

# B-Trees, Part II

Note Title

10/30/2007

## Review of Properties

Extension  
of  
INORDER

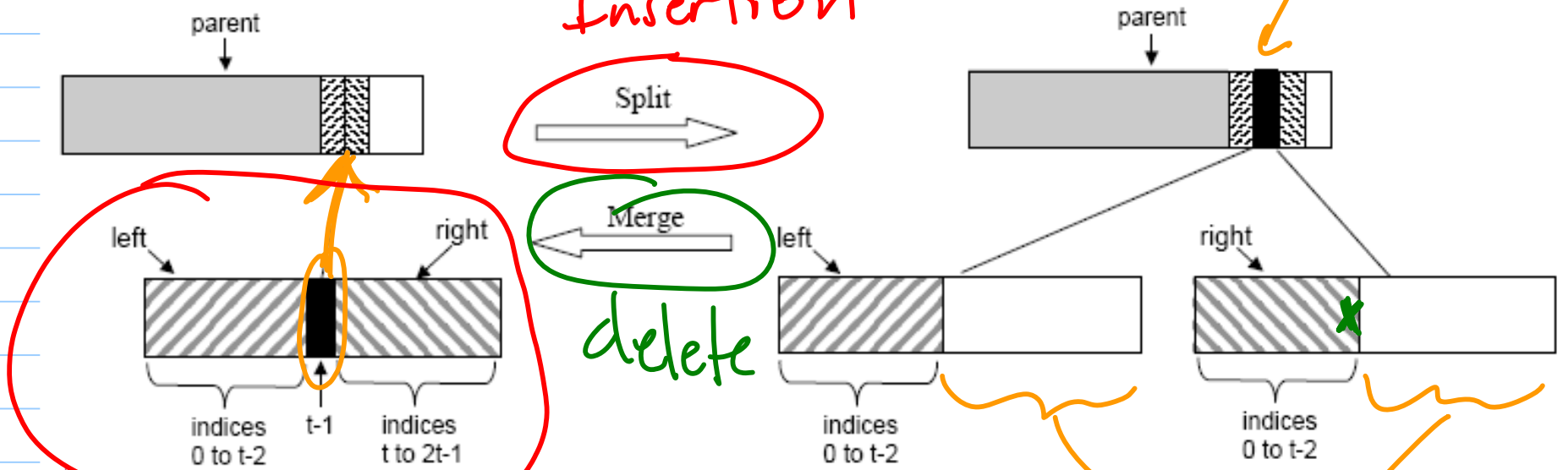
Let  $t$  be the order of the B-tree  
(parameter given to constructor)

**BALANCED** - all leaf nodes are at same height. So an internal node with  $x$  elements has  $x+1$  non-empty children

**NODE UTILIZATION** - With the exception of root all nodes have  $\geq t$  children ( $\neq$  so  $\geq t-1$  elements). The root has  $\geq 2$  children. All nodes have  $\leq 2t$  children ( $\neq$  so  $\leq 2t-1$  elements)

# Split (+ Merge)

Insertion



$2t-1$   
elements

$2t$  children  
full + we  
want to split

room for  
growth

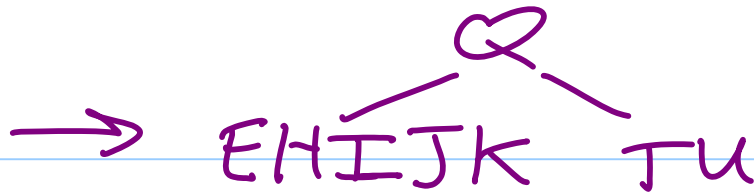
## Top-Down Insertion

Follow path to leaf where you'd insert (with natural extension of binary search tree insertion)

