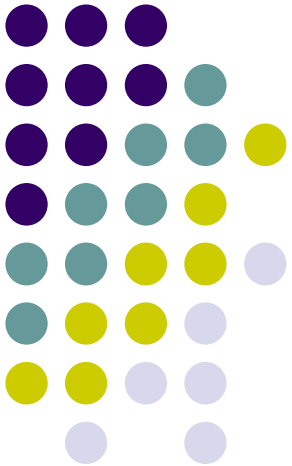
File Structures

Ch09. C. B-Tree Delete



2020. Spring

Instructor: Joonho Kwon

jhkwon@pusan.ac.kr

Data Science Lab @ PNU

Slides are based on the textbook.

datalab

data science laboratory

Outline

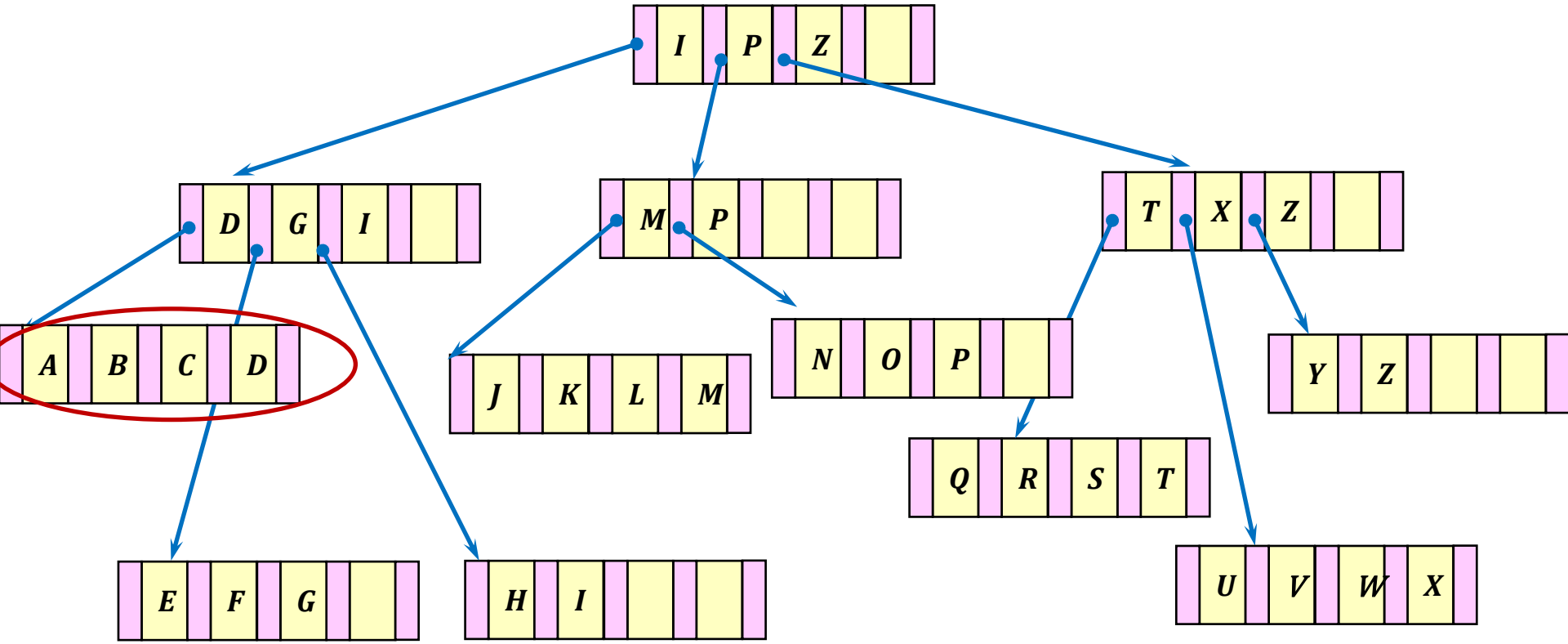
- 9.12 Deletion, Merging, and Redistribution
- 9.13 Redistribution During Insertion
- 9.14 B*-trees
- Skipped
  - 9.15 Buffering of Pages
  - 9.16 Variable-Length Records and Keys

B-tree properties

- Ensure that the B-tree properties
  - 1. Every page, except for the root and the leaves, has at least ceiling of m/2 (=$\lceil m/2 \rceil$) descendants
  - 2. A page contains at least $\lceil m/2 \rceil$ keys and no mor than m key
- During insertions
  - The process of page splitting guarantees that these properties are maintained
- During deletions
  - ??

