

Dynamické lineární uspořádání

List Order Problem

- Chceme udržovat posloupnost prvků
- Operace:
 - Insert nového prvku za zadaný
 - Compare - odpoví, zda je x před y --- chceme v $O(1)$
 - možná také Delete (ne via glob. přestavba)

řeší se pomocí

List Labelling

- Posloupnost prvků, každému přidělena známka, známky rostou zleva doprava
- Insert, Delete mohou přeznačovat

① exponenciální rozsah značek, předem víme max. # prvků M

- první prvek dostane 2^M
 - druhý buď 0 nebo 2^{M+1}
 - každý další je přírůstek sousedů
- nikdy není třeba přeznačovat, vše $O(1)$ w.c.
- jednodušší: nejprve vyřadíme značky na 0 a 2^{M+1} , ostatní prvky se vkládají vždy někde mezi ně
- pozitivně pro $M = O(\text{word size})$

② polynomiální rozsah značek

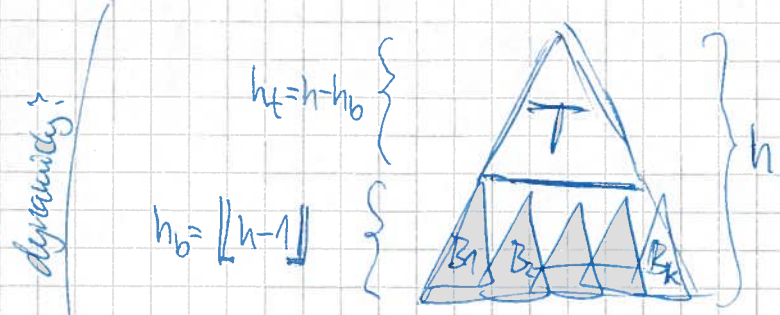
- ↳ prvky v ext. uzlech
- BVS, známka = posl. L/P na cestě z kořene do prvků - $O(\log)$ kóti → poly rozsah
- B+ stromy přepočítávají známky během rekonstrukce → $O(\log)$ amort. na Ins/Del
- ↳ prázdné, rotace jsou drahé

③ lineární rozsah - Ordered File Maintenance → pořadí ukládáme $O(\log^2 n)$ amort. nebo Packed Memory Array

- ↳ lze zrychlit indikací & bloky velikosti $O(\log n)$, v nich ①
 - ↳ ② nad bloky $O(n \log n)$ bloků
 - ↳ stojí $O(1)$ amort.
 - ↳ Insert v ① zrychlení $O(1/\log n)$ amort. ~~operaci~~ ve ②
 - ↳ ⇒ ② stojí $O(1)$ amort., ① také.
- label je dvojice, porovnáváme lexicograficky → Compare $O(1)$ w.c.
- ↳ [prázdné, 1 známka ve ② není $O(\log)$ dvojice, ale to lze udělat najednou]

Cache-Oblivious datové struktury

- I/O model, parametry B (velikost bloků), M (velikost cache)
 - c/I/O model - parametry uvažované, cache se obsluhuje optimálně
- úroveň počítáme jen část
- cache-aware (I/O): (a,b) -strom s $a, b \in O(B) \rightarrow O(\log n / \log B)$ I/O na operaci
 - cache-oblivious: staticky optimalizovaný BVS ve van Emde-Boasově uložení



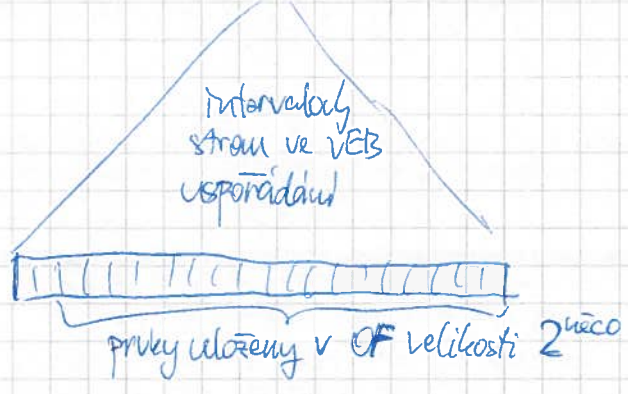
Nejprve T , pak $B_1 - B_k$
 vše rekursivně...