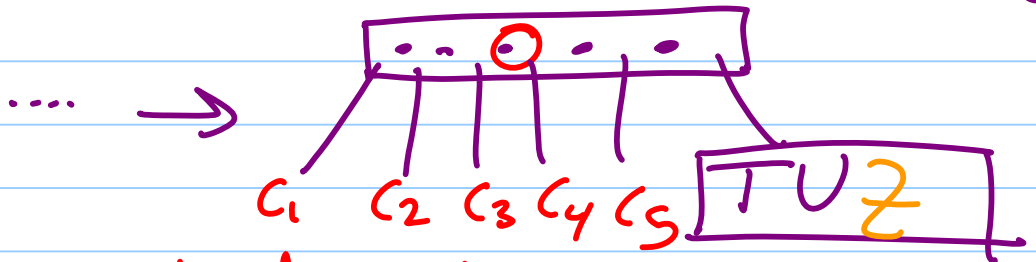
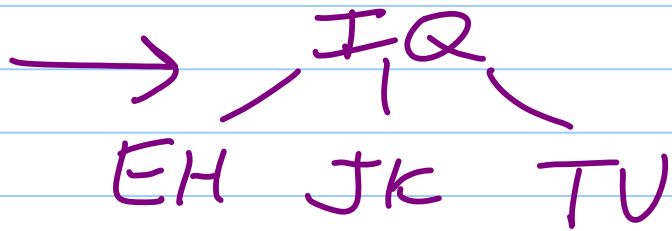
Whenever a full node (2+ children) encountered split it & then continue

Goal: minimize possibility of a page fault occurring twice on same page

**EHQTU**



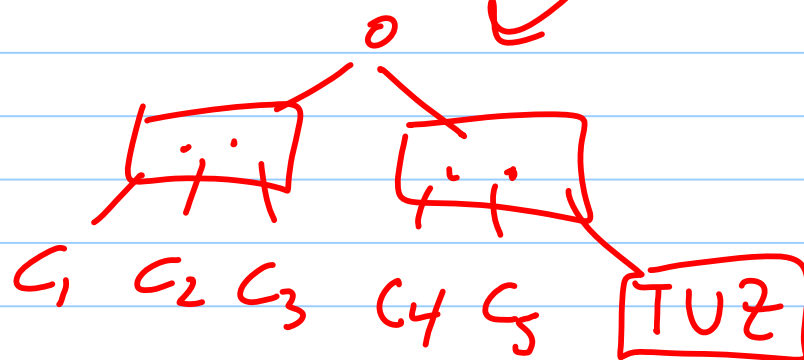
t=3  
5 elements  
is a full  
node



Insert z

top down

bottom-up



## Bottom-Up Insertion

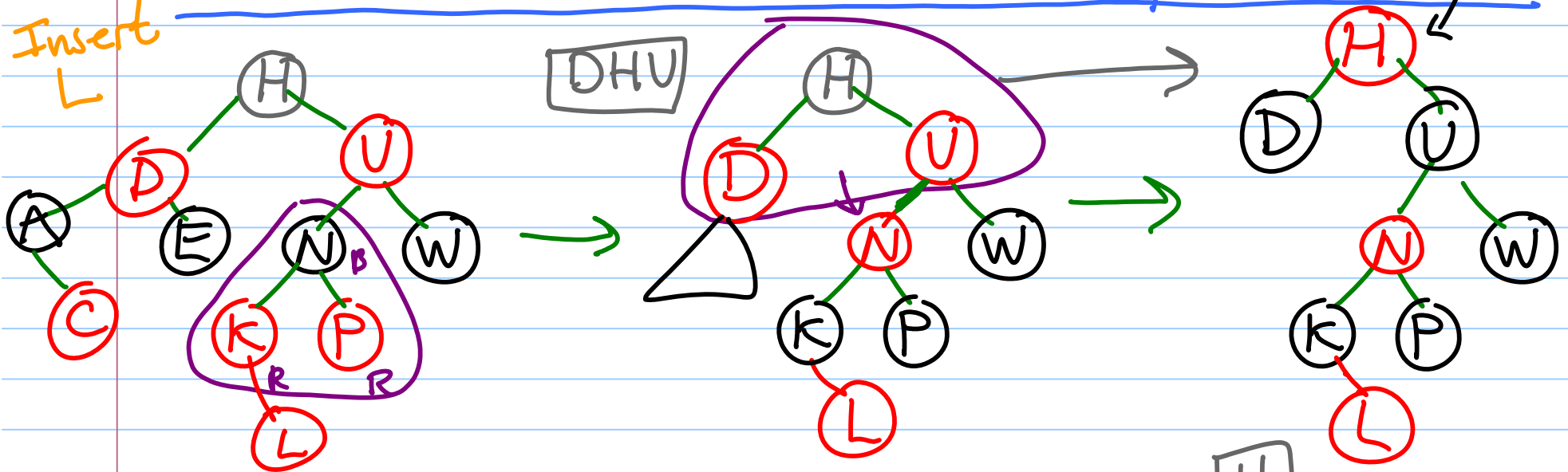
Do standard insertion in leaf if there is room.