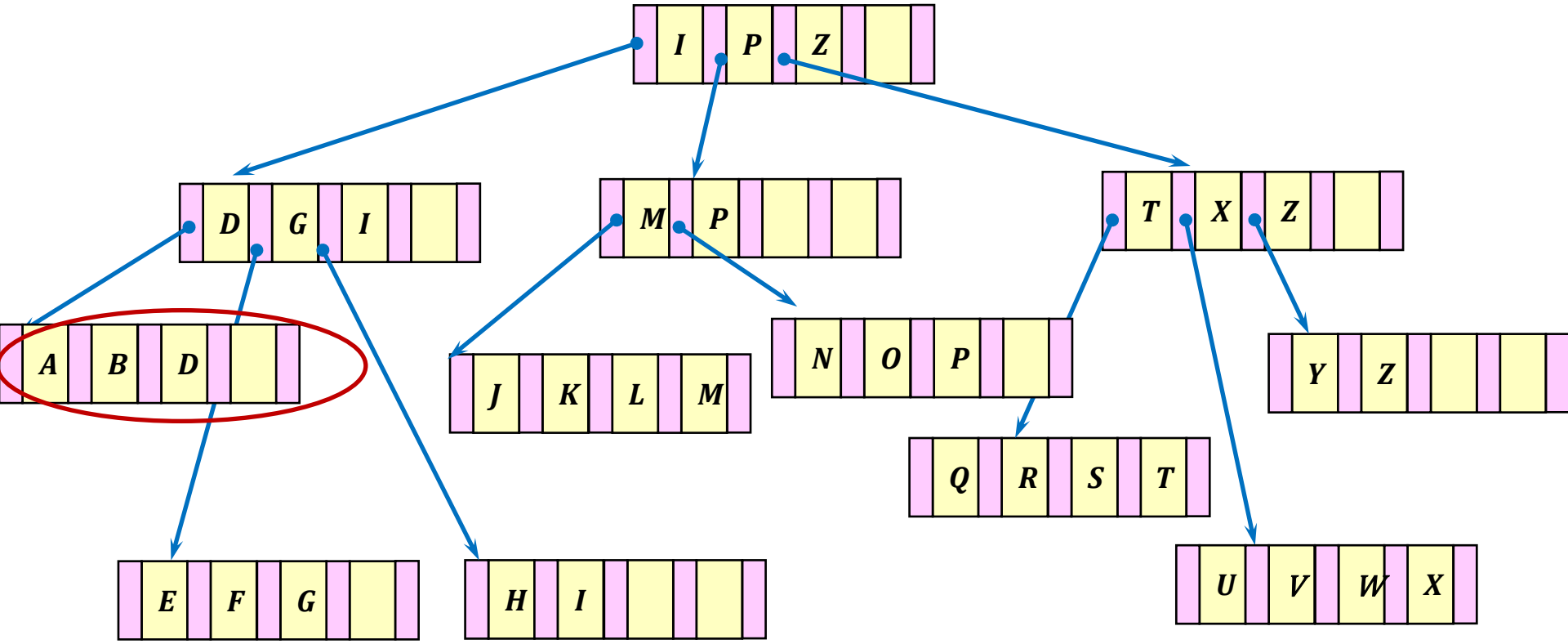
Deletion example: case1 (1/2)

Removal of key C

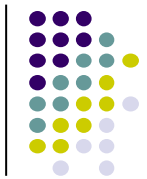


Deletion example: case1 (2/2)

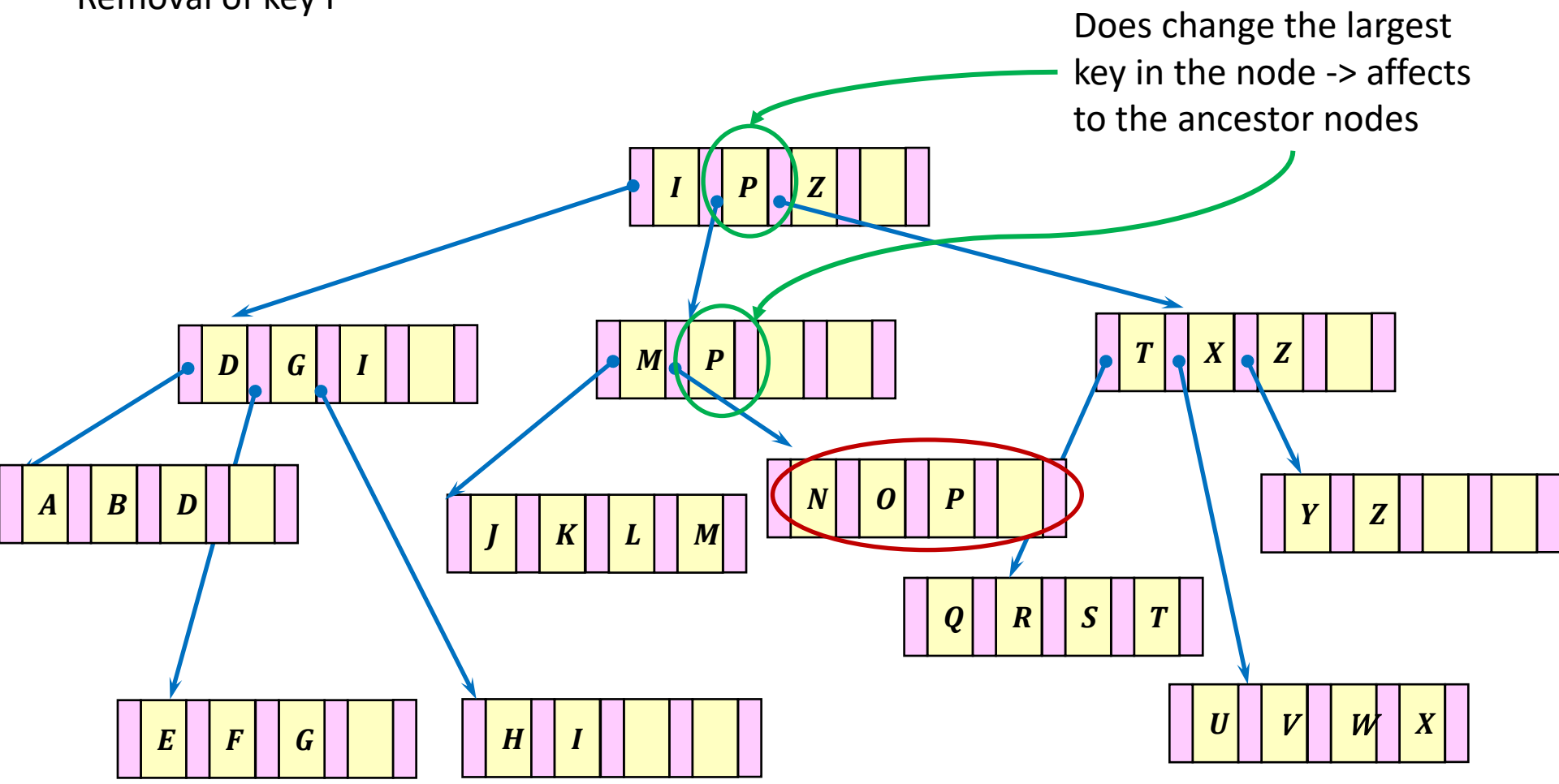
- Change occurs only in leaf node



Deletion example: case2 (1/4)