Věta: Průchod kořen - list vyžaduje $O(\log_B N)$ I/O

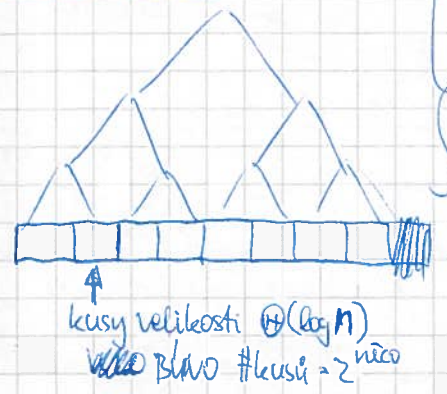
Náčrtek dle: "zaostříme" na úroveň rozkladu, když se stromu poprvé vejde do bloku → má velikost $O(\log B)$
 ⇒ na cestě takových prvků $O(\log n / \log B)$

* zlepšíme Ordered File s VEB uspoř. BVS:

- Find je plně v režii streamu
- Insert vloží do OF, to způsobí přečíslování nějakého intervalu klicí
⇒ důležitý update streamu



Ordered File



čistě konceptuální
úplný bin. strom
vnitřní vrchol ≈ interval

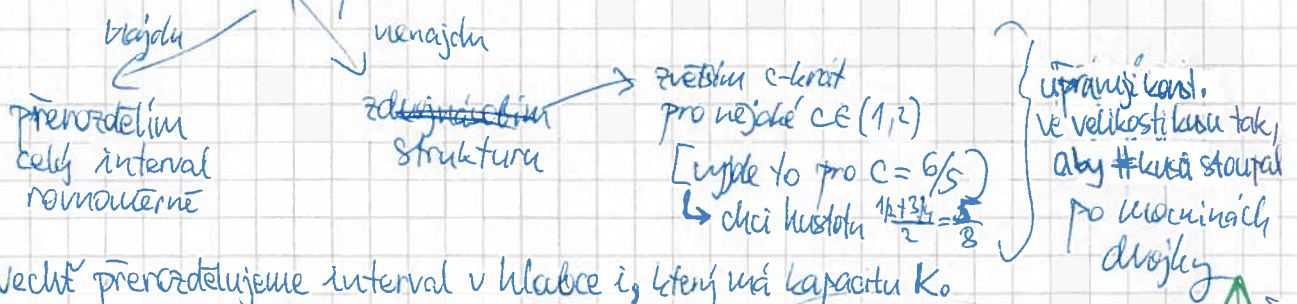
každý interval má kapacitu $\log n \cdot 2^{h-i}$
(h = hloubka stromu, i = hloubka vrcholu)
a hustoty $\rho = \#prvků / kapacita$

Standardní hustota $[\frac{1}{2} - \frac{i}{4n}, \frac{3}{4} + \frac{i}{4n}]$

v kořeni ($i=0$) $[\frac{1}{2}, \frac{3}{4}]$
v listech ($i=h$) $[\frac{1}{4}, 1]$
↓ čím výš, tím přesněji

Insert (Delete analogicky):

- vložíme do příslušného kusu ($O(\log n)$, úplně ho přepíšeme)
- pokud má stále std hustotu, končíme, jinak jdeme nahoru a hledáme první interval se std hustotou



Amortizace:

Nechť přerozdělujeme interval v hloubce i , který má kapacitu K_0 . Jeho ~~$\rho \leq \frac{3}{4} + \frac{i}{4n}$~~ $\rho \leq \frac{3}{4} + \frac{i}{4n}$, alespoň 1 syn má $\rho > \frac{3}{4} + \frac{i+1}{4n}$

zvětší amortizaci
různé velikosti
celé struktury

Přerozdelení stojí $\Theta(k)$
↓
Vytváříme $\Theta(h) \leq O(\log n)$ jednoduše prvky

+ prvek přispěje na přerozdelení v celkem $\log n$ úrovních, tedy celkově $O(\log^2 n)$

! velikost kusu nastavíme tak, aby se změna hustoty $\approx 1/4n$ projevila přidáním/ubráním aspoň 1 prvku
... i po započtení

↑ také velikost bloku musí být $\log n$ a $8 \log n$

Cache-oblivious:

Hledání + přerozdelení prvku jsou 2 protažené scény (popřední + pozpátku)
⇒ ~~klas-~~ $O(k/B)$ bloků, $O(\log n / B)$ na prvek.

Zpět k C/O strukturám

Insert : $O(\log^2 n)$ času, $O(\log n / B)$ I/O na update OFM
 + $O(\text{#změněných prvků} \cdot OFM/B + \log n / \log B)$ na další update VEB] \rightarrow amortizované se sčítají do ceny OFM + $\log n / \log B$

Find : $O(\log n)$ času, $O(\log n / B)$ I/O na VEB

Zrychlení \dots jako obvykle indirekci \mathbb{C} Fragmenty velikosti $F = \Theta(\log n)$ nad jejich reprezentantů převodů struktura

- uvnitř fragmentu vše v čase $O(\log n)$ a $O(\log n / B)$ I/O
 - jednou za amort. $O(1/F)$ operací, provedeme $O(1)$ operací na pův. struktuře
 \rightarrow jedna stojí amort. $O(\log n)$ času a $O(\log n / \log B)$ I/O
 [$\log n / B \approx OFM + \log n / \log B \approx VEB$]
 - dotazy : nejprve ve VEB, pak selektivně ^{1/4} fragmenty : $O(\log n)$ času, $O(\log n / \log B)$ I/O
 - jednou za čas globální přestavba, abychom udrželi $F = \Theta(\log n)$ apod.
- \rightarrow Asymptoticky stejně rychlé jako C/A B-strony, ale je to C/O.