Reprezentace množiny řetězců nad abecedou Σ : např.: $\{0\_9\}, \{a\_z\}$

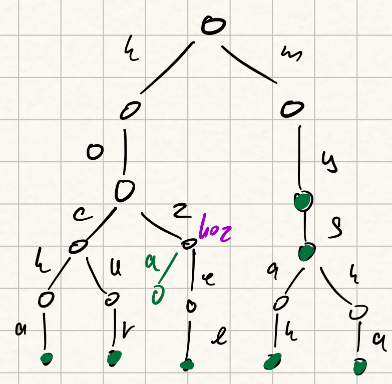
Pokud použijeme BST: $\Theta(\log n \cdot \underbrace{\text{délka řetězce}}_{\leq y})$

Find(y)

↓

Písmenkový strom (trie)

$\{$kocka, kocour, kozel, mys, mysak, myska$\}$



- vrcholy odpovídají prefixu
- strom + značky ve vrcholech

Operace

Member (y): $\Theta(|y|)$

Insert (y): $\Theta(|y|)$

- zakládá hrany, co neexistují
- indexuje značkou tam, kde dané slovo skončilo

Delete (y): $\Theta(|y|)$

- smažu značku a smažu mrtvé větve (po písmenkách)

Paměť: $\Theta(\sum_i |x_i|)$

- Maximální celková délka všech řetězců, pokud se písmena nepřekrývají.
- tedy lineární vzhledem k velikosti slovníku

Co když je ale abeceda velká?

- lineárně se prodlužuje Insert, delete a paměť, protože v každém vrcholu je pole indexů, které se musí projít
- Proto ve vrcholu nebude pole, ale BST/Set

Pak: Paměť: $\Theta(\sum_i |x_i|)$ --> * velikost abecedy

Čas operací: $\Theta(|y| \cdot \log|\Sigma|)$

Číslicový strom / radix trie

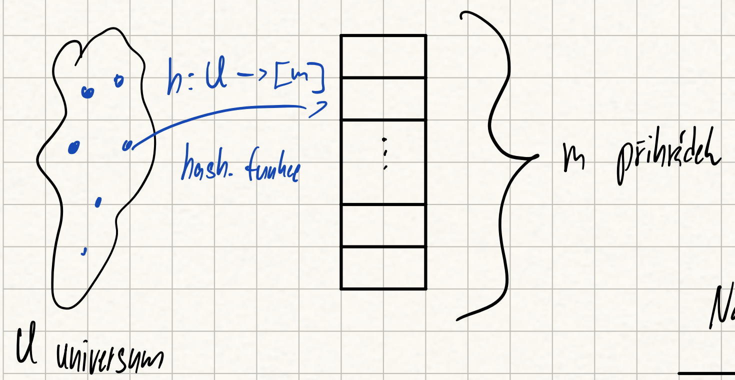
- zápis čísla v soustavě o základu Z uložíme jako řetězec

Složitosti

základ 2, n čísel z rozsahu $\{0 - L-1\}$ --> #číslic $\sim \log_2 L$

Čas: $\Theta(\log_2 L)$ Pokud jsou čísla různá, pak $L \geq n \Rightarrow \log L \geq \log n$

HASHOVÁNÍ



Kolize ≡ více prvků v přihrádce

↳ Triviální řešení: každá přihrádka obsahuje seznam [tabulka = pole ukazatelů]

Na základě nějakého konkrétního předpisu získám index, pomocí kterého přistupuji k datům

Snažíme se docílit rovnoměrného rozložení v přihrádkách

↳ To proto, aby měly všechny operace konstantní složitost

=> Find, Insert, Delete v čase $O(1)$

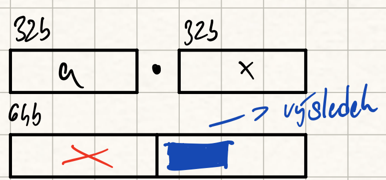
↳ ve skutečnosti je avg: $\Theta(1)$, worst $\Theta(n)$

Resp. složitost struktury, kterou se ukládají prvky do přihrádky

Jak zvolit hashovací f?