Otherwise split leaf (which could propagate to the root). Stop as soon as parent is not full.

Reduces unnecessary splits.

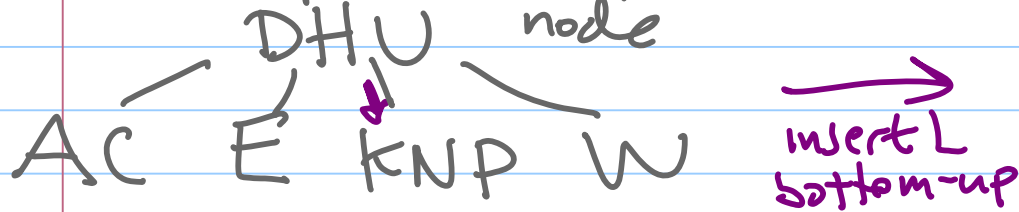
# Relation between red-black tree insertion + 2-3-4 bottom-up insertion

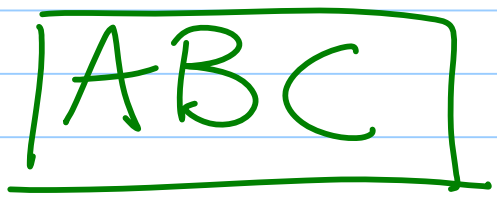
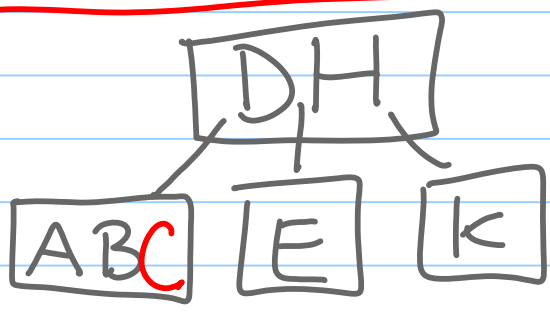
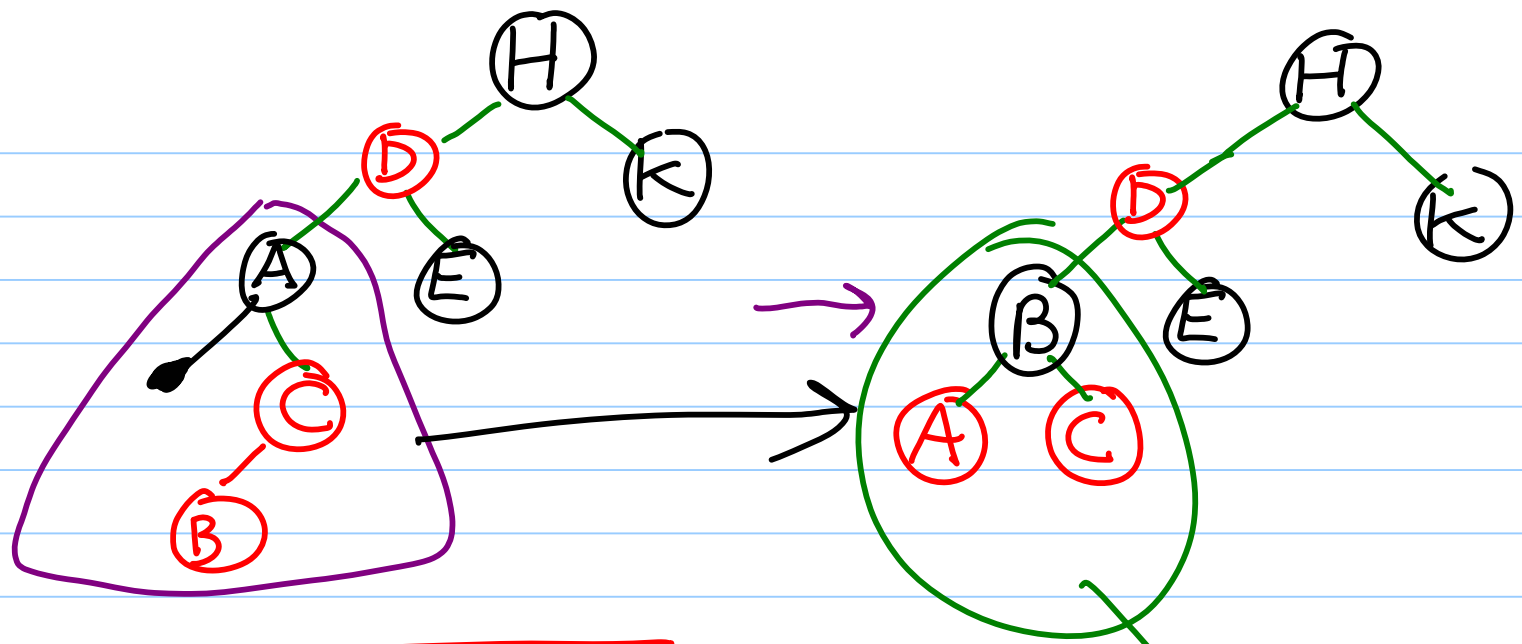
Insert L



