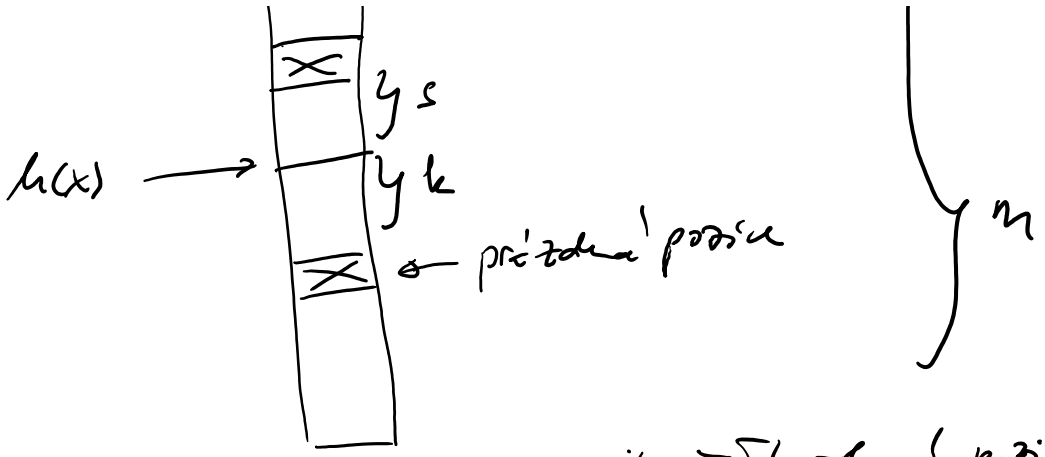
$$\underset{k \ll n}{\approx} \frac{n^k}{k!}\cdot\frac{1}{n^k}\cdot\frac{1}{e} = \frac{1}{e\,k!}$$

pro $k \approx \Theta\left(\frac{\lg n}{\lg\lg n}\right)$ $\Pr[\text{existuje koš s } k \text{ míčky}] \le \frac{1}{n^2}$

→ s velkou pstí, maximum v libovolném koši $\Theta\left(\frac{\lg n}{\lg\lg n}\right)$.

lineární přidávání – analýza

Uvažujme prvek x a předpokládejme, že h distribuuje prvky zcela náhodně.



$P_{k,s}$ ... pravděpodobnost, že nejbližší volná pozice je po $\leq$ krocích po $h(x)$ a před $h(x)$ je dalších s prvků

$$P_{k,s} = \binom{n}{k+s} \left(\frac{k+s}{n}\right)^{k+s} = (*)$$

$k+s$ prvků z n se muselo mapovat do kbelíků obsahujících $k+s$ z celkového počtu n pozic.

$$(*) \leq \frac{n^{k+s}}{(k+s)!} \cdot \frac{(k+s)^{k+s}}{n^{k+s}} = \left(\frac{n}{n}\right)^{k+s} \cdot \frac{(k+s)^{k+s}}{(k+s)!} =$$

$$\approx \left(\frac{n}{n}\right)^{k+s} \cdot \frac{1}{\sqrt{k+s}} \cdot e^{(k+s)} \cdot \frac{1}{\sqrt{2\pi}} = (**)$$

Stirlingova aprox: $a! \approx \sqrt{2\pi a}\left(\frac{a}{e}\right)^a$

Nechť $n \geq 3n$

$$(**) \leq \left(\frac{e}{3}\right)^{k+s} \frac{1}{\sqrt{(k+s)\,2\pi}}$$

střední doba vyhledávání $\leq \sum_{k \geq 1} k \cdot \Pr[\text{vyhledávání trvá čas } k]$

$$\leq \sum_{k \geq 1} k \cdot \sum_{s \geq 0} \left(\frac{e}{3}\right)^{k+s} \frac{1}{\sqrt{(k+s)2\pi}} = O(1)$$

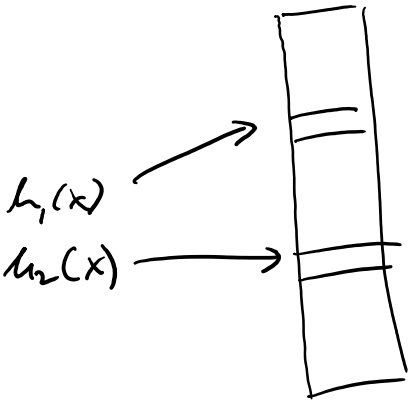
$$\leq \left(\frac{e}{3}\right)^k \cdot \sum_{s \geq 0} \left(\frac{e}{3}\right)^s \leq O\left(\left(\frac{e}{3}\right)^k\right)$$

$O(1)$ ∎

• Balls & Bins s volbou: n míčků, n košů, pro každý míček zvolím náhodně dva koše a míček hodím do prázdnějšího.