- $x \mapsto (ax) \bmod m$
  ↳ nesoudělné s m
  typicky $a \doteq 0{,}618m$
- Multiply-shift

32-bit PC:



pro $U = [2^w]$, $m = 2^h$

$x \mapsto (ax \bmod 2^w) >> w-h$ --> Celkem rychlé procesorové instrukce

- $x_0 \_\_ x_{a-1} \mapsto \left(\sum_i a_i x_i\right) \bmod m$ (skalární součin)

  parametry $a_0 - a_{a-1}$

- $x_0 \_\_ x_{a-1} \mapsto \left(\sum_i a^i x_i\right) \bmod m$ (polynom)

Chceme náhodnou hash. funkci

– z nějakého systému (množiny) funkcí

parametrizovaní:

$$\mathcal{H} := \{h_a \mid a \in \mathbb{Z}_p\}$$
$$h_a(x) := (ax) \bmod m$$
$$h_a : \mathbb{Z}_p \mapsto [m]$$

– Chceme nějaké vlastnosti

Df: Systém funkcí $\mathcal{H}$ z $\mathcal{U}$ do $[m]$ je c-univerzální pro $c > 0$

$$\equiv \forall x, y \in \mathcal{U}, x \neq y : \underset{h \in \mathcal{H}}{P}\left[h(x) = h(y)\right] \leq \frac{c}{m}$$

Prakticky to říká, že šance, že něco spadne do stejné přihrádky je $c * \frac{1}{\#\text{počet přihrádek}}$

Přehashování:

Sleduje se hustota/faktor naplnění a jakmile to přeteče celá tabulka se přepočítá do více přihrádek což ale stojí „čas"

$$\alpha := \frac{n}{m}$$

Chceme $\alpha \leq$ konst.

Například $m \to 2m$ přihrádek

$2 \quad 2^i$ do 2^{i+1} : $\Theta(n + 2^{i+1}) = \Theta(n)$

$\frac{n}{2^i} >$ konst, $\frac{n-1}{2^i} <$ konst $\Rightarrow$ $2^i < \frac{n}{\text{konst}}$, $2^i > \frac{n-1}{\text{konst}}$ $\Rightarrow$ 2^i je $\Theta(n)$

Jelikož ale mezi přehashováním proběhne in avg $\mathcal{O}(n)$ operací, lze to rozpočítat mezi operace na $\mathcal{O}(1)$

Příklad: Skalární součin nad tělesem $\mathbb{Z}_p$

$\mathcal{U} = \mathbb{Z}_p^d$, $a \in \mathbb{Z}_p^d$, přihrádky: $\mathbb{Z}_p$

$$\mathcal{H} = \{h_a \mid a \in \mathbb{Z}_p^d\}$$
$$h_a(x) = a \cdot x \text{ (v tělese)}$$
$$\left(\sum_i a_i x_i\right) \bmod p$$

Věta: Tento systém je 1-univerzální

Dk: pro $x \neq y$: $\underset{h \in \mathcal{H}}{P}\left[h(x) = h(y)\right] = P\left[ax = ay\right] \Rightarrow a \cdot (x - y) = 0$

$$\Updownarrow$$

$$\sum_i a_i (x_i - y_i) = 0$$

BÚNO $x_1 \neq y_1$

$$\left.\underbrace{a_1}_{\text{nevíme}} \underbrace{(x_1 - y_1)}_{\text{nenulová konstanta}} + \underbrace{\sum_{i \neq 1} a_i (x_i - y_i)}_{\text{konst}} = 0 \right\} \text{lineární rovnice}$$

Něco $*$ konst + konst = 0 $\Rightarrow$ $\exists!$ řešení a_1

$$P\left[a_1 \text{ je řešení}\right] = \frac{1}{p} \quad \boxed{\times}$$