

Lecture 23: B Trees II

2023/10/12

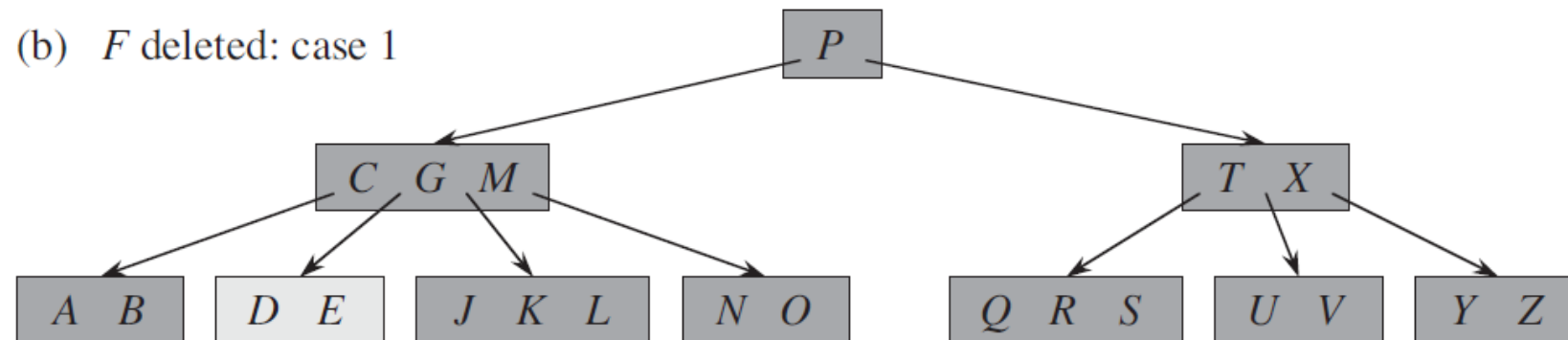
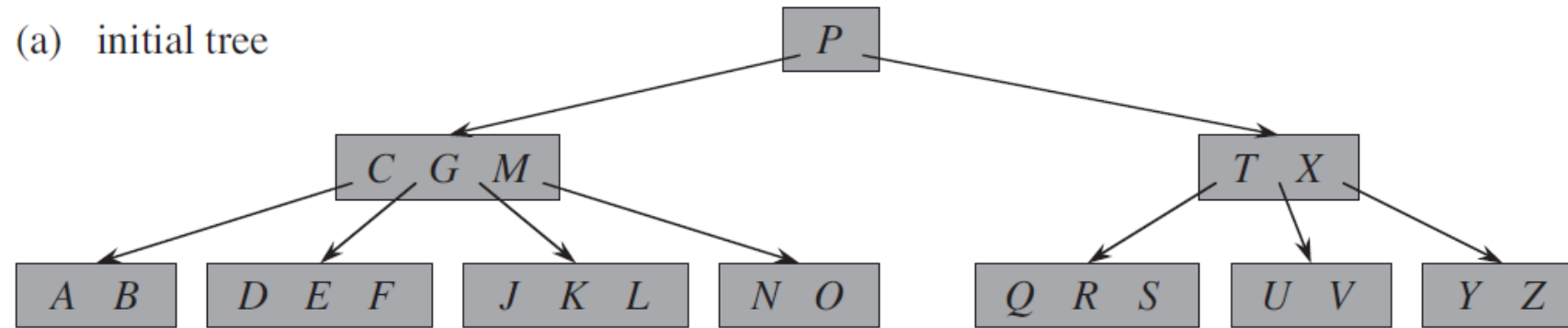
詹博华 (中国科学院软件研究所)

Deletion on B Trees

- More complicated: when deleting from an internal node, need to rearrange its children.
- Also need to guard against a node having too few keys.
- Aim for a single-pass algorithm.
- When called recursively on a node x , make sure x has at least t keys (rather than the minimum $t - 1$).

