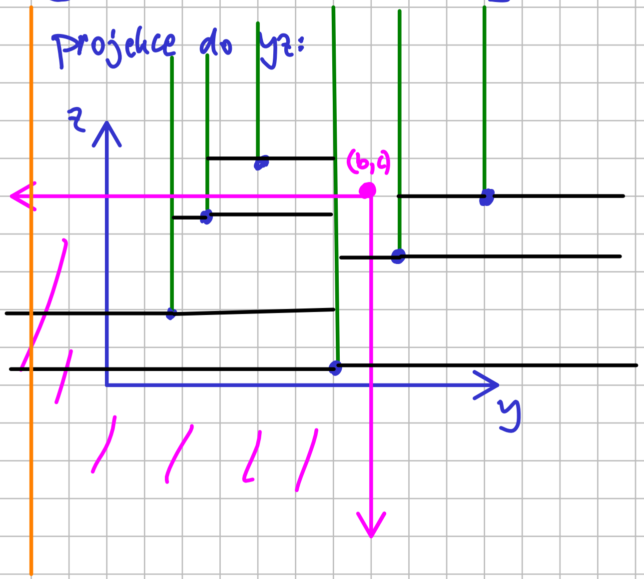
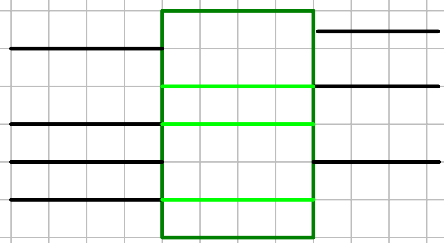
Range queries v $\mathbb{R}^3$: $[a_1,a_2]\times[b_1,b_2]\times[c_1,c_2]$ — $O(\log n + k)$ (↑ # bodů výsledku)

↳ vyjmenovat body

① $\mathbb{R}\times(-\infty,b]\times(-\infty,c]$

projekce do yz:





hive graph

dotaz $O(\log n + k)$ (log n → 1)

prostor $O(n)$

build $O(n\log n)$

amortizace:

pokračování účtují koncům } celkem $O(n)$

+1 za oblast } $O(n)$

② $[a_1,a_2]\times(-\infty,b]\times(-\infty,c]$

- intervalový strom podle x
- pro ∀ vrchol (x-ový pás) ①

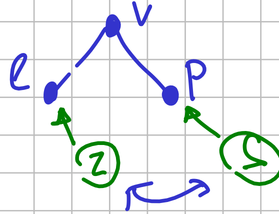
dotaz $O(\log^2 n + k)$ ($\log^2 n$ → log)

prostor $O(n\log n)$

build $O(n\log^2 n)$

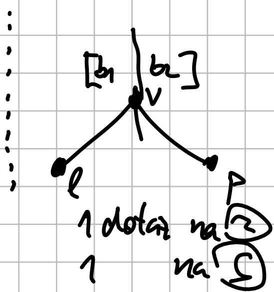
③ $[a_1,a_2]\times[b_1,b_2]\times(-\infty,c]$

- intervalový strom podle y



③' = ② podle y

$[a_1,a_2]\times[b,+\infty)\times(-\infty,c]$



dotaz $O(\log n + \log^2 n + k)$ (↑ zde, ↑ ②; $\log^2 n$ → log)

prostor $O(n\log^2 n)$

build $O(n\log^3 n)$

④ $[a_1,a_2]\times[b_1,b_2]\times[c_1,c_2]$

- tentýž trik v z

dotaz $O(\log n + \log n + \log^2 n + k)$ (↑ zde, ↑ ③, ↑ ②; $\log^2 n$ → log)

prostor $O(n\log^3 n)$

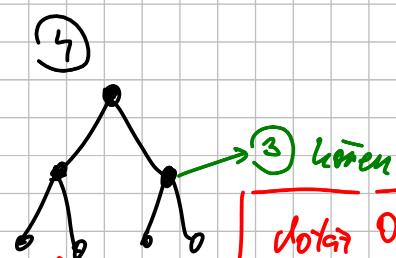
build $O(n\log^4 n)$

Aplikujeme Fr. C.