last step make H black

Red node is part of its parents



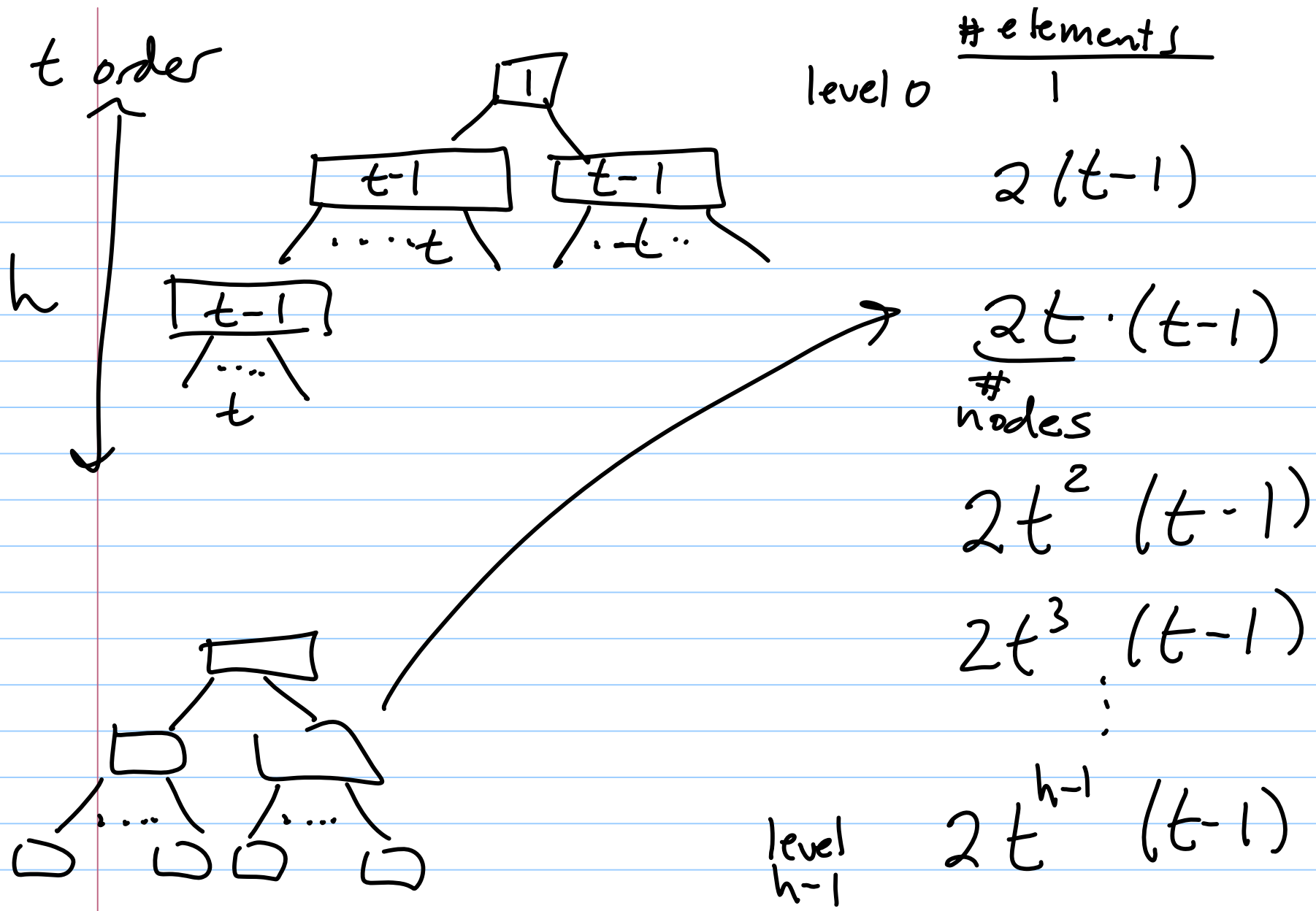


## Analysis of height

suppose the B-tree has height  $h$

What is the min # of elements it might hold?

$$n \geq F(h) \quad \text{solve for } h \quad h \leq f(n)$$



$$n \geq 1 + 2(t-1) + 2t(t-1) + 2t^2(t-1) + \dots + 2t^{h-1}(t-1)$$

$$n \geq 1 + 2(t-1)(1 + t + t^2 + \dots + t^{h-1})$$

$$n \geq 1 + 2 \cancel{(t-1)} \frac{(t^h - 1)}{\cancel{(t-1)}}$$

$$n \geq 1 + 2(t^h - 1)$$

$$n \geq 1 + 2t^h - 2$$

$$n \geq 2t^h - 1$$

$$2t^h \leq n + 1$$

$r \neq 1$

$$\sum_{i=0}^x r^i = \frac{r^{x+1} - 1}{r - 1}$$

$$2t^h \leq n+1$$

$$t^h \leq \frac{n+1}{2}$$