→ očekávaná maximální zaplněnost koše $O(\lg \lg n)$.

→ Kukačí hašování [Pagh - Rodler 2004]



$h_1, h_2 : U \to \{1, \ldots, m\}$
dvě nezávislé vybrané hašovací fce.

x je buď na pozici $h_1(x)$ nebo na pozici $h_2(x)$

Insert (x):

if $T[h_1(x)] = x$ or $T[h_2(x)] = x$ then return

```
pos = h1(x)
loop n times {
   if T[pos] is empty then { T[pos] = x; return; }
   swap(x, T[pos]);
   if pos = h1(x) then pos = h2(x) else pos = h1(x).
}
rehash(); insert(x);
```

end

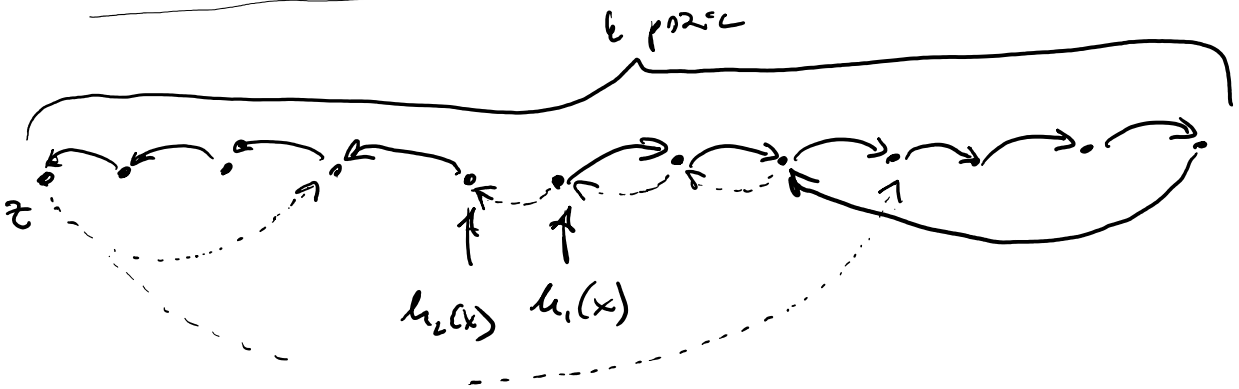
- Find O(1)
- Delete O(1)

} v nejhorším případě

- Insert O(1) v průměrném případě (přes volbu hašovacích fcí.)

→ pouze dvě možné pozice pro x.

- Kukačí hašování funguje obvykle pro $m \geq 2n$.
- Pohled na Insert(x):

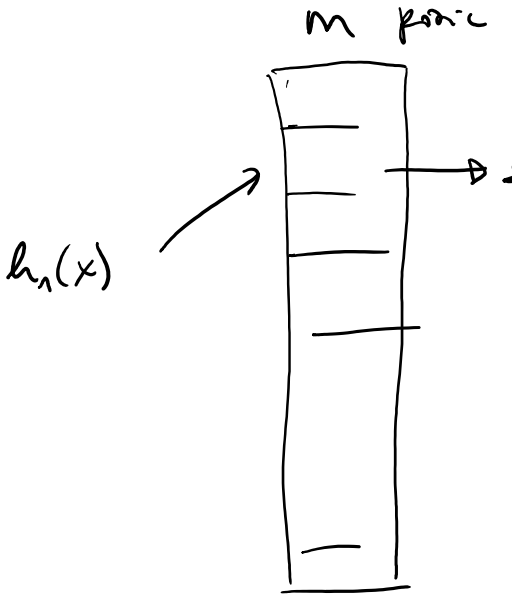


při neúspěšném insertu je pozice z obsazená a její prvek se mapuje na již prozkou

pozici → nekonečný cyklus - prvky se
posunují opozici dopředu a zpět
k pozic pro k+1 míček

- cyklus skončí zaručeně po $\approx 6 \lg n$ opakováních

Iceberg Hashing [Bender - Conway - Farach-Colton - Kuszmaul - Tagliavini '21]



$h_1 : U \to [m]$

$$m = \frac{n}{\lg^3 n}$$

C vhodná konst.