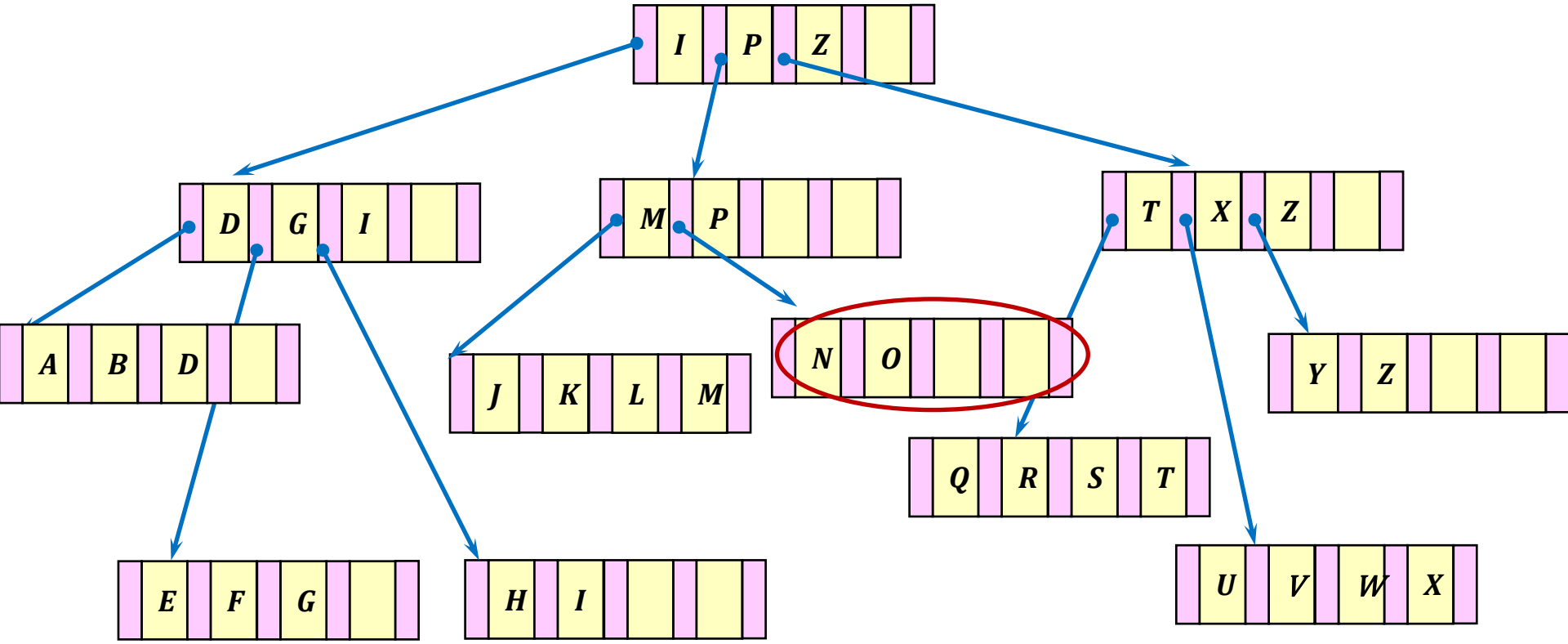


Removal of key P



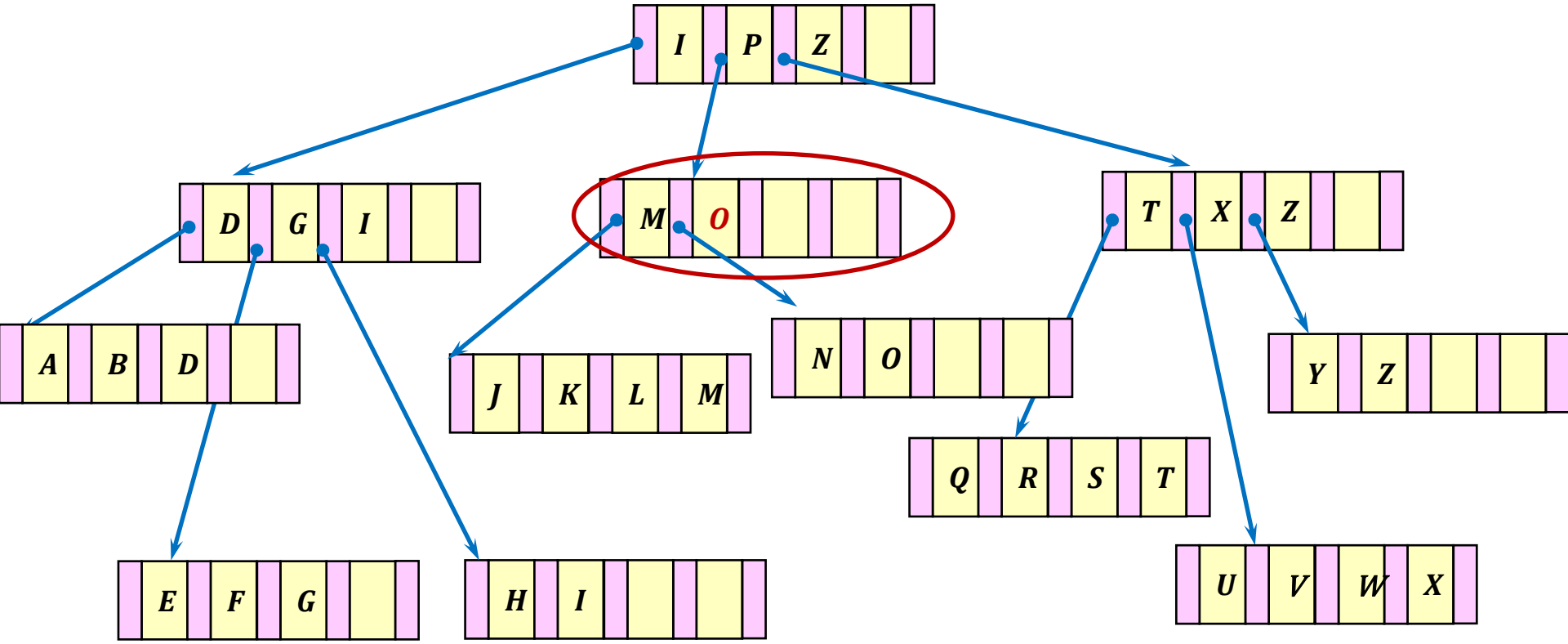
Deletion example: case2 (2/4)

