# COMP2010
# Data Structures and Algorithms

## Lecture 13: $B^+$-Trees (Part 2)

Department of Computer Science & Technology

United International College

# B+ Tree Review

- A B+ tree of order M
  - Each internal node has at most M children (M-1 keys)
  - Each internal node, except the root, has between $\lceil M/2 \rceil$-1 and M-1 keys. The root node can have only one key (2 children)
  - Each leaf has between $\lceil L/2 \rceil$ and L data items

# Deletion

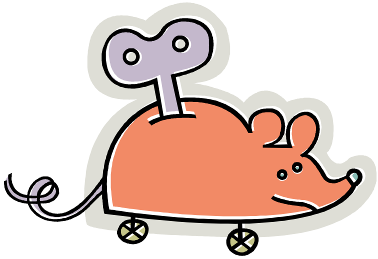
- To delete a key target, we find it at a leaf x, and remove it.
- Two situations to worry about:

  (1) target is a key in some internal node (needs to be replaced, according to our convention)

  (2) After deleting target from leaf x, x contains less than $\lceil L/2 \rceil$ keys (needs to merge nodes)

# Situation 1: Removal of a Key

- target can appear in at most one ancestor y of x as a key (why?)
- Node y is seen when we searched down the tree.
- After deleting from node x, we can access y directly and replace target by the new smallest key in x



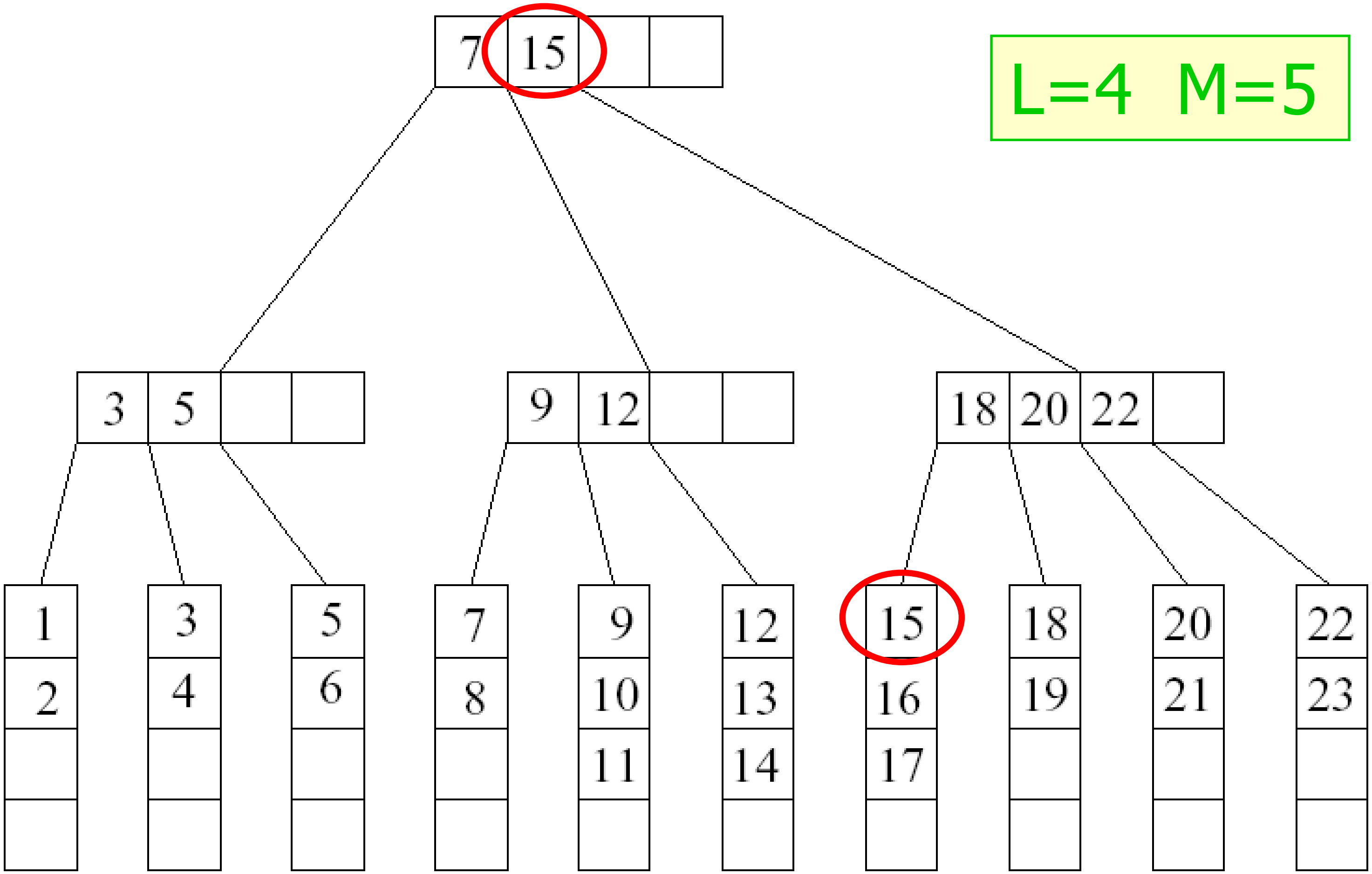
# Situation 2: Handling Leaves with Too Few Keys

- Suppose we delete the record with key target from a leaf.
- Let u be the leaf that has $\lceil L/2 \rceil - 1$ keys (too few)
- Let v be a sibling of u with at least $\lceil L/2 \rceil + 1$ keys
- Let k be the key in the parent of u and v that separates the pointers to u and v
- There are two cases

# Handling Leaves with Too Few Keys

- Case 1: v contains $\lceil L/2 \rceil+1$ or more keys and v is the right sibling of u
  - Move the leftmost (**smallest**) record from v to u
- Case 2: v contains $\lceil L/2 \rceil+1$ or more keys and v is the left sibling of u
  - Move the rightmost (**largest**) record from v to u
- Then set the key in parent of u that separates u and v to be the new smallest key in u