- v ∀ vrcholu stromů v ④ – ②
  přidám z-sorted seznam bodů v pásu

protože hledání podle z : $O(\log n)$

\+ $O(\log n)$ FC kroků } $O(\log n + k)$



v $\mathbb{R}^d$

dotaz	$O(\log^{d-2} n)$
prostor	$O(n\log^d n)$
build	$O(n\log^{d+1} n)$

Non-blocking binary search trees

[Ellen et al. '10]

Cíl: paralelní BST
↳ find, insert, delete

Vlastnosti:
- potřebujeme CAS + atomická registry
- lock-free (≥1 proces uspěje)
  ↳ find je read-only a skoro wait-free
- bez vyvažování

Idea: jednoduchý stromy, zamykání + pomáhání

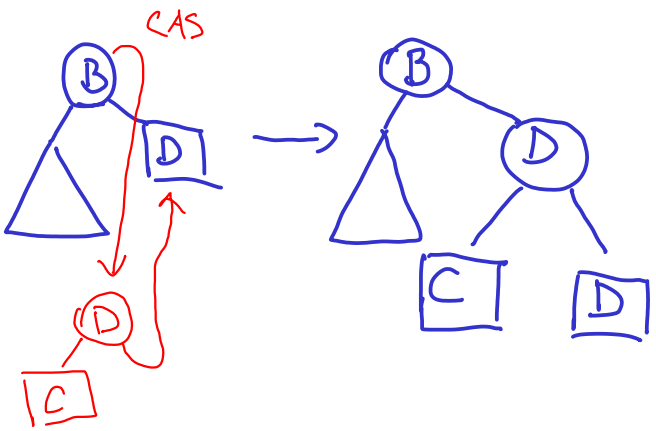
Struktura

- BST s hodnotami jen v listech
- zakazujeme duplicitní hodnoty
  ↳ ve vnitřních vrcholech ale duplicitní klíče jsou
- vrchol obsahuje
  → klíč
  → pointery na děti
  → jedno slovo na značku + pointer

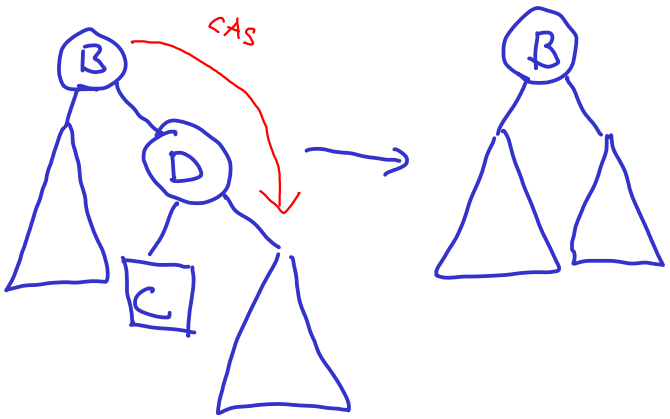
Updaty

: změny se dají jen u listů

insert(C):