- P from the second leaf node
  - Change the largest key in the node



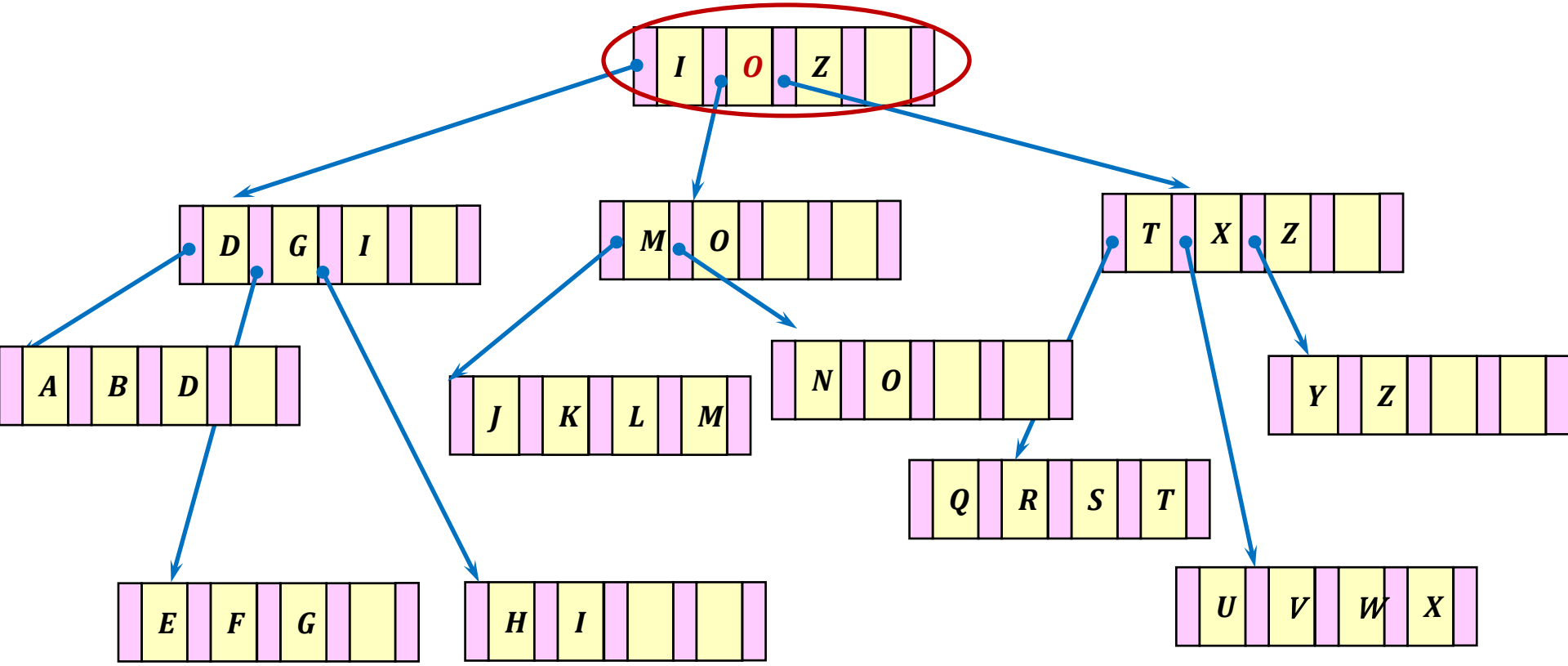
Deletion example: case2 (3/4)

- The second-level node is modified
  - It contains O instead of P

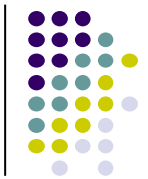


Deletion example: case2 (4/4)