Deletion: examples ($t = 3$)

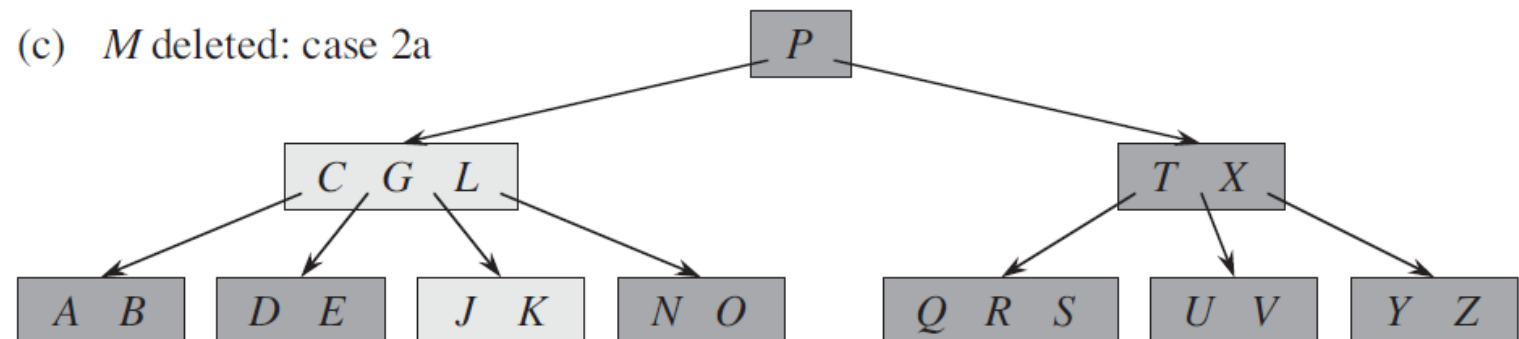
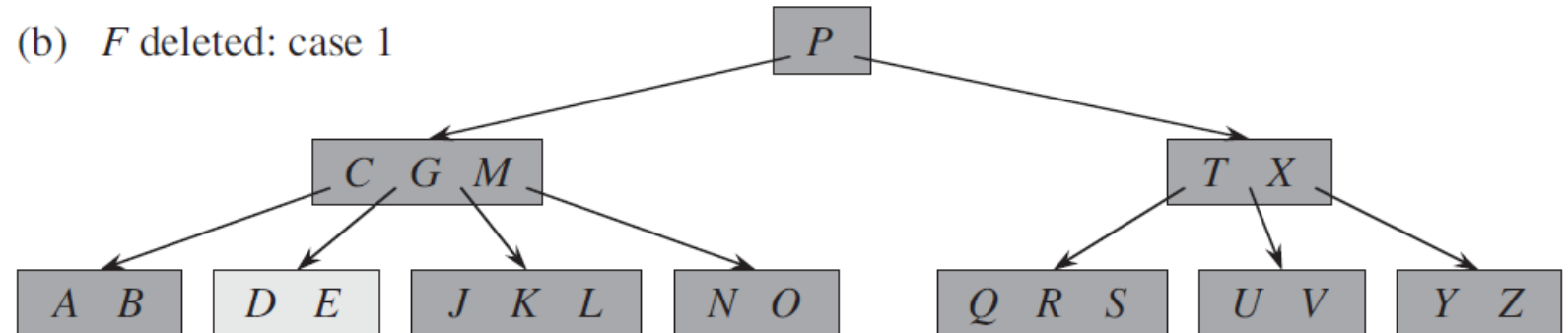
- (Case 1) When k is in a leaf node x , simply delete k from x .



Deletion: examples ($t = 3$)