# Deletion Example

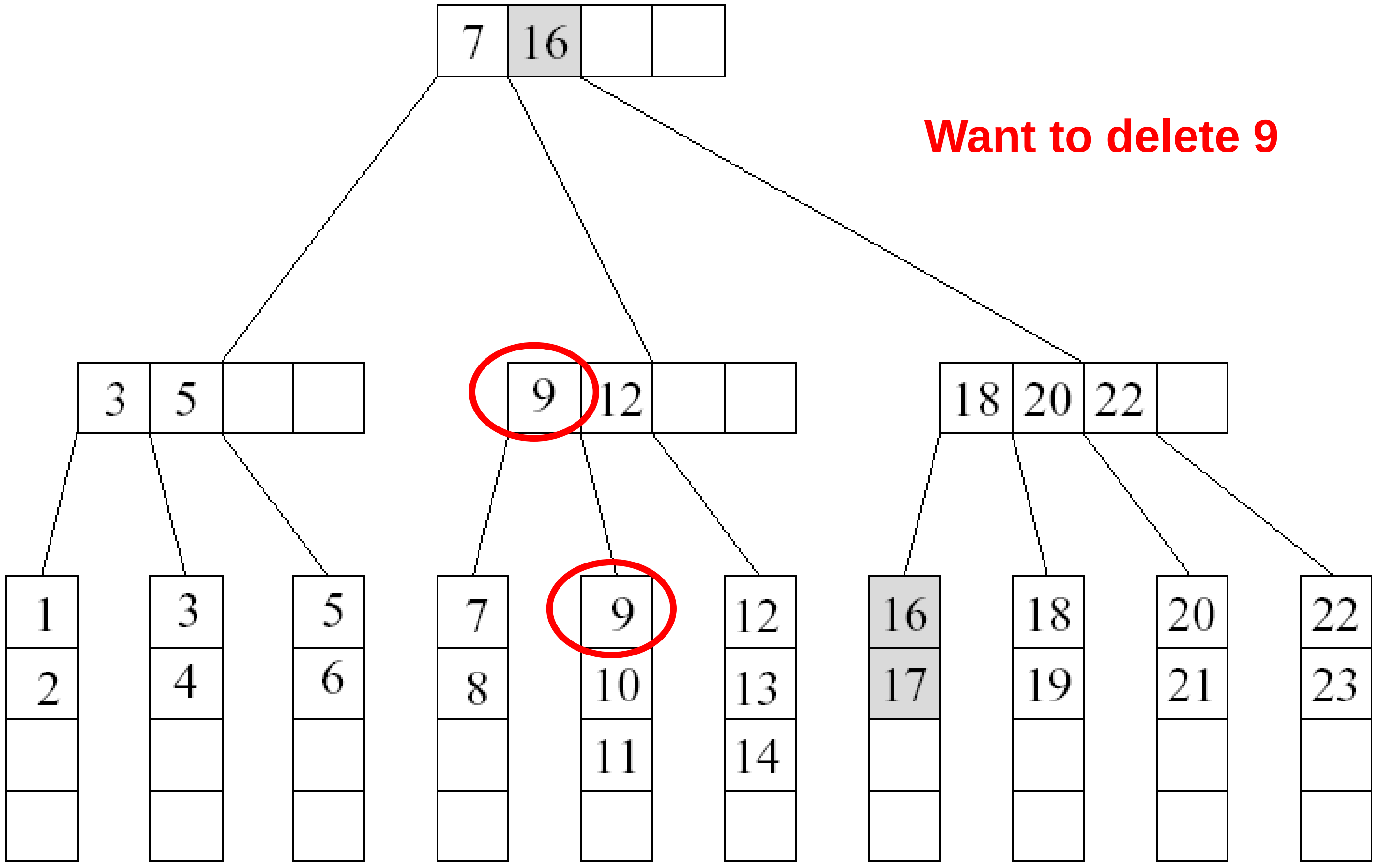


L=4 M=5

Initial tree, M = 5

**Want to delete 15**

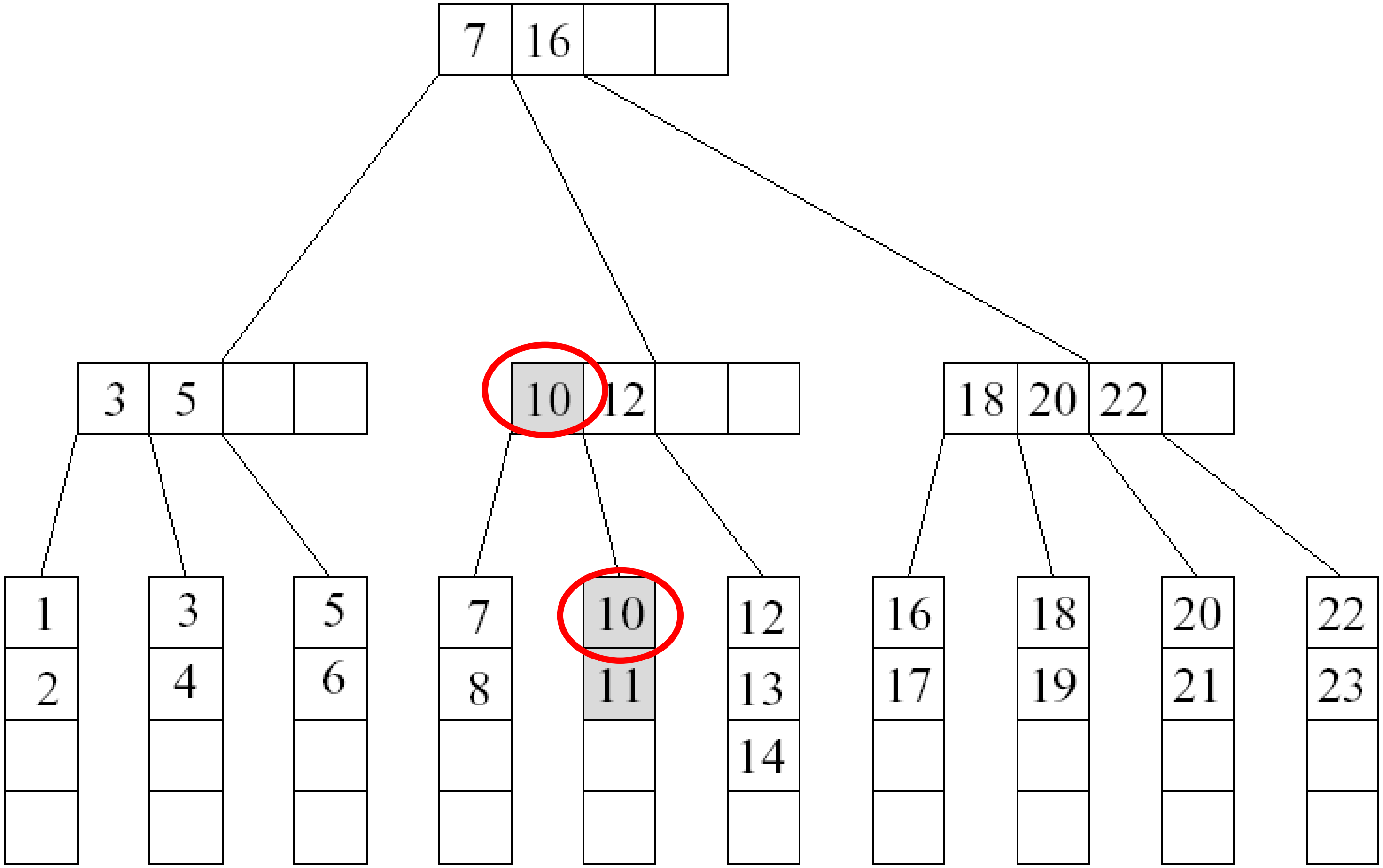
# Deletion Example (Cont’d)