$$h \leq \log_t \left( \frac{n+1}{2} \right)$$

$$\log_t \left( \frac{n+1}{2} \right)$$

$$= \log_t (n+1) - \log_t 2$$

$$\approx \log_t n$$

Maximum height  
of B-tree with  
n elements

## Cost for insertion

$$O\left(\underbrace{\log_t n}_{\text{\# nodes on path down}} \cdot \underbrace{\log_2(2t-1)}_{\text{time per node (sorted array)}}\right) = O(\log_2 n)$$

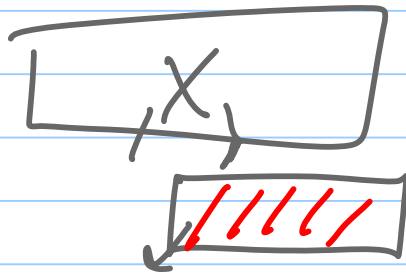
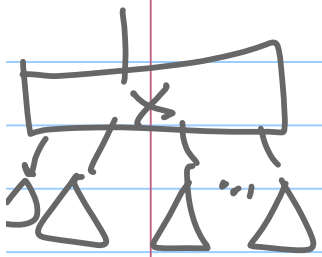
# nodes on path down

time per node  
(sorted array)

Max # of page faults

# Overview of B-tree Deletion

Like binary search tree, for removing element in an internal node, then replace  $x$  by its successor  $\rightarrow$  remove the successor