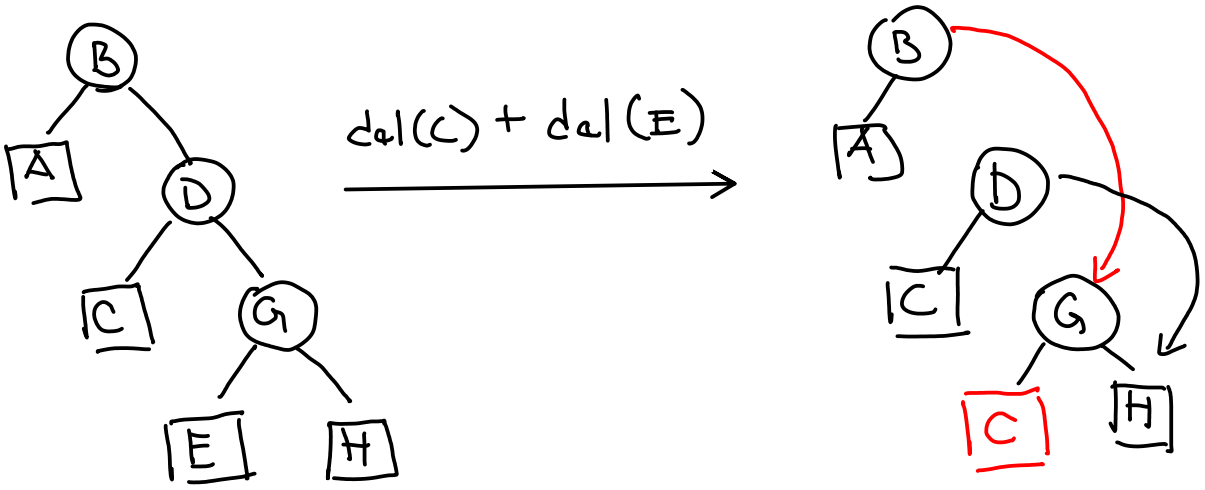


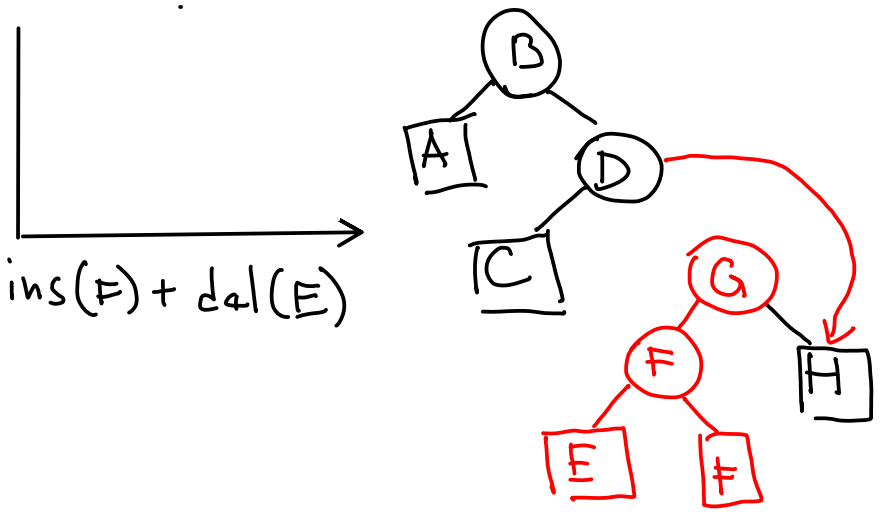
dalete(C)



: insert i delete lze udělat 1 CASem

Problém: delete lze ostatním do zad





Řešení: "Zamykání" pomocí značek

Tři značky:

- iflag ~> otec x (p) při insert(x)
- dflag ~> dědeček x (g) při delete(x)
- mark ~> otec x při delete(x)

+ clean

~> update se nejdřív pokusí označkovat vrchol
→ jen při úspěchu pokračuje

→ vznik značky = linearizační bod

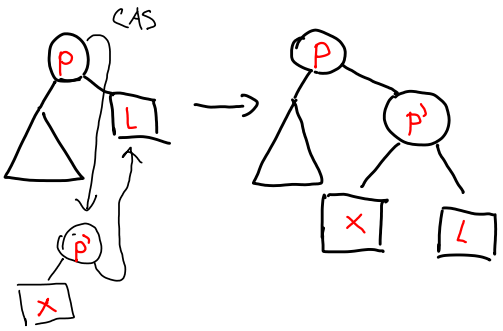
? Jak zabránit blokování ?

→ procesy si pomáhají

→ spolu se značkou proces uloží pointer na informace nutné k dokončení operace

⇒ nemůžu dokončit svoji op. kvůli značce
→ dokončím operaci, která mě blokuje a začnu odznova

Find(x):
jako v obyč. BST, ignoruje všechny značky a spol.



Insert(x):

forever:

find (x) ~ p budoucí otec

x je na stromě => return

x označkovaná => help(x); continue

info ← pointer na informace k dokončení

CAS: označkuj p (iflag, info), pokud bylo (clean, NULL)

↳ fail: help(x); continue

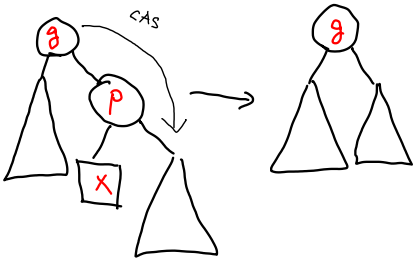
↳ succes:

CAS: změň syna v p z l na p'

CAS: označkuj p (clean, NULL), pokud bylo (iflag, info)

return

help pro insert



Delete(x)

- analogicky insertu
  - ale značkuji g i p
- najdřív dostane g značku dflag (+info)
- následně p dostane značku mark a stejné info jako
- při naražení pošlu na příslušný vrchol help
  - → help může rekurzit
- navíc naražení při mark vyvolá rollback a nový pokus o delete