Want to delete 9

15 deleted, shaded entries have been changed

# Deletion Example (Cont’d)

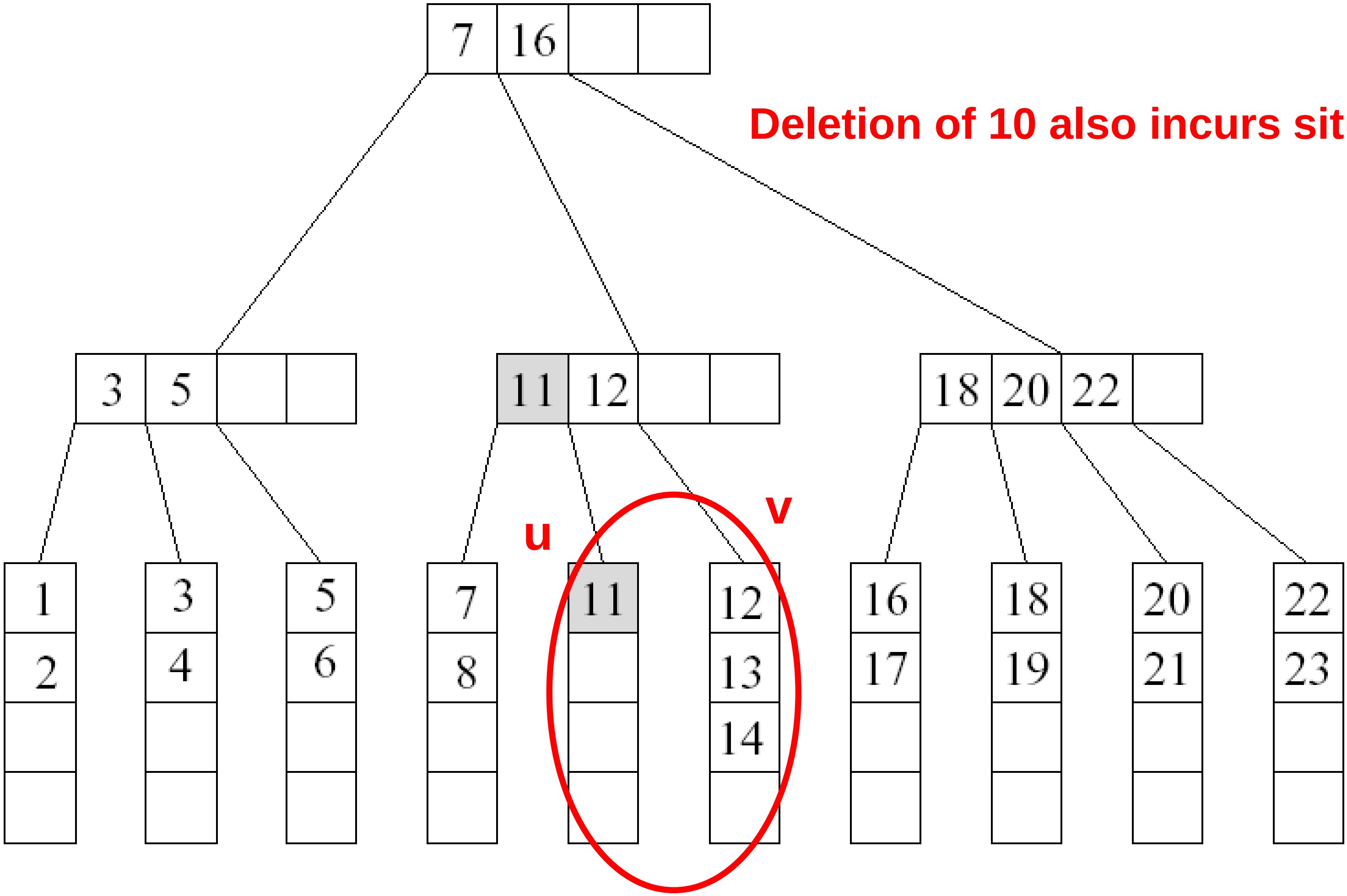


Want to delete 10, situation 1

9 deleted

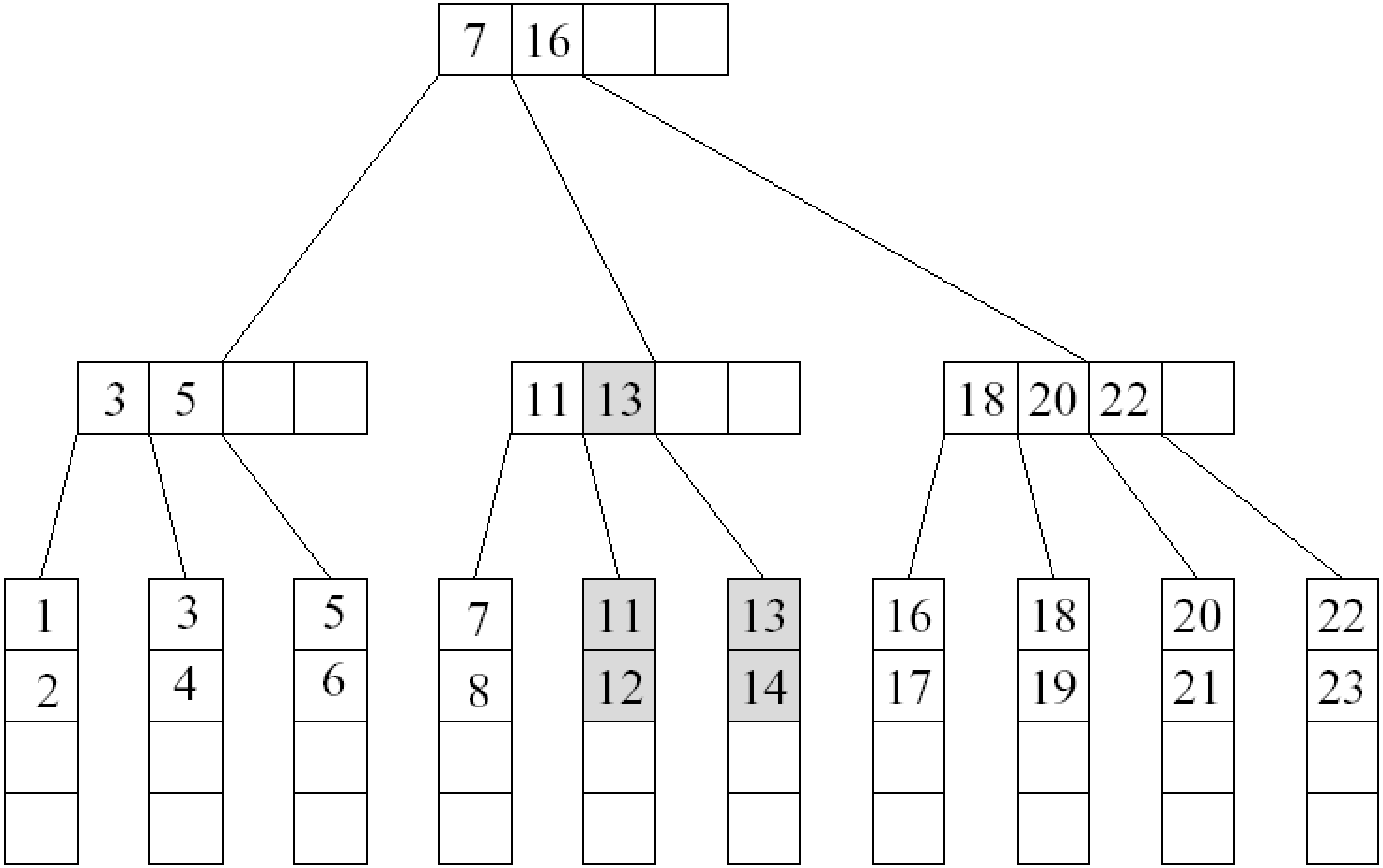
# Deletion Example (Cont'd)

Deletion of 10 also incurs situation 2



10 deleted, step 1

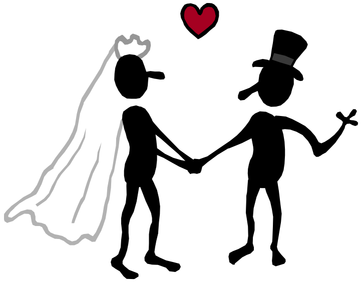
# Deletion Example (Cont'd)