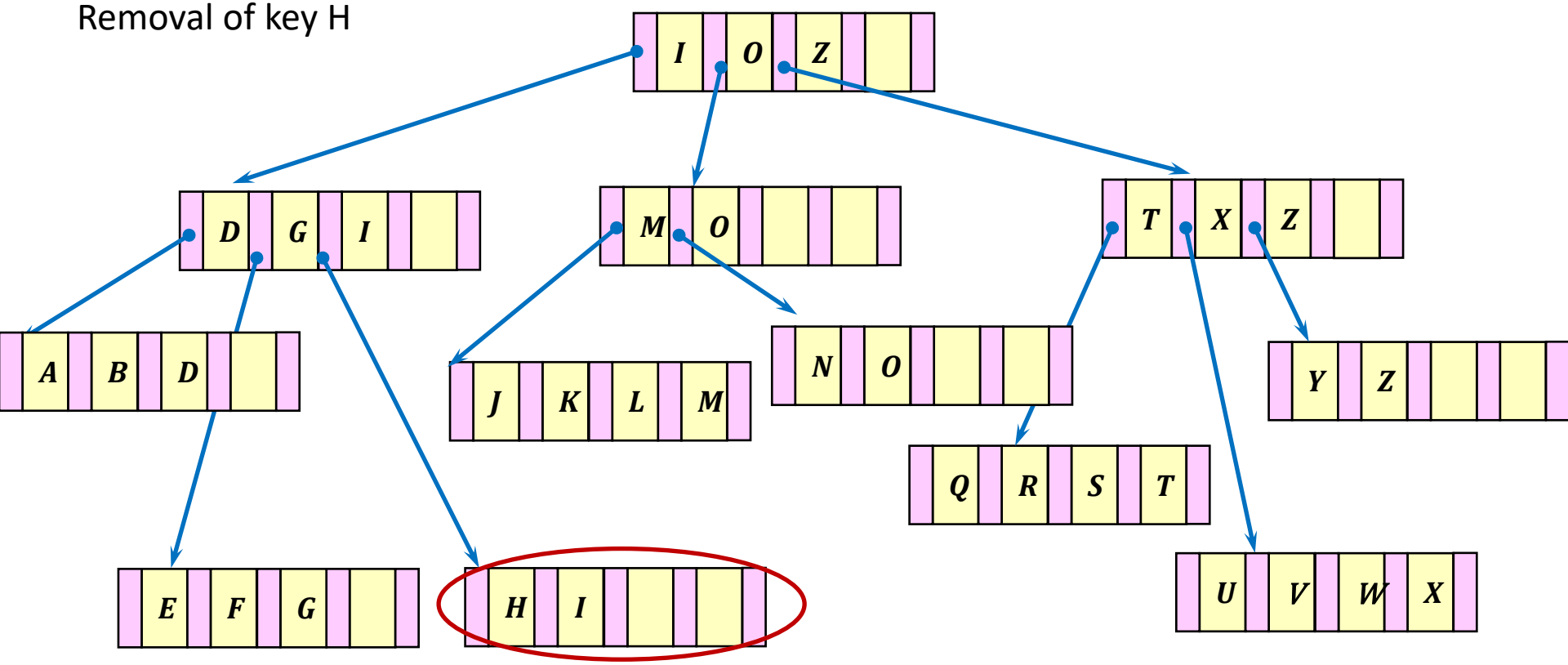
- The root is also modified
  - Key P is replaced by Key O



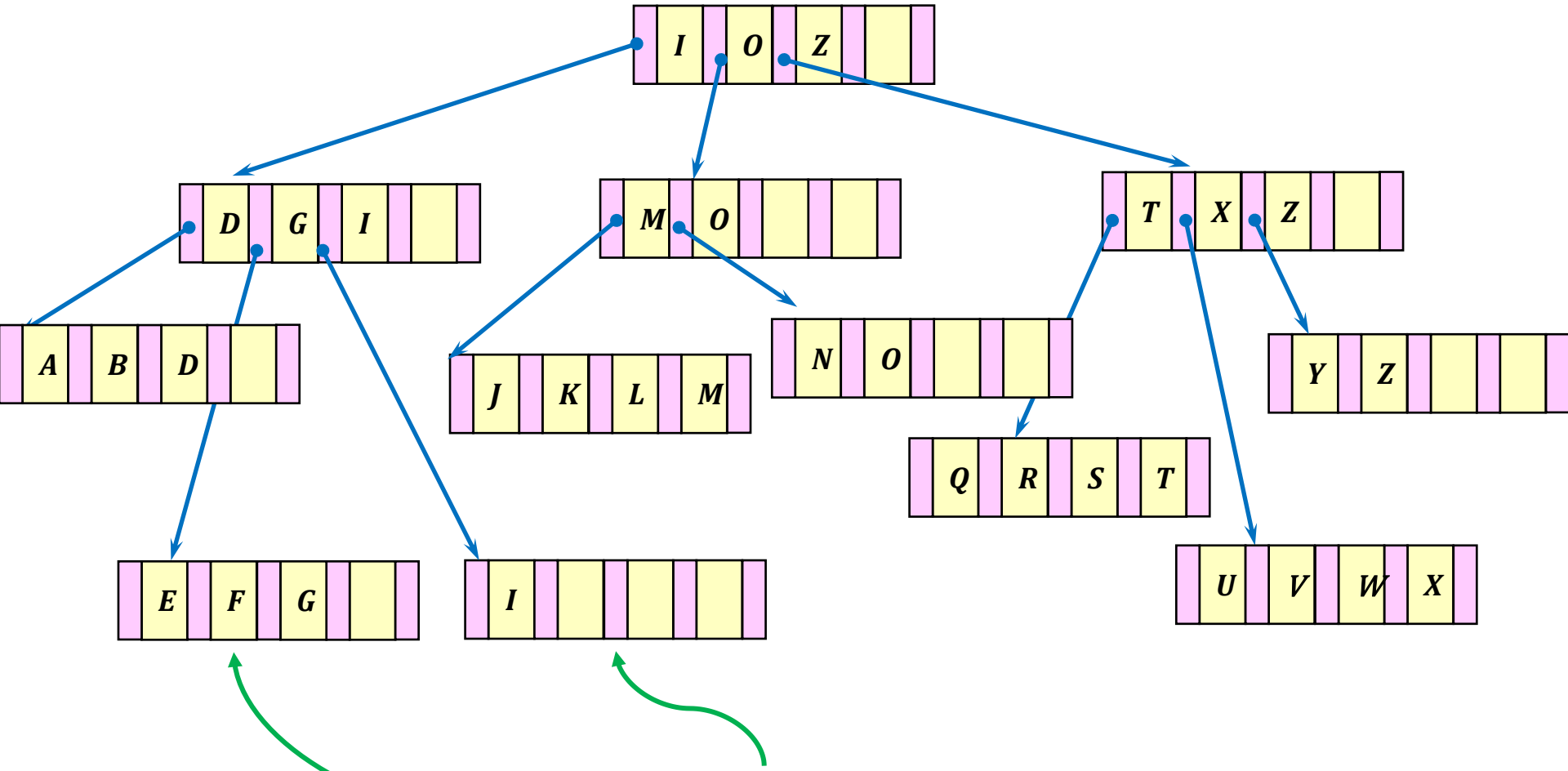
Deletion example: case3 (1/3)