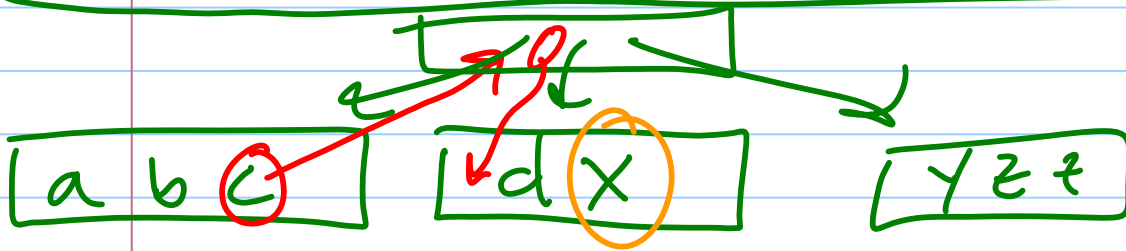
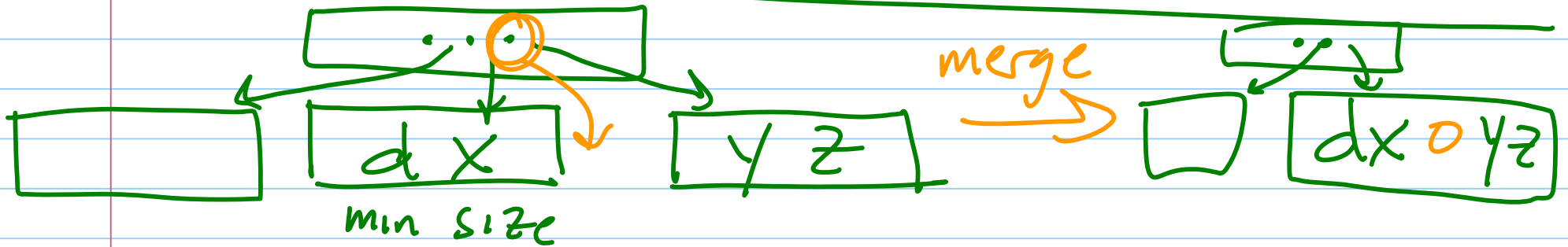
from marked node take leftmost child until reach leaf  $\rightarrow$  succ. leftmost element