10 deleted, final step: borrow from right sibling

# Merging Two Leaves

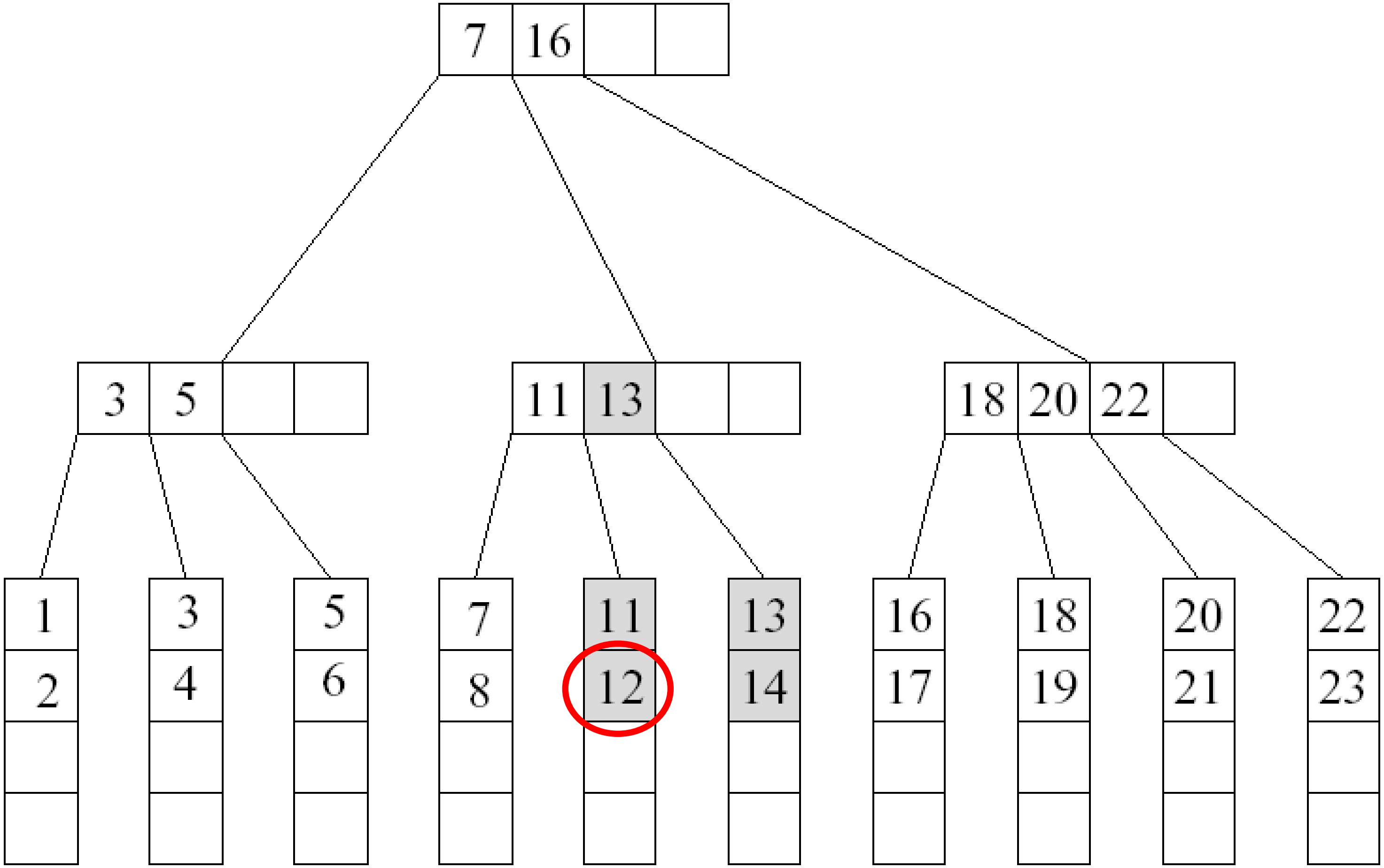
- If no sibling leaf with $\lceil L/2 \rceil+1$ or more keys exists, then merge two leaves.
- Case 1: Suppose that the right sibling v of u contains exactly $\lceil L/2 \rceil$ keys. Merge u and v
  - Move the keys in u to v
  - Remove the pointer to u at parent
  - Delete the separating key between u and v from the parent of u



# Merging Two Leaves (Cont’d)

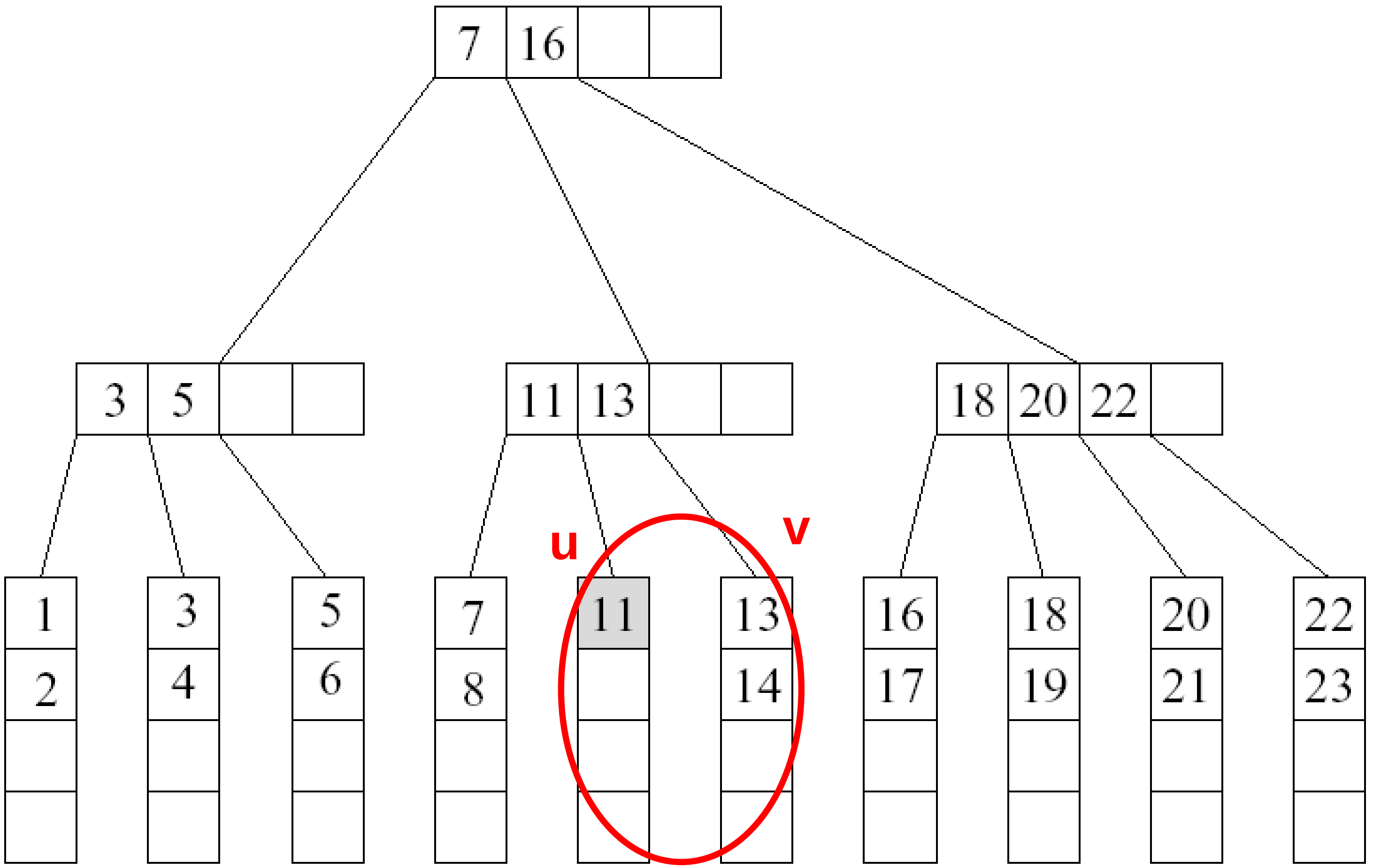
- Case 2: Suppose that the left sibling v of u contains exactly $\lceil L/2 \rceil$ keys. Merge u and v
  - Move the keys in u to v
  - Remove the pointer to u at parent
  - Delete the separating key between u and v from the parent of u