Removal of key H

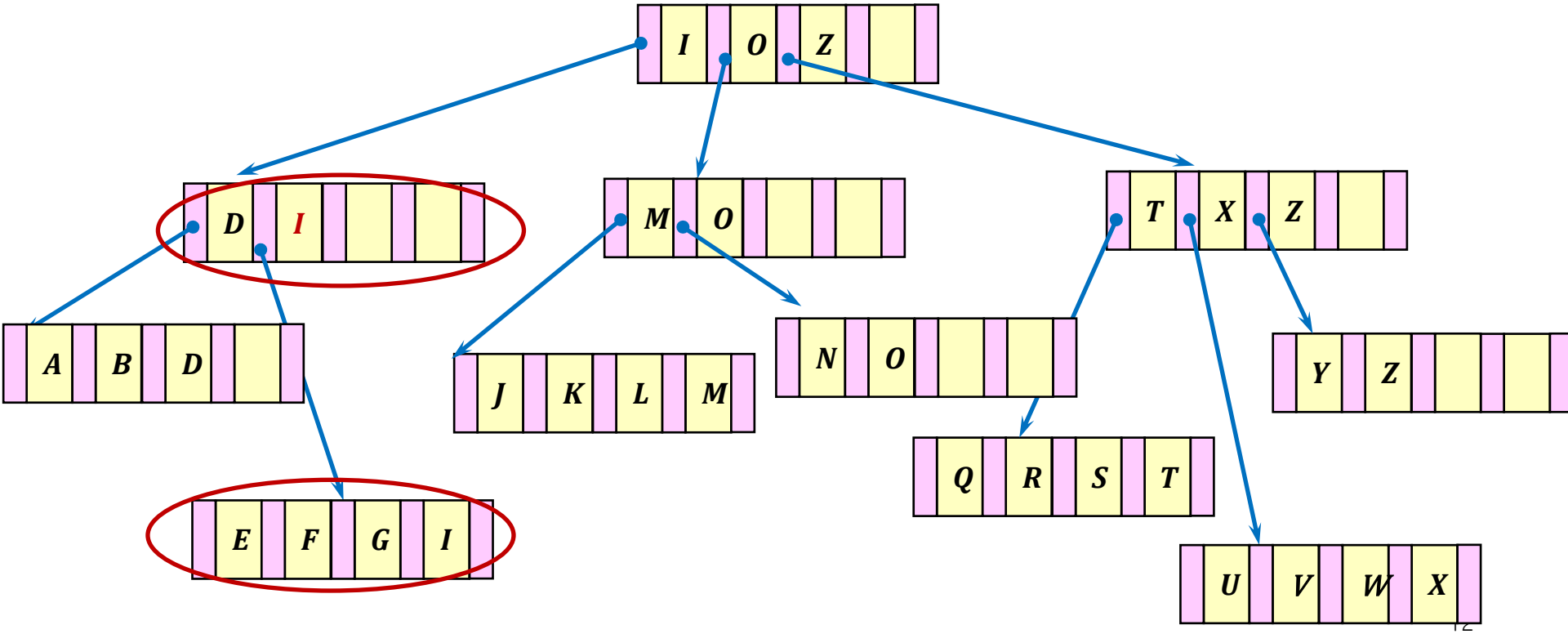


Deletion example: case3 (2/3)



Deletion example: case3 (3/3)

- After the merger
  - Second level node is modified to reflect the current status of the leaf nodes



Deletion, Merging, and Redistribution (1/2)

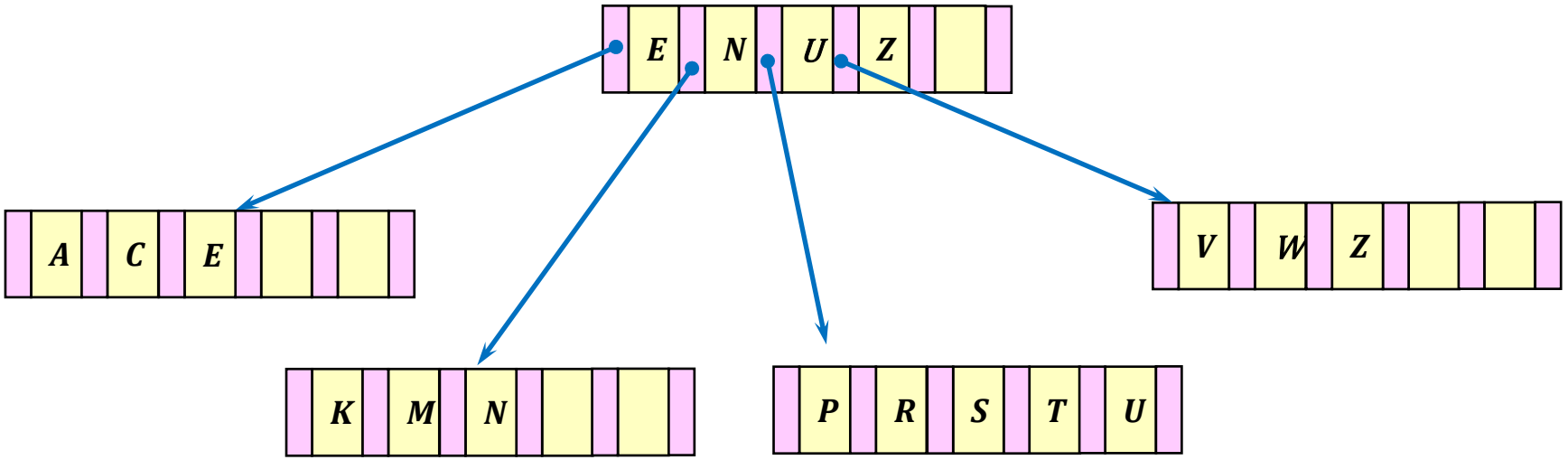
- Rules for deleting a key k from a node n (order m)
  - 1. # of keys in n > m/2, k: not the largest in n
    - simply delete k from n
  - 2. # of keys in n > m/2, k: the largest in n
    - delete k
    - modify the higher level indexes to reflect the new largest key in n
  - 3. # of keys in n = m/2, one of the siblings of n has few enough keys
    - merge n with its sibling
    - delete a key from the parent node

