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 (lze via glob. přestavba)

řeší se pomocí ↓

List Labelling – Posloupnost prvků, každému přidělena značka, značky 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 průměr sousedů

} nikdy není třeba přeznačovat, lze $O(1)$ w.c.

použitelné pro $M = O(\text{word size})$

② polynomiální rozsah značek
- BVS, značka = pos. L/P na cestě z kořene do prvku – $\Theta(\log n)$ bitů → poly rozsah
- BB-α stromy přepočítávají značky během rekonstrukcí → $\Theta(\log n)$ amort. na Ins/Del
  ↳ pozor, rotace jsou drahé!

③ Lineární rozsah – Ordered File Maintenance → pořád děláme $O(\log^2 n)$ amort
neboli Packed Memory Array

lze zrychlit indirekcí: bloky velikosti $\Theta(\log n)$, v nich ①
② nad bloky ($\Theta(n/\log n)$ bloků)
stojí $O(1)$ amort.

Insert v ① způsobí $O(1/\log N)$ amort. operací ve ②
⇒ ② stojí $O(1)$ amort., ① taktéž.

label je dvojice, porovnáváme lexikograficky → Compare v $O(1)$ w.c.

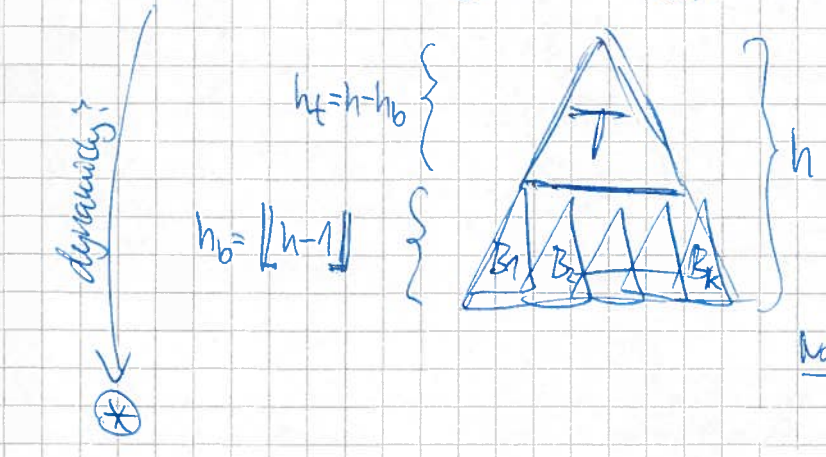
[pozor, 1 změna ve ② umí $\Theta(\log n)$ dvojic, ale to lze udělat najednou]

Cache-Oblivious datové struktury

- I/O model, parametry B (velikost bloku), M (velikost cache)
- C/O model – parametry neznáme, cache se obsluhuje optimálně

} Dílko počítáme jen čtení

- cache-aware (I/O): (a,b)-strom s $a,b \in \Theta(B)$ → $\Theta(\log n / \log B)$ I/o na operaci
- cache-oblivious: statický úplný BVS ve van Emde-Boasově uložení



Nejprve T, pak B_1 — B_k (lze rekurzivně)

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

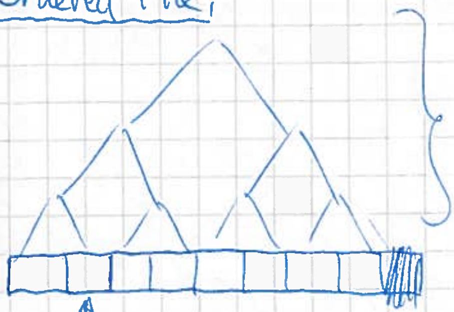
Počítek Dk: "zaostříme" na úroveň rozkladu, kdy se strom poprvé vejde do bloku
→ jsou uloženy $\Theta(\log B)$
⇒ na cestě takových projdeme $\Theta(\log N / \log B)$

(*) ↳ zkřížíme Ordered File s vEB uspoř. BVS:

- Find je plně v režii stromu
- Insert vloží do OF, to způsobí přečíslování nějakého intervalu klíčů ⇒ dávkový update stromu

prvky uloženy v OF velikosti $2^{\text{něco}}$

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 hustotu ϱ = #prvků / kapacita

kusy velikosti $\Theta(\log n)$, #kusů = $2^{\text{něco}}$

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

v kořeni ($i=0$) $\left[\frac{1}{2}, \frac{3}{4}\right]$

v listech ($i=h$) $\left[\frac{1}{4}, 1\right]$

čím výš, tím přísnější

Insert (Delete analogicky):

- vložím do příslušného kusu ($O(\log n)$, úplně ho přepíši)
- pokud má stále std. hustotu, končím, jinak jdu nahoru a hledám první interval se std. hustotou
  - najdu → přerozdělím celý interval rovnoměrně
  - nenajdu → ~~zdvojnásobím~~ zvětším strukturu c-krát pro nějaké $c \in (1,2)$ [vyjde to pro $c = 6/5$ ↳ chci hustotu $\frac{1/2 + 3/4}{2} = \frac{5}{8}$]

  upravuji konst. ve velikosti kusu tak, aby #kusů stoupal po mocninách dvojky

Amortizace: Nechť přerozdělujeme interval v hloubce i, který má kapacitu K. Jeho $\varrho \leq \frac{3}{4} + \frac{i}{4h}$, alespoň 1 syn má $\varrho > \frac{3}{4} + \frac{i+1}{4h}$

↳ po minulém přerozdělení otec měl ϱ stejnou jako otec ⇒ vzrostla o alespoň $1/4h$ ⇒ v podstromu přibylo alespoň $\Theta(K/h)$ prvků

zvlášť amortizuji změny velikosti celé struktury

Přerozdělení stojí $\Theta(K)$
⇓
Účtujeme $\Theta(h) \leq O(\log n)$ jednomu prvku
⇓
∀ prvek přispěje na přerozdělování v celkem $\log n$ úrovních, tedy celkově $O(\log^2 n)$

! velikost kusu nastavíme tak, aby se změna hustoty o $1/4h$ projevila přidáním/ubráním aspoň 1 prvku 😊 … i po zaokrouhlení

takže velikost bloku být musí $4 \log n$ a $8 \log n$

Cache-oblivious: Hledání intervalu + přerozdělování prvků jsou 2 prokládané scany (popředu + pozpátku) ⇒ $O(K/B)$ bloků, $O(\log^2 n / B)$ na prvek.

Zpět k C/O strukturám

Insert: $O(\log^2 n)$ času, $O(\log^2 n / B)$ I/O na update OFM
+ $O(\#\text{změněných prvků v OFM}/B + \log n/\log B)$ na dávkový update vEB] → amortizováno se schová do ceny OFM + $\log n/\log B$

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

Zrychlení ... jako obvykle indirekcí & Fragmenty velikosti $F = \Theta(\log n)$, nad jejich reprezentanty původní struktura

- uvnitř fragmentu vše v čase $O(\log n)$ a $O(\log n/B)$ I/O
- jednou za amort. $O(1/F)$ operací, provedené $O(1)$ operací na pův. struktuře
  ⇒ jedna stojí amort. $O(\log n)$ času a $O(\log n/\log B)$ I/O
  [$\log n/B$ ≈ OFM + $\log n/\log B$ ≈ vEB]
- dotazy: nejprve ve vEB, pak sekvenčně $\log(n)$ fragmentů: $O(\log n)$ času, $O(\log n/\log B)$ I/O
- jednou za čas globální přestavba, abychom udrželi $F = \Theta(\log n)$ apod.

→ Asymptoticky stejně rychlé jako C/A B-stromy, ale je to C/O.