# Example (M=5, L=4)



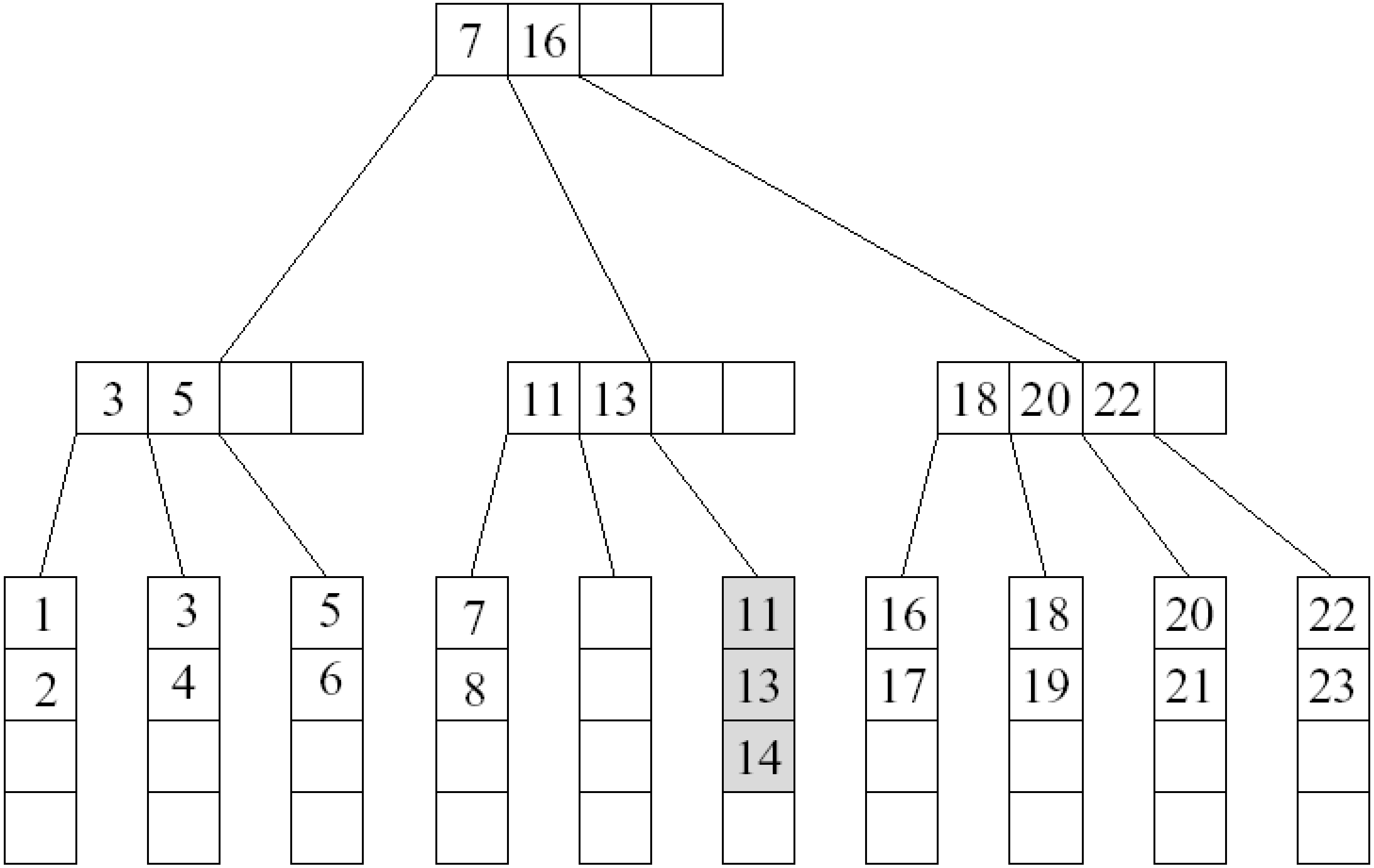
Want to delete 12

# Example (Cont'd)



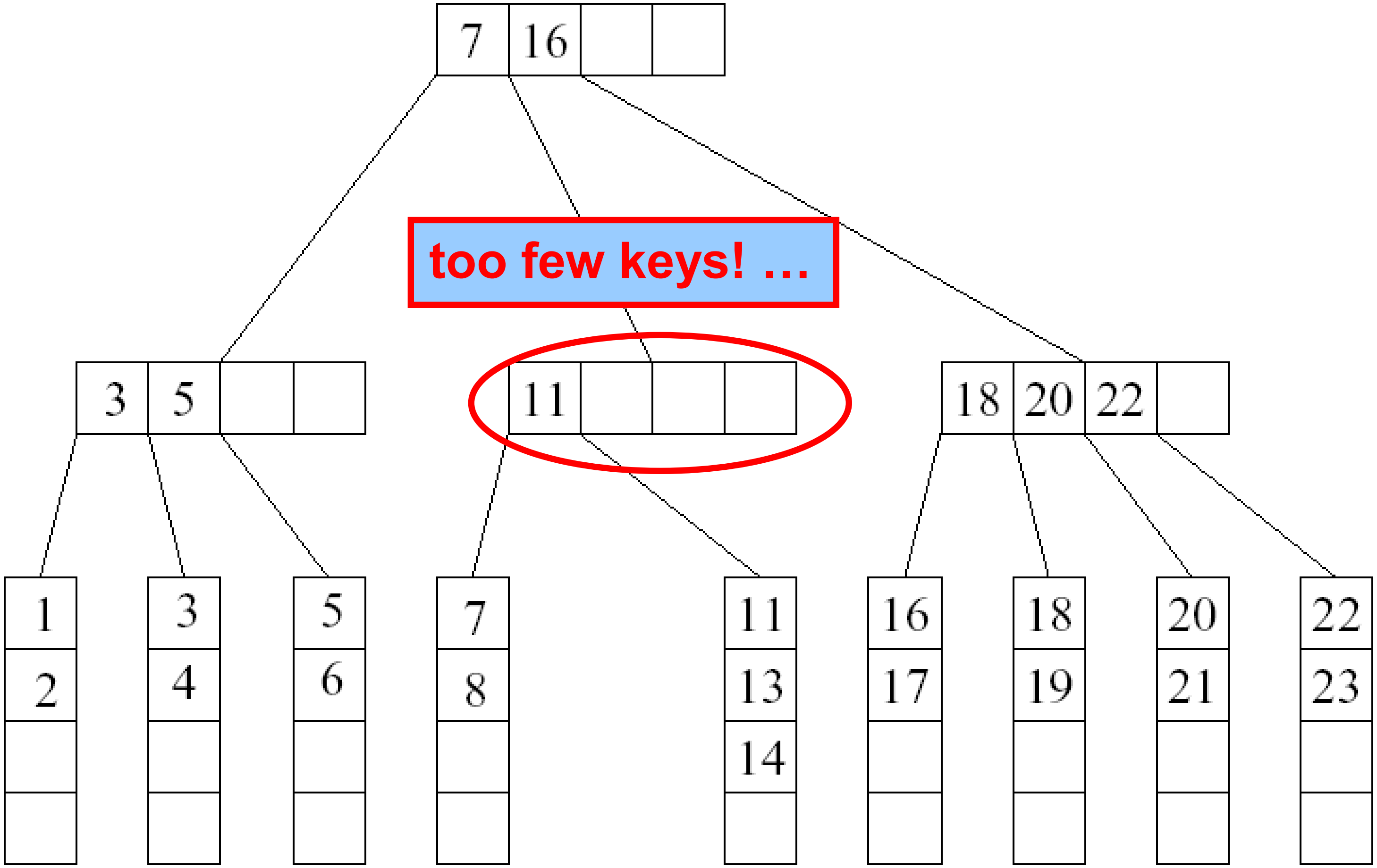
12 deleted, step 1

# Example (Cont'd)



12 deleted, merge with right sibling

# Example (Cont’d)



too few keys! …

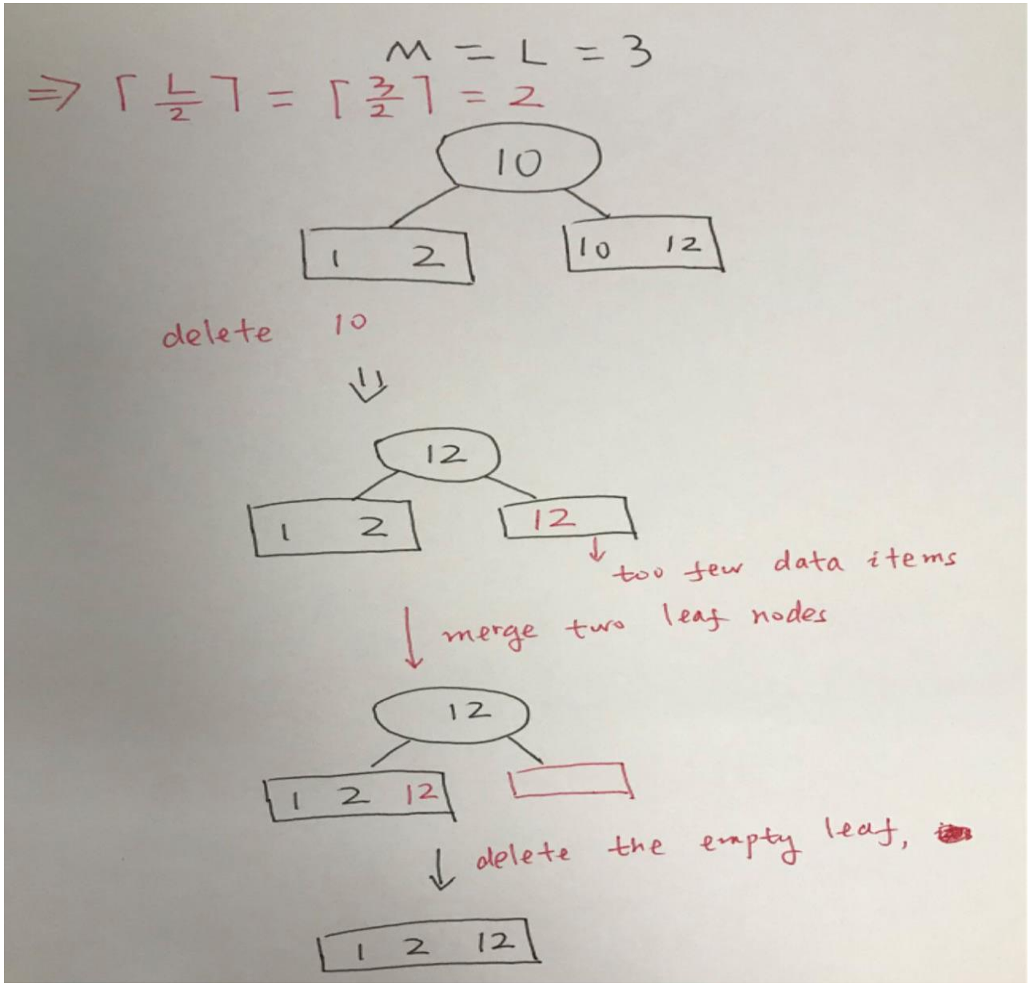
12 deleted, delete the empty leaf and the separating key 13 in parent

