\[ \Rightarrow \text{height} \leq \log n \]

query: still \( O(\log n) \)

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

split \( v \) into 2 nodes of \( \deg \approx \frac{b+1}{2} \)

\[ \Rightarrow \quad O(\log n) \text{ splits (up a path)} \]

\[ \Rightarrow \quad O(\log n) \text{ time.} \]

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

fuse \( v \) with sibling & split if \( \deg > b \)

\[ \Rightarrow \quad O(\log n) \text{ fuses} \]

\[ \Rightarrow \quad O(\log n) \text{ time} \]

Rumk - useful in external memory: "B-tree"

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

- related to "skip lists" & random

Q: insert & delete in \( O(1) \) time??

(given ptr to elem)

Def: We say op takes \( O(T) \) \text{ amortized time}
**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.

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

**PF:** Consider any seq of $n$ ops.

$$\text{total time} = O(n + \#\text{splits})$$

$$\leq O(n + \#\text{internal nodes})$$

$$\leq O(n).$$

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

**PF Sketch:**

- **Insert**: $\Rightarrow$ \\
- **Delete**: $\Rightarrow$

\[
\# \text{splits/fuses at depth } i \leq \frac{\# \text{splits/fuses at depth } i+1}{b/8}
\]

\[
N_i \leq \frac{N_{i+1}}{b/8}
\]

$h = \text{height of tree}$
\[ N_i \leq \left( \frac{\alpha^n}{(b/8)^{n_i}} \right) \]

\[ \text{total } O\left( \sum_{i=0}^{n-1} \left( \frac{\alpha^n}{(b/8)^{n_i}} \right) \right) = O(n) \]

**Method 4**  
Weight-balanced trees or BB[\alpha] tree  
(Nievergelt–Renzigold '70)

(very general)

**Invariant:** for each node \( v \),
- \( \text{size(left}(v)) \leq \alpha \text{size}(v) \)
- \( \text{size(right}(v)) \leq \alpha \text{size}(v) \)

Site of Subtree rooted at \( v \)

\[ \alpha^{\text{height}(v)} \geq \text{const} \]
\[ \frac{\alpha^n}{(b/8)^{n_i}} \approx n \]

\[ \Rightarrow \text{height } O\left( \log \frac{n}{\alpha} \right) \]

Whenever invariant is violated at some node \( v \),
rebuilt subtree at \( v \)
with perfectly balanced tree

worst-case: \( O(n) \quad \text{BAD!} \)

amortized?

Define potential \( \Phi = \sum \left( \text{size(left}(v)) - \text{size(right}(v)) \right) \)
In insert/delete, \( \Phi \) increases by \( O(\log n) \).

If rebuild at \( u \), runtime is \( O(\text{size}(u)) \).

But \( \Phi \) decreases by

\[
\geq (\alpha \cdot \text{size}(v) - (1 - \alpha) \cdot \text{size}(u)) - O(\Phi_{\text{before}})
\]

\[
= \Omega(\text{size}(v)).
\]

Total rebuild time \( \leq O(\text{total decrease in } \Phi) \)

\( \leq O(\text{total increase in } \Phi) \)

Amortized cost \( \leq O(\log n) \).