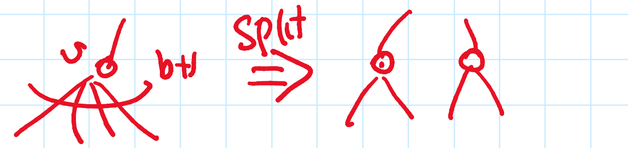
$\Rightarrow$ height $\leq \log_a n$

query: still $O(\log n)$

insert: whenever $\deg(v) = b+1$ for some v:

split v into 2 nodes of deg $\sim \frac{b+1}{2}$

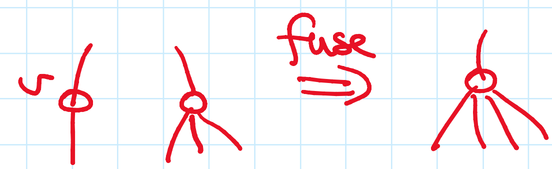


$\Rightarrow$ $O(\log n)$ splits (up a path)

$\Rightarrow$ $\boxed{O(\log n)}$ time.

delete: whenever $\deg(v) = a-1$ for some v:

fuse v with sibling & split if $\deg \geq b$

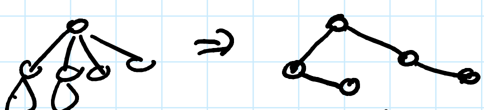


$\Rightarrow$ $O(\log n)$ fuses

$\Rightarrow$ $\boxed{O(\log n)}$ time

Rmk. useful in external memory: "B-tree"

- 2-3-4 trees related to red-black trees



- related to "skip lists" & rand.

Q: insert & delete in $O(1)$ time?? (given ptr to elem)

Def We say op takes $O(T)$ amortized time

op A: T_A

op B: T_B

Def We say op takes $O(T)$ amortized time if any sequence of n ops takes total worst-case time $O(nT)$.

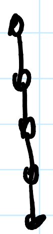
starting with initially empty DS

op A: T_A
op B: T_B

any seq of n_A op A's n_B op B's has total cost $O(n_A T_A + n_B T_B)$.

Claim (a,b)-tree has $O(1)$ amortized insert time if no deletions.

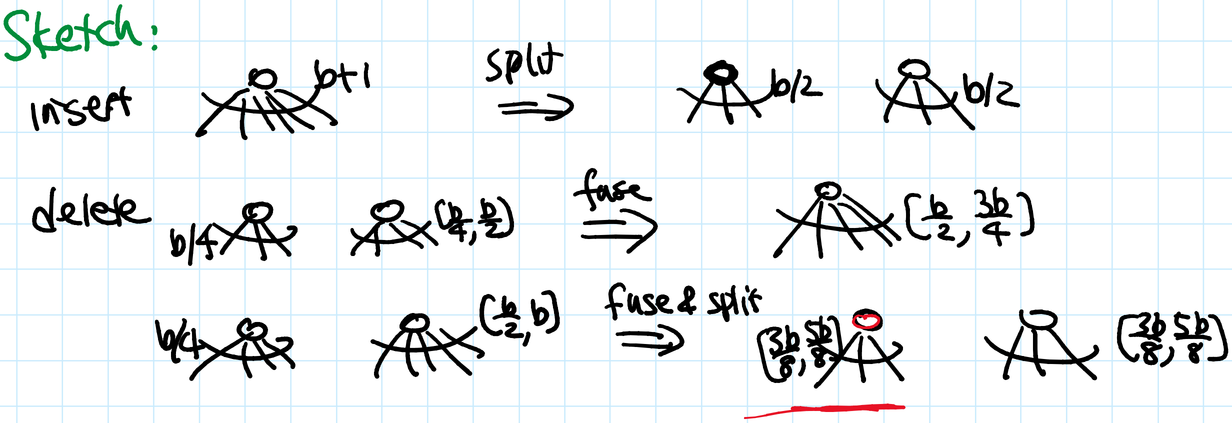
Pf: Consider any seq of n ops.



$$\text{total time} = O(n + \#\text{splits}) \le O(n + \#\text{internal nodes}) \le \boxed{O(n)}. \quad \square$$

Claim (a,b)-tree has $O(1)$ amortized insert & delete time if $a \approx \frac{b}{4}$.

Pf Sketch:



$$\#\text{ splits/fuses at depth } i \le \frac{\#\text{ split/fuses at depth } i+1}{b/8}$$

$$N_i \le \frac{N_{i+1}}{b/8}$$

h = height of tree

$$\Rightarrow \quad N_i \le \frac{\cancel{N_h}\; n}{(b/8)^{h-i}}$$

$$\Rightarrow \quad \text{total } O\left(\sum_{i=0}^{h-1} \frac{n}{(b/8)^{h-i}}\right) = \boxed{O\left(\frac{n}{b}\right)}$$

Method 4 Weight-balanced trees or BB[α] tree
(Nievergelt - Reingold '70)

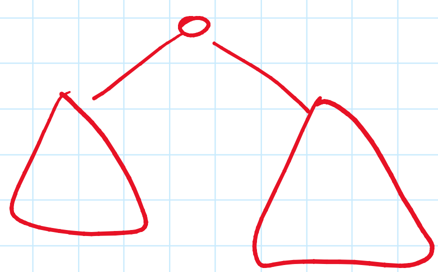
(very general)

Invariant: for each node v,

size of subtree rooted at v

$$\text{size}(\text{left}(v)) \le \alpha\, \text{size}(v)$$
$$\text{size}(\text{right}(v)) \le \alpha\, \text{size}(v)$$

param α with $\frac{1}{2} < \alpha < 1$.



$\alpha^h n \approx \text{const}$

$\left(\frac{1}{\alpha}\right)^h \approx n$

$\Rightarrow$ height $O(\log_{1/\alpha} n)$

whenever invariant is violated at some node v,
rebuild subtree at v
with perfectly balanced tree

worst-case: $O(n)$ BAD!
amortized?

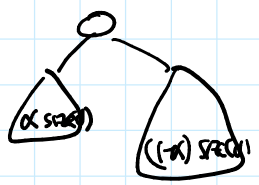
Define potential $\Phi = \sum \underbrace{\left|\text{size}(\text{left}(v)) - \text{size}(\text{right}(v))\right|}_{\text{imbalance}(v)}$

In insert/delete, Φ increases by $O(\log n)$.

if rebuild at v,

runtime is $O(\text{size}(v))$

but Φ decreases by



$$\geq \big(\alpha\,\text{size}(v) - (1-\alpha)\,\text{size}(v)\big) - 0$$

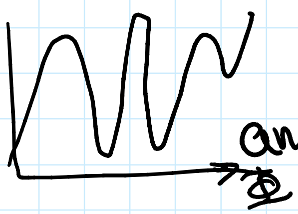
(Φ before)

$$= \Omega(\text{size}(v)).$$

total rebuild time $\le O(\text{total decrease in } \Phi)$

$\le O(\text{total increase in } \Phi)$

$\le \boxed{O(n \log n)}$.



amortized cost $\le \boxed{O(\log n)}$. □