# Deleting a Key in an Internal Node

- Suppose we remove a key from an internal node u, and u has less than $\lceil M/2 \rceil -1$ keys after that
- Case 1: u is a root
  - Thus u has only one child, then we remove u and make its child the new root

# Example for case 1

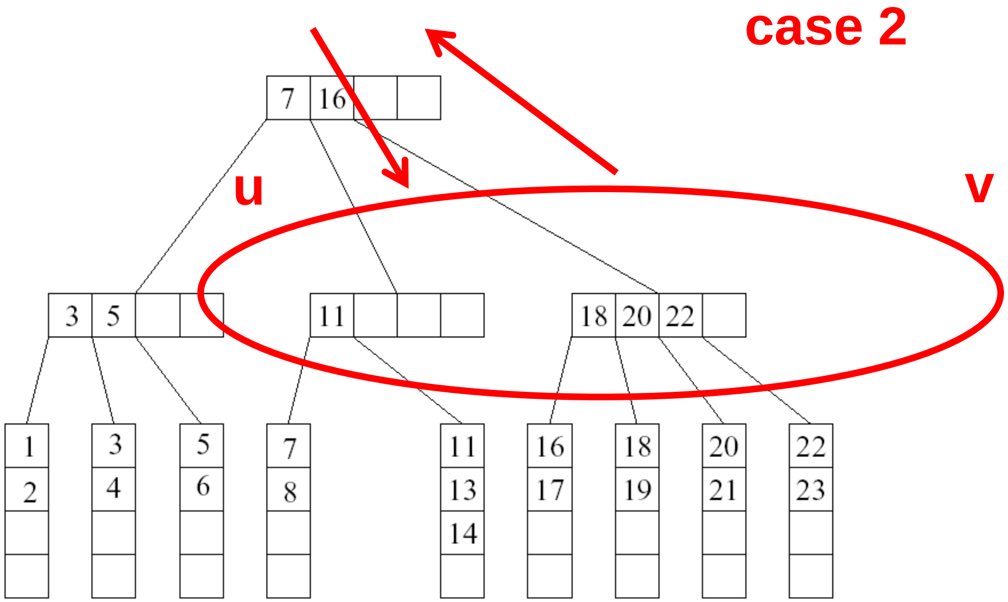
- In this example, M = L =3, and the root node only has 1 key.