hold this in memory until successor is found to replace it

Focus on removing an element  
in a leaf

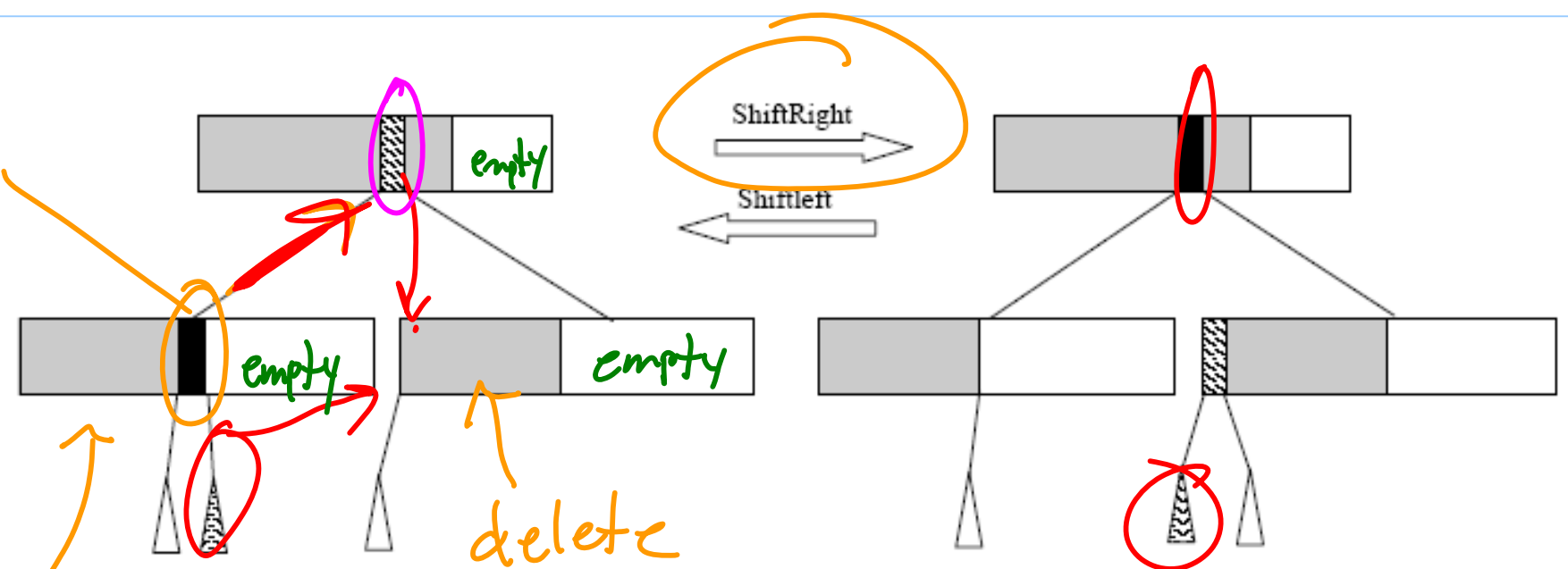
$t-3$   
↑  
2-5  
elements

leaf  $[cdmx]$  ← not minimum size





max  
in  
siblings



not  
minimum-sized

delete  
something  
here  
(min sized)

## Basic Flow for B-tree deletion

Top-down

On search to leaf (for element to delete or its successor)

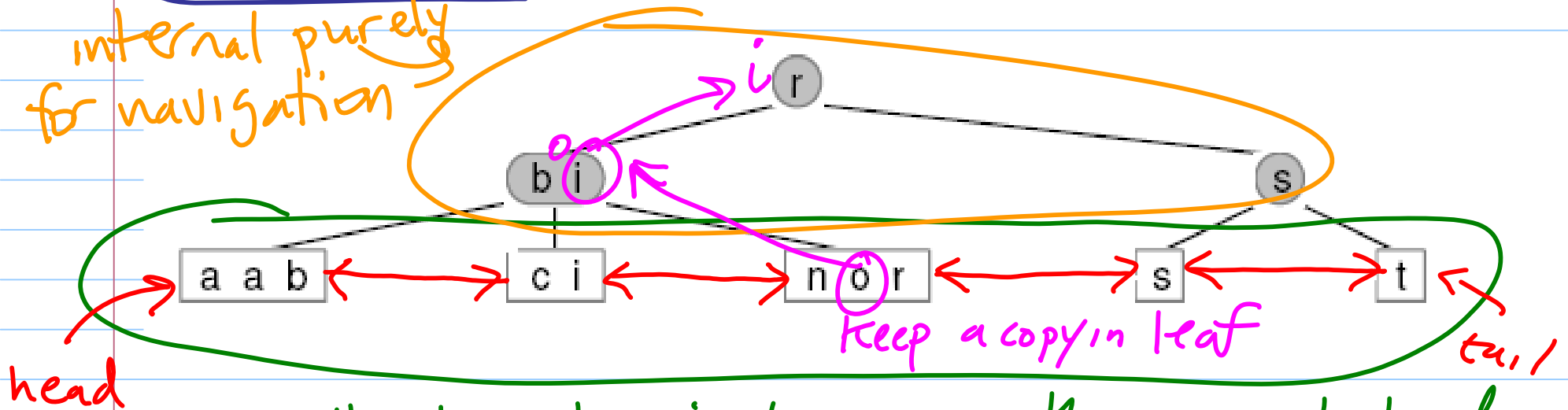
If a minimum-sized node is encountered "fix that" by  
① merging or ② shift left/right

Ensures that the leaf holding element to remove is not min sized

# B+ tree

$t=2$  For letters  
"abstraction"

internal purely  
for navigation



all elements in leaves & they are linked together in a doubly-linked chain

insert p

