L4

13.3.17

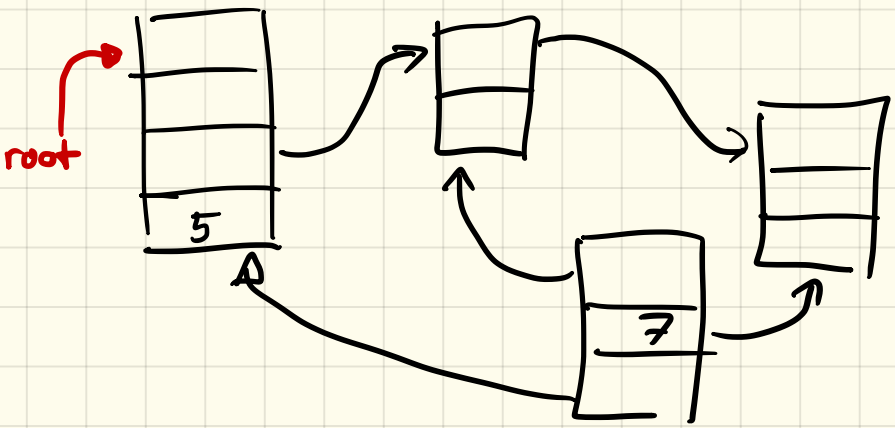
∃ lecture on 02.05 and 04.05

Topic Persistence / Temporal DS

Model : So far we assumed RAM machine

→ Now **pointer machine**

our DS consists of nodes with ~~feelings~~ fields



In C NODE = STRUCT

→ O(1) FIELDS

Allowed operations

- x = new node
- x = y.field
- x.field = y
- x = y+2 (manipulate data)
- destroy x (if no pointers to x)

Temporal DS

Persistence → never destroy old version
↳ update → makes new version

Retroactivity: next week (use time travel)

Persistence

presentation might change, behavior is preserved.

- partial : only latest update
- full : update any version
- confluent : merging two version
- functional : ~~implement in HASKELL~~ never modify versions

today

blackhole : never store updates

Partial Persistence:

superposition of a and v.

Driscoll et. al. 1982 :

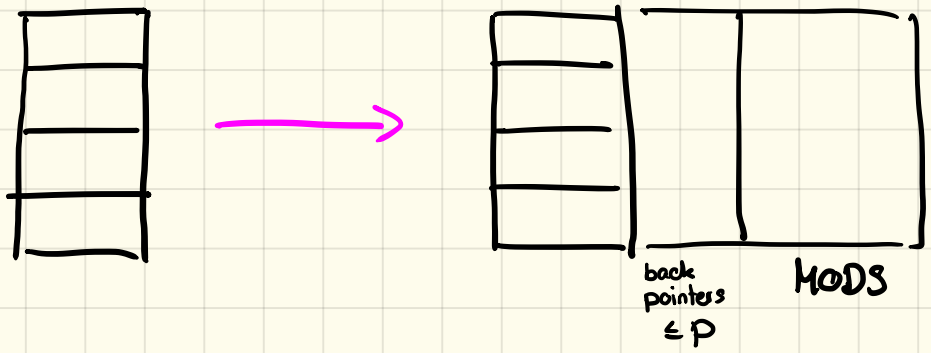
∀ pointer machine DS with

indegree ≤ p = O(1) (# pointers to any node)

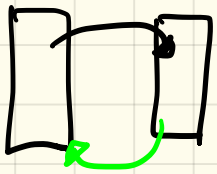
can be made partially persistent with

time & space

- O(1) amortized multiplicative overhead
- O(1) space per update



① NODE stores back pointers
- latest version only



② Allow ≤ 2p MODS

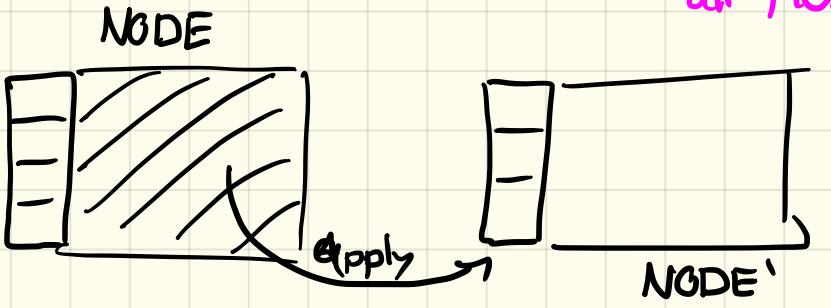
MOD = (VERSION, FIELD, VALUE)

≅ History of $O(1)$ versions.