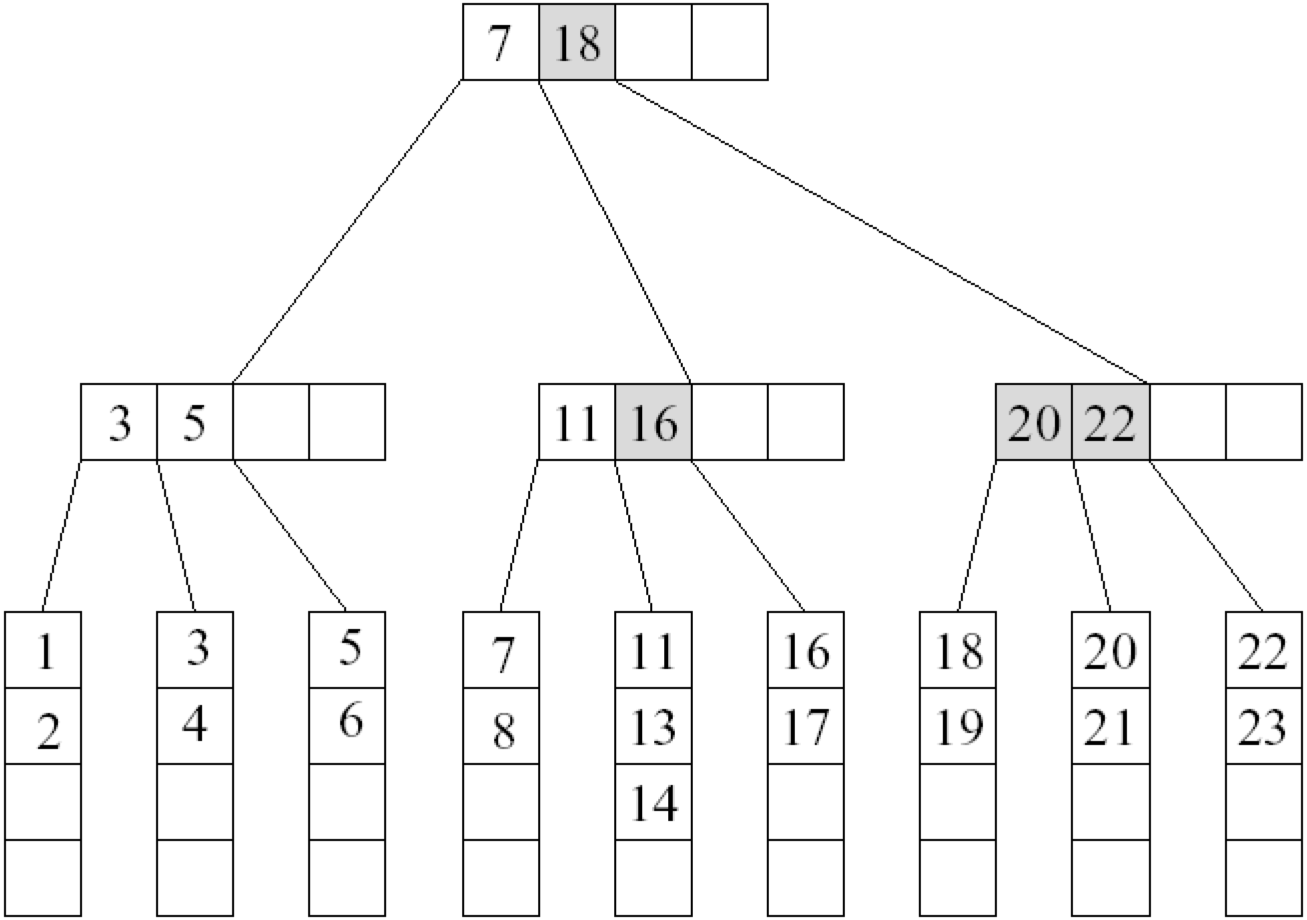
# Deleting a key in an internal node

- Case 2A: the right sibling v of u has ⌈M/2⌉ keys or more
  - Move the separating key between u and v in the parent of u and v down to u
  - Move the leftmost key in v to become the separating key between u and v in the parent of u and v.
  - Make the leftmost child of v the rightmost child of u
- Case 2B: the left sibling v of u has ⌈M/2⌉ keys or more
  - Move the separating key between u and v in the parent of u and v down to u.
  - Move the rightmost key in v to become the separating key between u and v in the parent of u and v.
  - Make the rightmost child of v the leftmost child of u