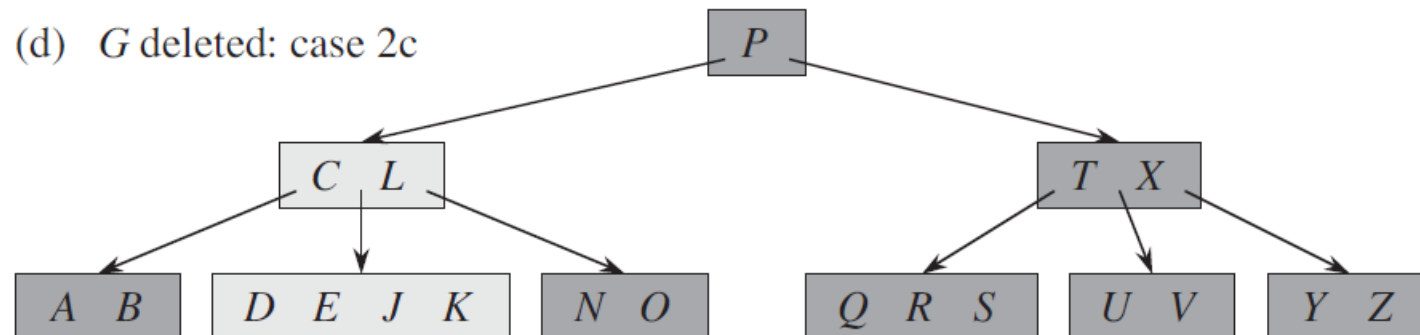
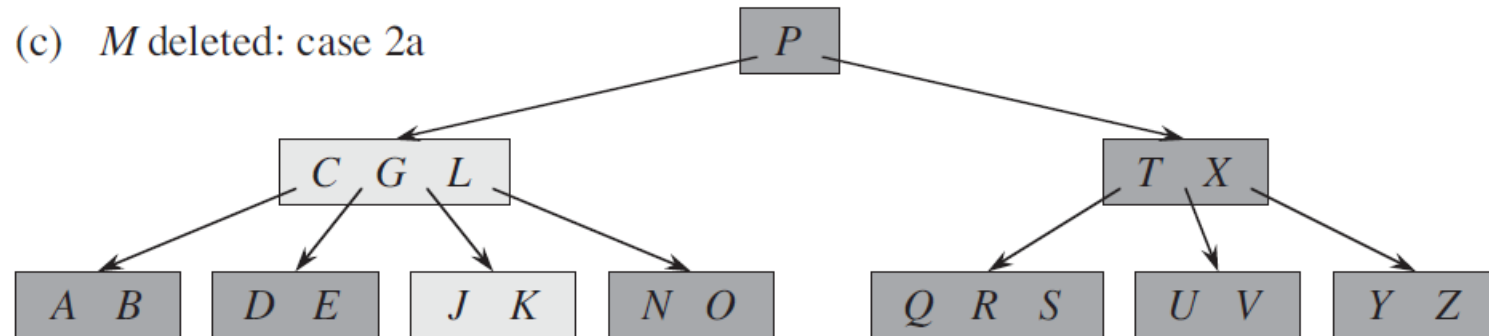
- (Case 2a) When k is in an internal node, and the child node preceding it has at least t keys, delete its predecessor k' and replace k by k' .

(Case 2b)
Similarly for
the case of
succeeding
child node.



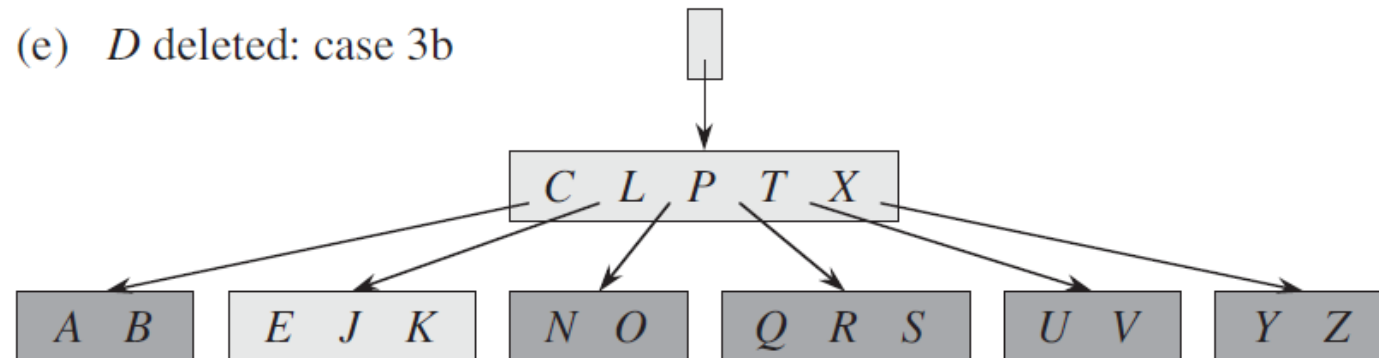
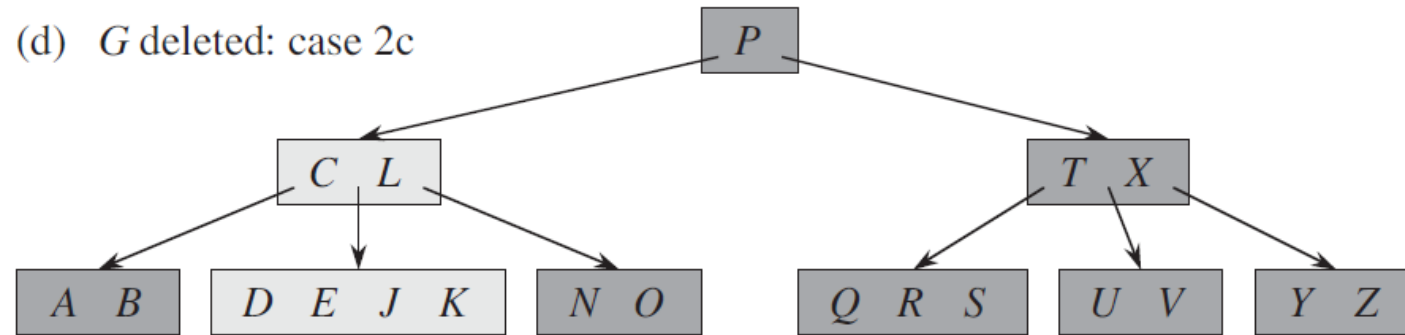
Deletion: examples