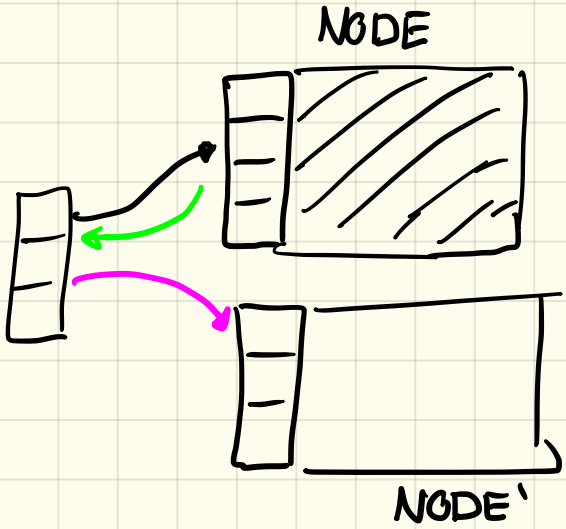
UPDATE:

If NODE not full
- add MOD
  if switching pointers → update back pointers

if full
- create new NODE (NODE' = NODE with all MODS applied)



- change <u>back pointers</u> NODE → NODE'
  - ↳ follow NODE' pointers
- change pointers to NODE'
  - ↳ follow NODE' backpointers and appr. change <u>recursively</u>



Q: Does it terminate?

<u>Potential Analysis</u>:

$$\Phi = c \cdot \sum_{\text{in latest version}} \#\text{ mods in nodes}$$

↑ large constant.

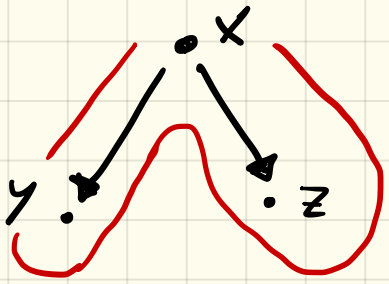
amortized cost =

$$\underbrace{O(1)}_{\text{compute + modify}} + \underbrace{O(1)}_{\text{potential}} + \underbrace{p \cdot \text{recursion} - 2 \cdot c \cdot p}_{\text{or this is } 0} \leq 2c$$

$\leq c$

Full Persistence (Driscoll 89)

Problem We have a tree of versions?



in order traversal

$b_x\ b_y\ e_y\ b_z\ e_z\ e_x$

Solution Linearize versions

Note $b_v < b_w < e_w < e_v \iff$ v is ancestor of w

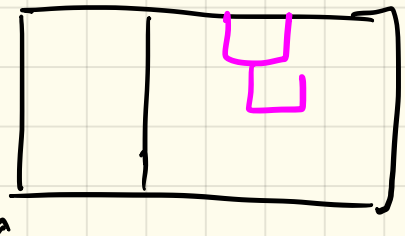
Maintaining ordered list:

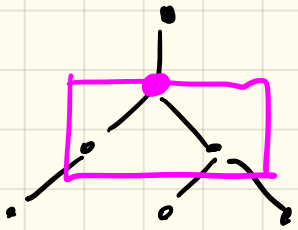
Insert, delete $O(1)$

Order query $O(1)$

amortized $O(1)$

$\longrightarrow$ allows us to maintain tree of NODES





(1.) VERSIONS ROOT VERSION,
SUBTREE OF VERSIONS

- Query: Scan all MODS ✓

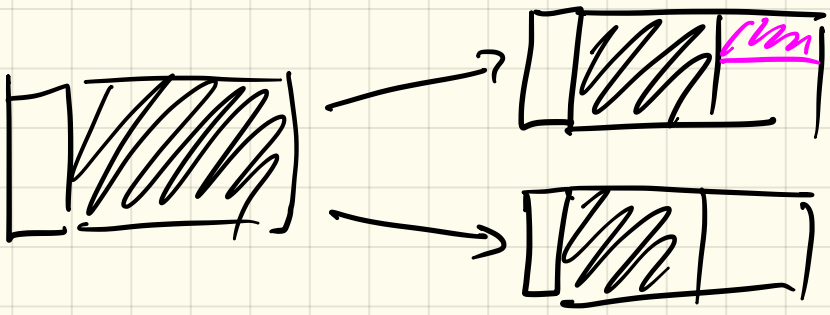
(2.) Backpointers are always stored.

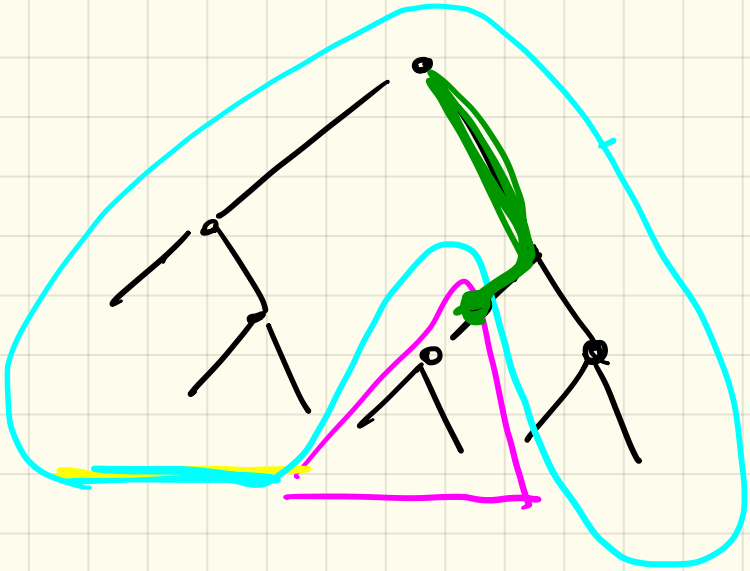
p - max in/out degree

d - # fields

store $2(d+p+1)$ mods
↑ or some other constant

Problem: Constantly discharging (Huge cost)





Split of NODE