Deletion, Merging, and Redistribution (2/2)

- Rules for deleting a key k from a node n (order m)
  - 4. # of keys in n = m/2 and one of the siblings of n has extra keys
    - redistribute by moving some keys from a sibling to n, and
    - **modify the higher level indexes** to reflect the new largest keys in the affected nodes

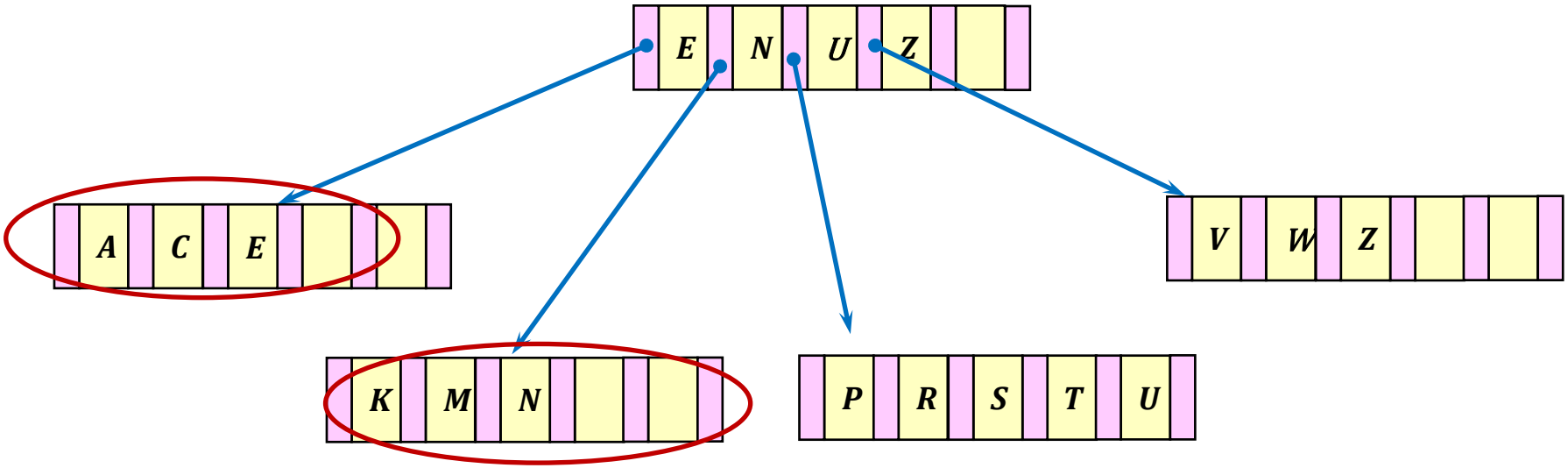
Another Example (1/4)

- A B-tree of order 6(m=6)
  - 3 is minimum number of keys for each node ($\lceil 6/2 \rceil$=3)



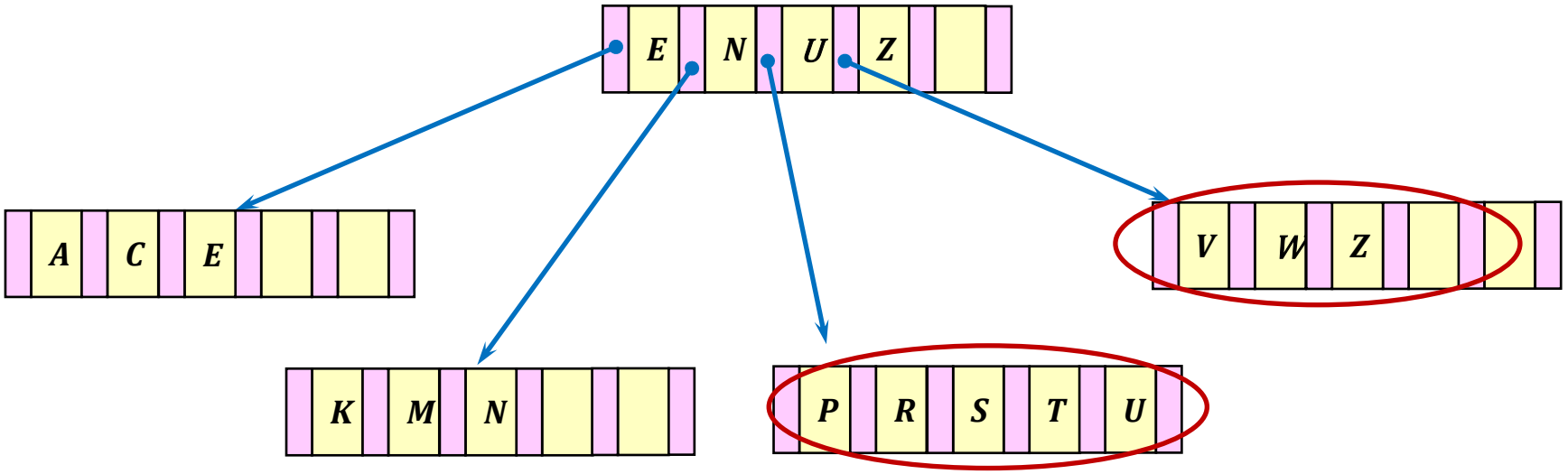
Another Example (2/4)

