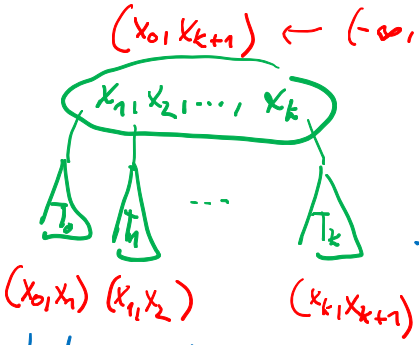
(a,b)-trees



k keys $x_1 < x_2 \dots < x_k$

k+1 children

$\forall y \in T_i : x_i < y < x_{i+1}$

multiway search tree

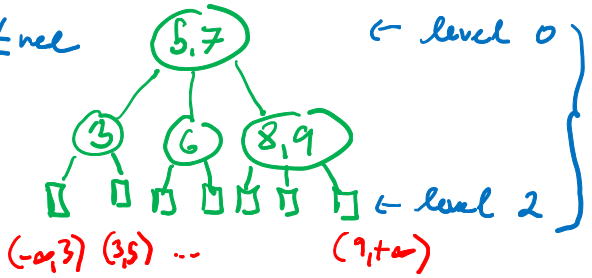
- internal node:
- external node (leaf):

no keys ... empty open interval between keys

Def: An (a,b)-tree for parameters $a \geq 2$, $b \geq 2a-1$ is a MST s.t.

1) every internal node has $a \leq$ #children $\leq b$ (except root has $2 \leq$ #children $\leq b$),
2) all external nodes are in the same level.

Ex: (2,3)-tree ← smallest possible



height $h = 2$

An (a,b)-tree of height h has $2a^{h-1} - 1 \leq n \leq b^h - 1$ keys.

($2a^{h-1}-1$: for the root min # children; $b^h - 1$: #leafs $= n+1$)

$$\Rightarrow \log_b(n+1) \leq h \leq \left(\log_a \frac{n+1}{2}\right) + 1$$

(if we care about the choice of a,b)

$$\Omega(\log_b n) = h = O(\log_a n) = O(\log n / \log a)$$

FIND(x) ... similar to BST, use binary search of keys in a node

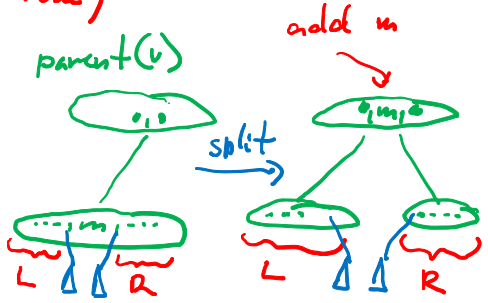
$\Rightarrow O(\log b)$ per node, $h = O(\log n / \log a) \Rightarrow O\left(\log n \frac{\log b}{\log a}\right)$ time

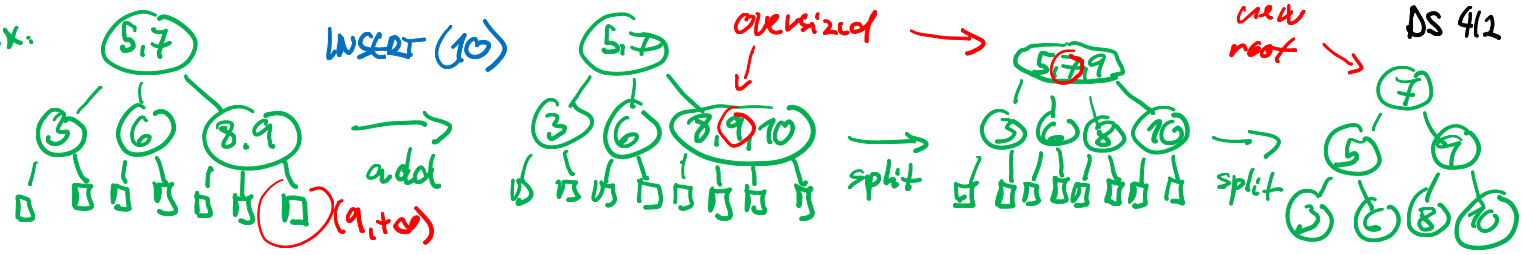
$= O(\log n)$ if $b = \text{poly}(a)$

INSERT(x): $v :=$ parent(FIND(x)) (external node)

(assuming x is new)

- add x to v (+ new external node)
- if v oversized (i.e. b keys)
  - split $v \to (L, m, R)$ (m: median)
  - $x := m$, $v :=$ parent(v)



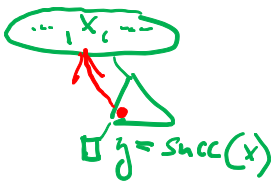
Ex. 