- (Case 2c) When k is in an internal node, and the child nodes preceding/succeeding it has only $t - 1$ keys, merge the two child nodes.



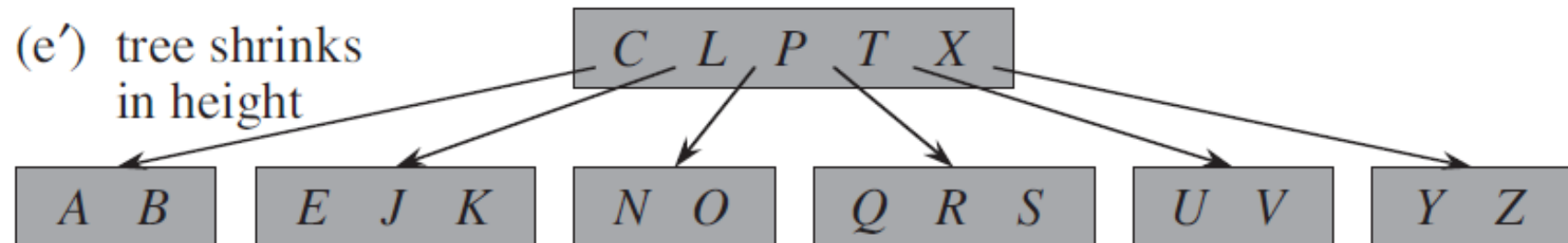
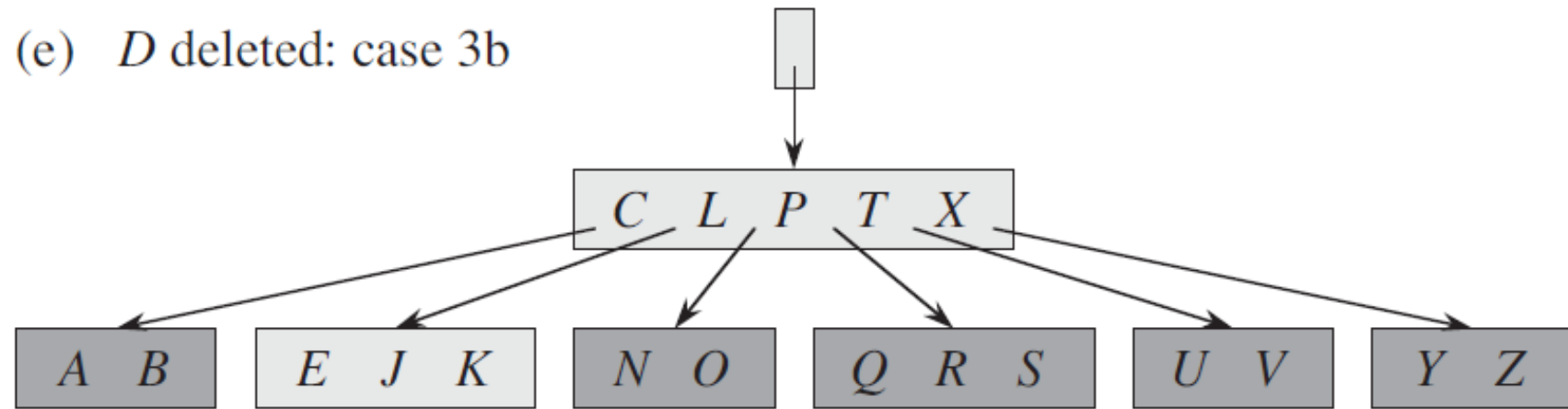
Deletion: examples