- B-strom pro až $\lg^3 n + C \lg^2 n$ prvků
  pokud B-strom přeteče, prvky se odložívají do "sklepa"

- $Pr[\text{daná přihrádka přeteče}] \leq \frac{1}{n^{C'}}$ (černorova nerovnost)
  C' libovolná konstanta, závisí na volbě C

- B-strom organizovaný podle klíče $h_2(x)$

• B-strom organizovaný podle klíče $h_2(x)$

$$h_2 : U \to [\ lg^{10} n\ ] \qquad B = \sqrt{lg\, n}$$

klíč $h_2(x)$ má $10 \lg\lg n$ - bitů

celý obsah B-stromu se vejde do jedné pomístné buňky velikosti $O(lg\, n)$ - bitová lze v níž vyhledat za použití běžných CPU instrukcí v čase $O(1)$. (viz níže)

hloubka stromu ≤ 7.

$\to$ počet instrukcí $O(1)$ na vyhledání.

- pokud dojde ke kolizi h_2 v rámci přihrádky, kolidující prvky se odloží do skladu (případně celá přihrádka)

$$Pr[\text{v dané přihrádce nastane kolize}] \leq \binom{lg^3 n + c\, lg^2 n}{2} \cdot \frac{1}{lg^{10} n} \leq O\left(\frac{1}{lg^4 n}\right)$$

↑ viz "perfektní hešování" dále

s velkou pravděpodobností, počet přihrádek s kolizí $O\left(\frac{n}{lg^3 n} \cdot \frac{1}{lg^4 n}\right)$ $\to$ počet prvků

v těchto přihrádkách je $O\left(\frac{n}{\lg^4 n}\right)$.

→ stačí sklep pro $O\left(\frac{n}{\lg^4 n}\right)$ prvků a pro něj lze použít např. kukačkové hašování.

paměť: přihrádka $\lg^3 n + c\lg^2 n$ prvků po $\lg|U|$ bitech.
+ B strom pro $\lg^3 n + c\lg^2 n$ prvků.
B strom obsahuje klíče velikosti $O(\lg\lg n)$ - bitů a ukazatele velikosti $O(\lg\lg n)$ - bitů
[B-strom bydlí v rámci přihrádky]

celkový počet bitů: $\frac{n}{\lg^3 n}\left[\left(\lg^3 n + c\lg^2 n\right)\cdot\left(\lg|U| + O(\lg\lg n)\right)\right]$

$= \left(1 + \frac{c}{\lg n}\right)\lg|U| + O(n\lg\lg n)$.

+ sklep $O\left(\frac{n}{\lg^4 n}\cdot\lg|U|\right)$ bitů.

→ d.s. velikosti $(1+o(1))\, n\lg|U|$
pro uložení n prvků
v. $\geq 2n\lg|U|$ pro kukačkové hašování

→ v'ne efektivní

- Find $O(1)$ operací
  Insert $O(1)$ s velkou pravděpodobností
  (kdykoliv nejdeme do sklepa)
  $O(1)$ v průměru
  Delete $O(1)$

Operace na B-stromu:

- hledám klíč r, r má $k = O(\lg \lg n)$ bitů
  v uzlu s k-bitovými klíči: $k_1 < k_2 < k_3 \dots < k_B$
  uložíme jako $|0|k_1|0|k_2|0|k_3|..|0|k_B|$
  v jednom slově $O(\lg n)$ bitů

$B = \sqrt{\lg n}$

1) $|0|1|0|1|..|1|0|1| \cdot |1|r| = |1|r|1|r|..|1|r|$

2) $|1|r|1|r|\quad|1|r| - |0|k_1|0|k_2|..|0|k_B| =$

$|0/1|*|0/1|*|\quad|0/1|*|$

↳ "$r > k_1$" ↳ "$r > k_B$"

3) $|0/1|*|0/1|*|\quad|0/1|*|$ & $|1|0|1|0|..|1|0|$

$= |0/1|0|0/1|0|\quad|0/1|0|$

↳ "$r > k_1$" ↳ "$r > k_B$"

4) COUNT( ) = #i, r > ki