Is it correct? v oversized = b keys $\Rightarrow$ L, R have $\lfloor (b-1)/2 \rfloor, \lceil (b-1)/2 \rceil$ keys $\geq a-1$ as $b \geq 2a-1$ ✓ (that is why it is needed)

split in $O(b)$ $\Rightarrow$ INSERT in $O\left(b \frac{\log n}{\log a}\right)$

DELETE(x): (assuming x is there)

if v := FIND(x) not in the last internal level



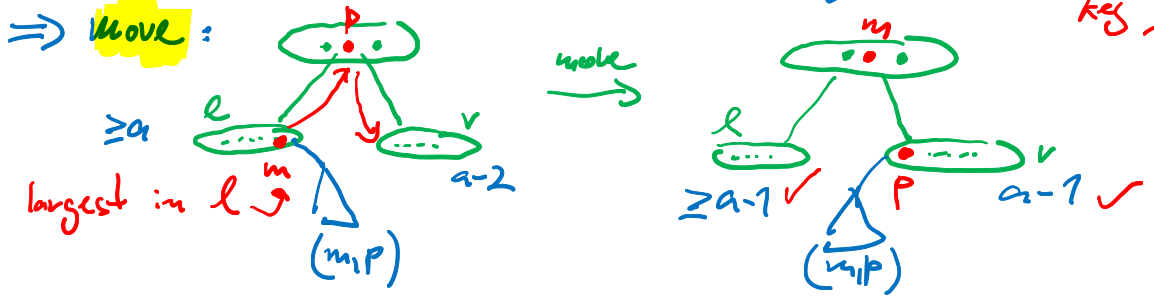
replace x with y := succ(x), DELETE(y)

remove x from v

1) if v not undersized (i.e $\geq a-1$ keys) $\Rightarrow$ DONE

2) if (left) sibling ℓ of v has $\geq a$ keys (some spare key)

$\Rightarrow$ **move**:

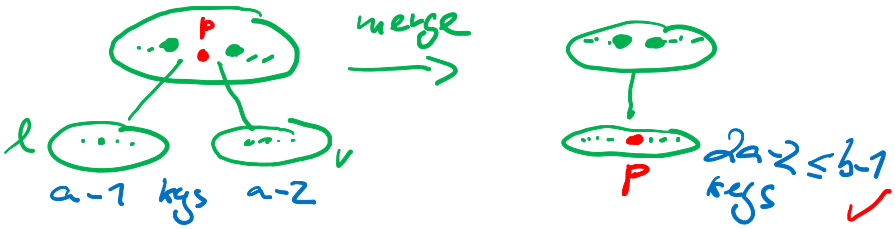


3) if (left) sibling ℓ of v has $a-1$ keys (no key to spare)

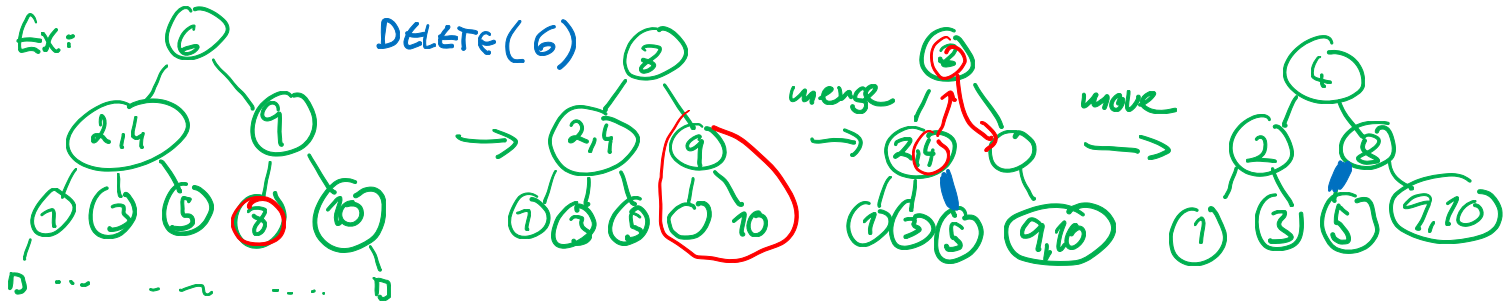
$\Rightarrow$ **merge**:

$x := p$

$v := parent(v)$



Ex: DELETE(6)



merge in $O(b)$ $\Rightarrow$ DELETE in $O\left(b \frac{\log n}{\log a}\right)$

Note: split oversized root $\Rightarrow$ increase height

merge children of root with a single key $\Rightarrow$ decrease height

Choice of parameters

FIND in $O(\log n \frac{\log b}{\log a})$, INSERT / DELETE in $O(\log n \frac{b}{\log a}) \Rightarrow$

b small ($b = 2a-1$ or $b = 2a$) $\Rightarrow$ FIND in $O(\log n)$, INSERT / DELETE in $O(\log n \frac{a}{\log a})$

$\Rightarrow$ a small: $(2,3)$-trees or $(2,4)$-trees good for RAM

in practice: 1 node $\sim$ 1 memory block (when read is more expensive than search in memory)

Ex: 4 kB block, 32-bit keys, 32-bit pointers

2^{12} B, 2^3 B $\Rightarrow$ $(256, 511)$-tree

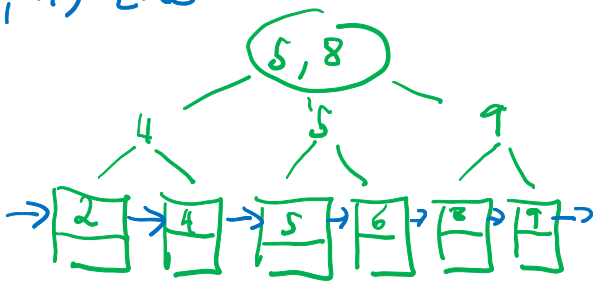
$2a^{h-1} \le n \le b^h - 1$ $\Rightarrow$ for $h=4$ $n > 33.5$ mil.

Ex: cache 64 B block (cache line), 32-bit keys, 32-bit pointers

2^6 $\Rightarrow$ $(4,7)$-tree

Variants

- B-tree of order n $\sim$ $(\lceil \frac{n}{2} \rceil, n)$-tree
- 1 key + data in external nodes, minima of subtrees in internal



- B^+-tree: —"— + pointers to successors
- B^*-tree of order n $\sim$ (not exactly) $(\lceil \frac{2n}{3} \rceil, n)$-tree

  INSERT: 2 full siblings $\Rightarrow$ split into 3 nodes
- RB-tree $\sim$ $(2,4)$-tree

  $\uparrow$ under some correspondence

(Amortized) modification costs

(sometimes)

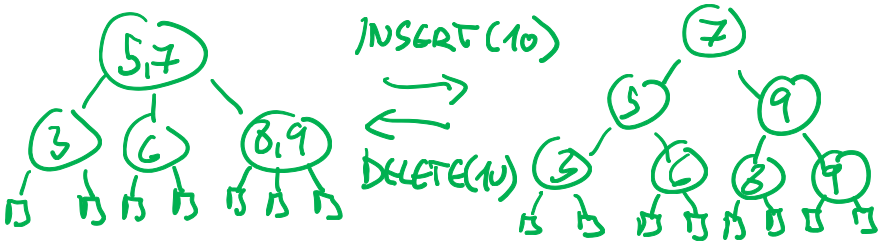
motivation: write more expensive than read (FIND can be neglected)

modifications: add / remove key, split, merge, move => O(1) real costs
1 3 3 3 # modified nodes

Thm: Any sequence of m INSERTS on initially empty (a,b)-tree does O(m) modifications.

Pf: # total splits ≤ # final internal nodes ≤ m => O(m) for splits
+ O(m) for add key => total modifications = O(m) □

(a, 2a-1)-tree cannot have O(1) amortized modification cost for mixed INSERTs and DELETEs.



Ω(log n) splits/merges
=> needs "breathing space"

... to be continued