- (Case 3b) When k is not in the internal node, and the node that is descended to has only $t - 1$ keys, with neighbors also have $t - 1$ keys, merge node with one of the neighbors.



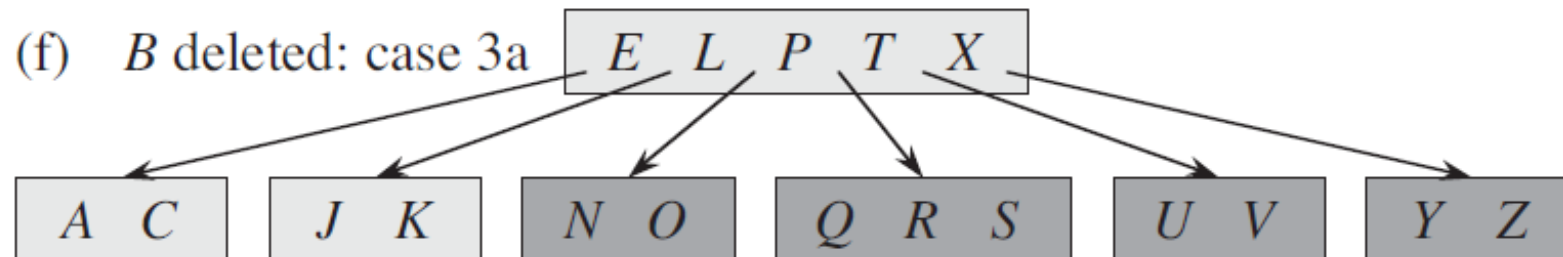
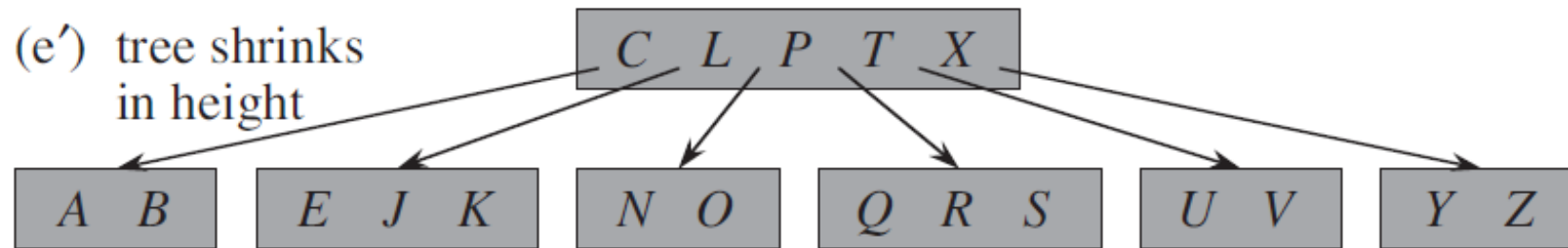
Deletion: examples

- If this causes root to become empty, remove root (shrinks height by 1).



Deletion: examples

- (Case 3a) When k is not in the internal node, and the node that is descended to has only $t - 1$ keys, with a neighbor has at least t keys, move a key from the neighbor and proceed to delete.



Exercise (18.3-1)

- Starting from the last tree on the previous slide, delete C , P and V in order.