- Delete C
  - Merging two sibling nodes



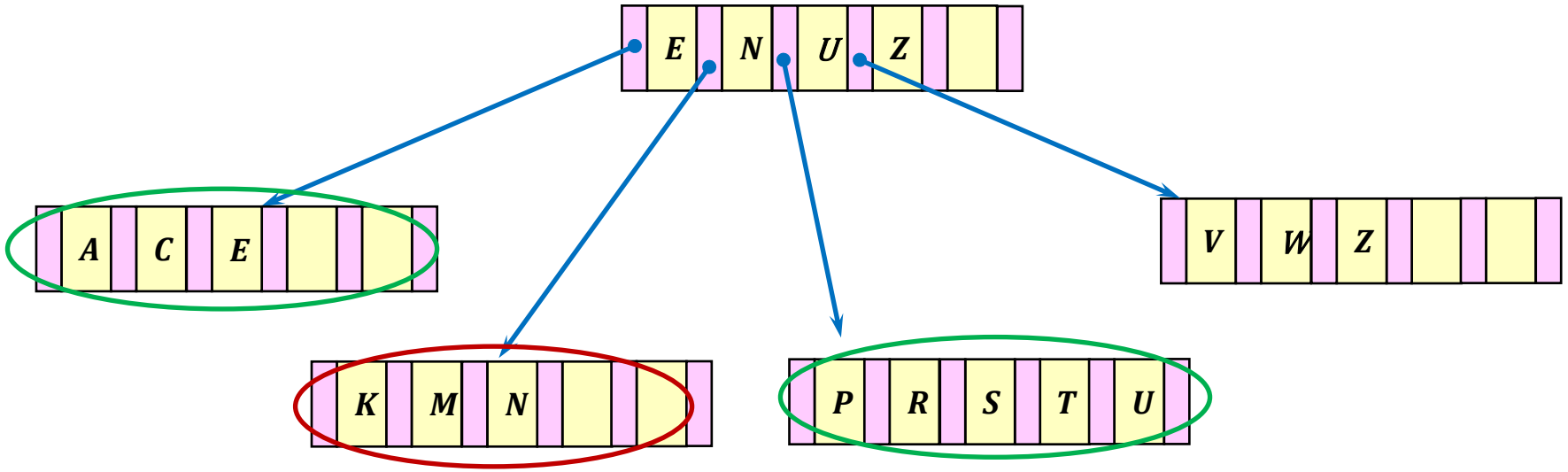
Another Example (3/4)

- Delete W
  - The only sibling has 5 keys, total 7 keys
    - redistribute



Another Example (4/4)

- Delete M
  - Two options
    - Merge with the left sibling
    - Redistribute keys in the right siblining



Deletion algorithm (1/2)

- Algorithm (with redistribution and merging)
  - 1. If the key to be deleted is not in a leaf
    - *swap* it with its immediate successor, which is in a leaf (might be redistributed or concatenated!)
  - 2. *Delete* the key

Deletion algorithm (2/2)

- Algorithm (with redistribution and merging)
  - 3. If underflow occurs
    - (the leaf now contains one too few keys),
    - 3.1 If the left or right sibling has more than the minimum number of keys , redistribute
    - 3.2 Otherwise, concatenate the two leaves and the median key from the parent into one leaf
    - 3.3 Apply above step 3 to the parent as if it were deleted

Redistribution

- Occur when a sibling has more than the minimum # of keys
- Idea: Move keys between siblings
- Result in a change in the key in the parent page
- Does not propagate : strictly local effects
- How many keys should be moved?
  - Not necessarily fixed
  - Even distribution is desired

Merge (Concatenation)

- Occur in case of underflow
- Combining the two pages and the key from the parent page ==> make a single full page
- Reverse the splitting

- Concatenation must involve demotion of keys
  - *may cause underflow in the parent page*

- The effects propagate upward

Redistribution During Insertion

- A way to improve storage utilization
- A way of avoiding the creation of new pages
- Tend to make an efficient B-tree in terms of space utilization
  - Worst case : around 50%
  - Average case : 67 ~ 69%
  - With redistribution during insertion : over 85%

Outline

- 9.12 Deletion, Merging, and Redistribution
- 9.13 Redistribution During Insertion
- 9.14 B*-trees
- Skipped
  - 9.15 Buffering of Pages
  - 9.16 Variable-Length Records and Keys

B* Trees (1/2)

- *Knuth*, 1973, *Addison-Wesley*
- Use redistribution operation during insertion
- Perform *two-to-three* split
  - When split, the page has at least one sibling that is also full
  - After split, the pages are about 2/3 full
  - The page with at least (ceiling of (2m -1)/3) keys

    c.f. remember (ceiling of (m/2)) -1 keys

B* Trees (2/2)

- Properties of a B*-tree of order m
  - extend the notion of redistribution during insertion to include new rules for splitting

  - 1. Every page has a maximum of m descendants
  - 2. Every page except for the root has at least (2m-1)/3 descendents (v.s. m/2)
  - 3. The root has at least two descendants (unless it is a leaf)
  - 4. All the leaves appear on the same level

Q&A