# ...Continue From Previous Example (M=5, L=4)



# Cont'd

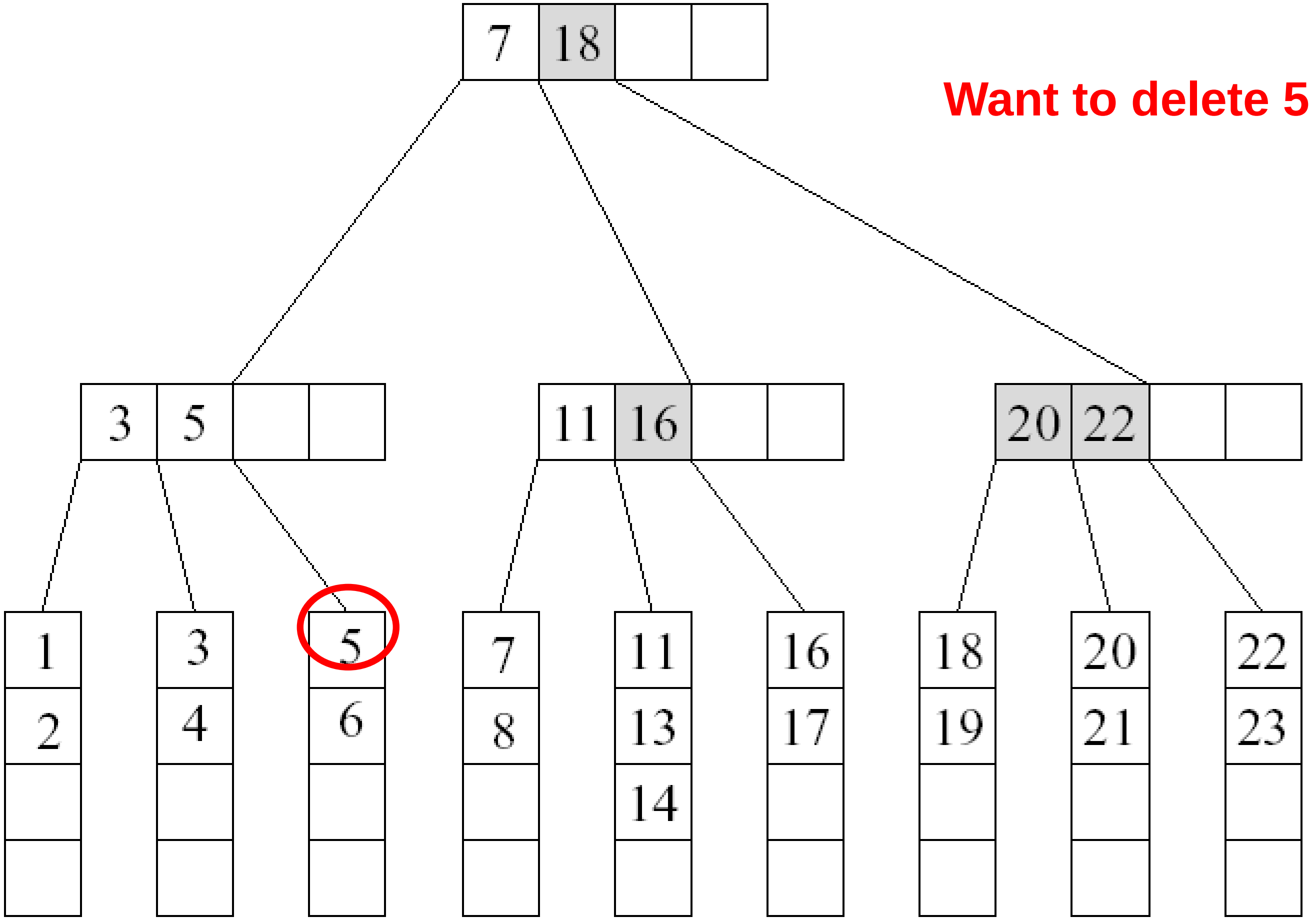


12 deleted, final step: borrow from parent and right sibling

# Deleting a key in an internal node

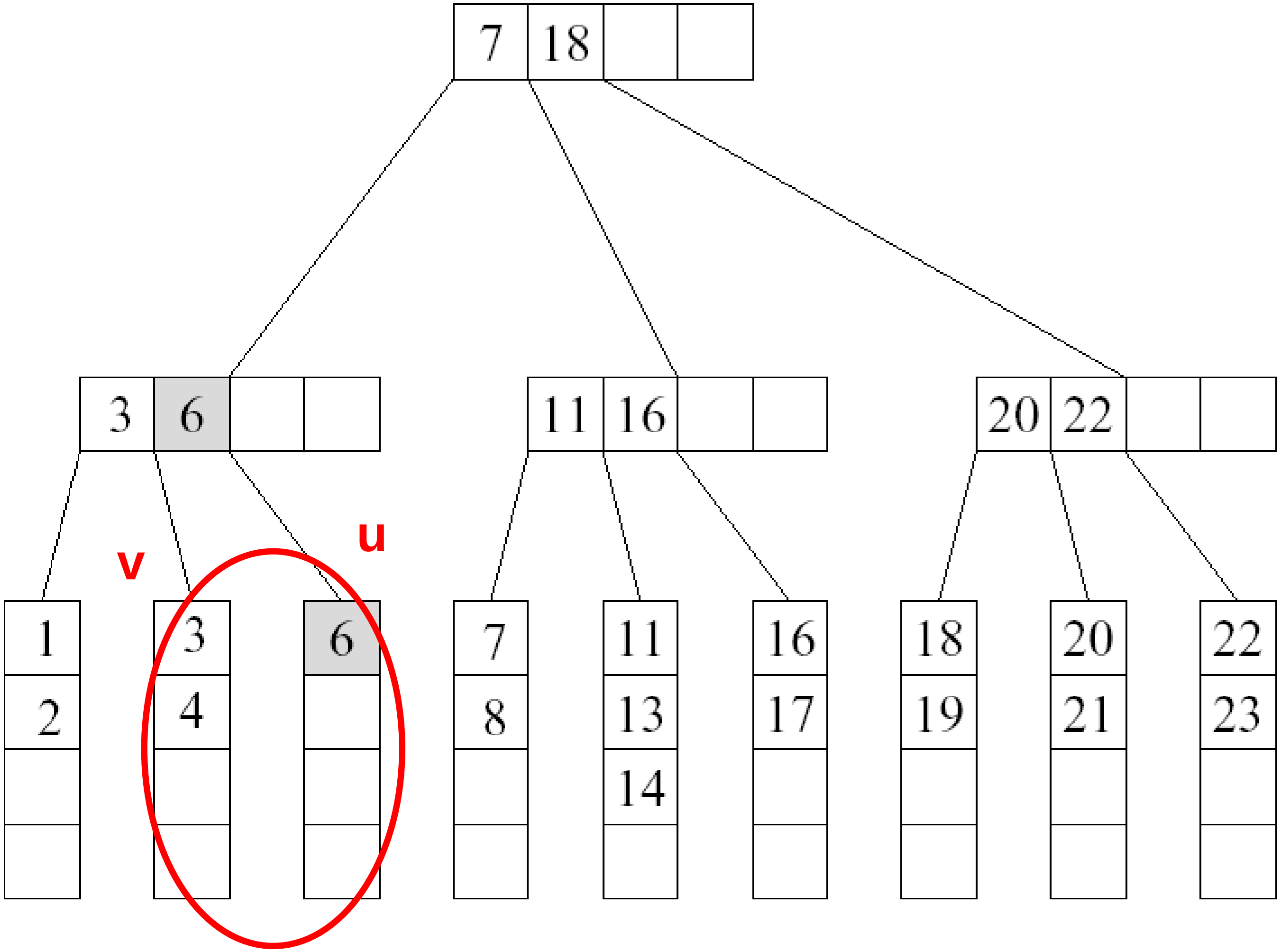
- Case 3: all sibling v of u contains exactly $\lceil M/2 \rceil - 1$ keys
  - Move the separating key between u and v in the parent of u and v down to v
  - Move the keys and child pointers in u to v
  - Remove the pointer to u at parent.

# Example (M=5, L = 4)



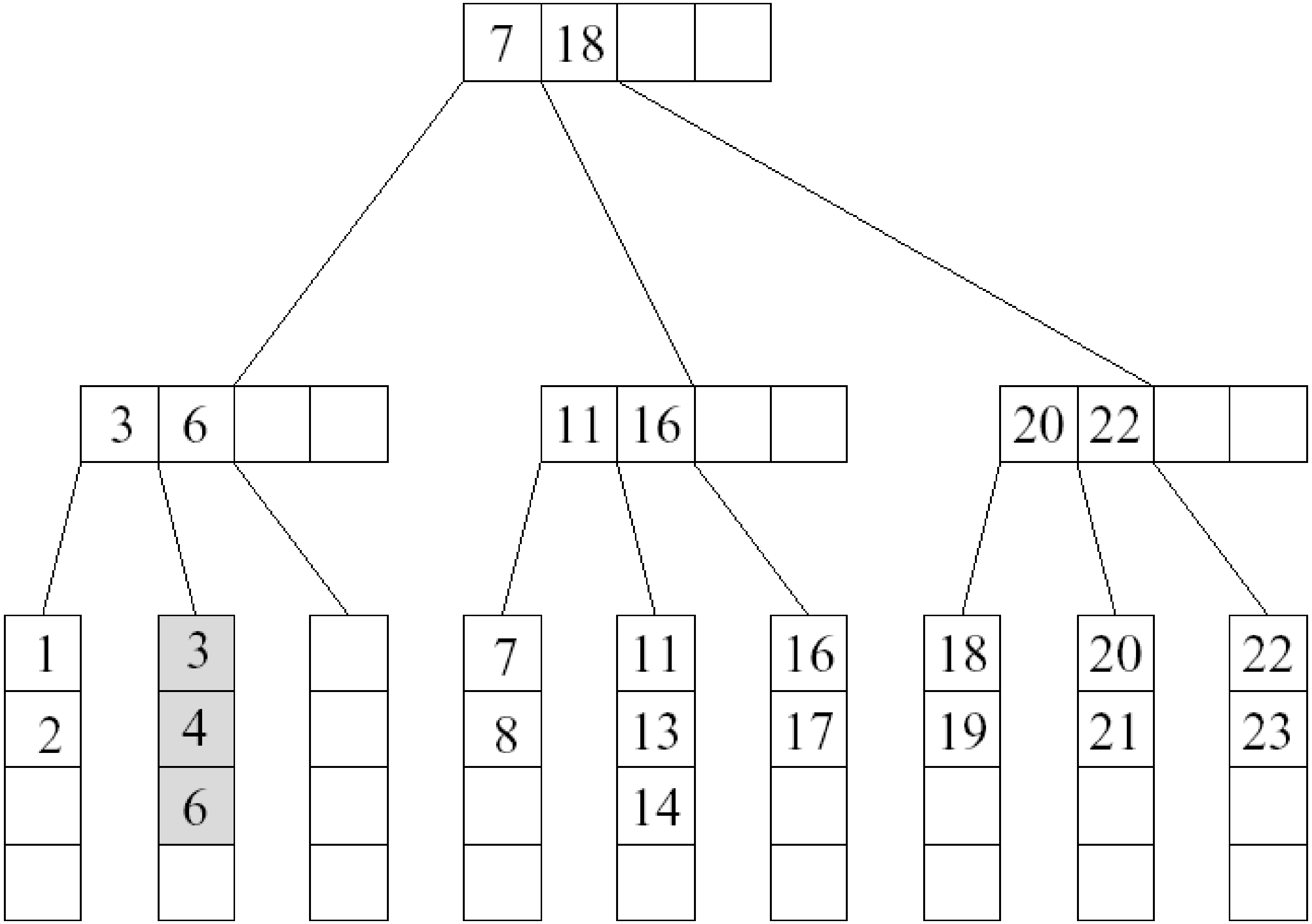
Want to delete 5

# Example (Cont'd)



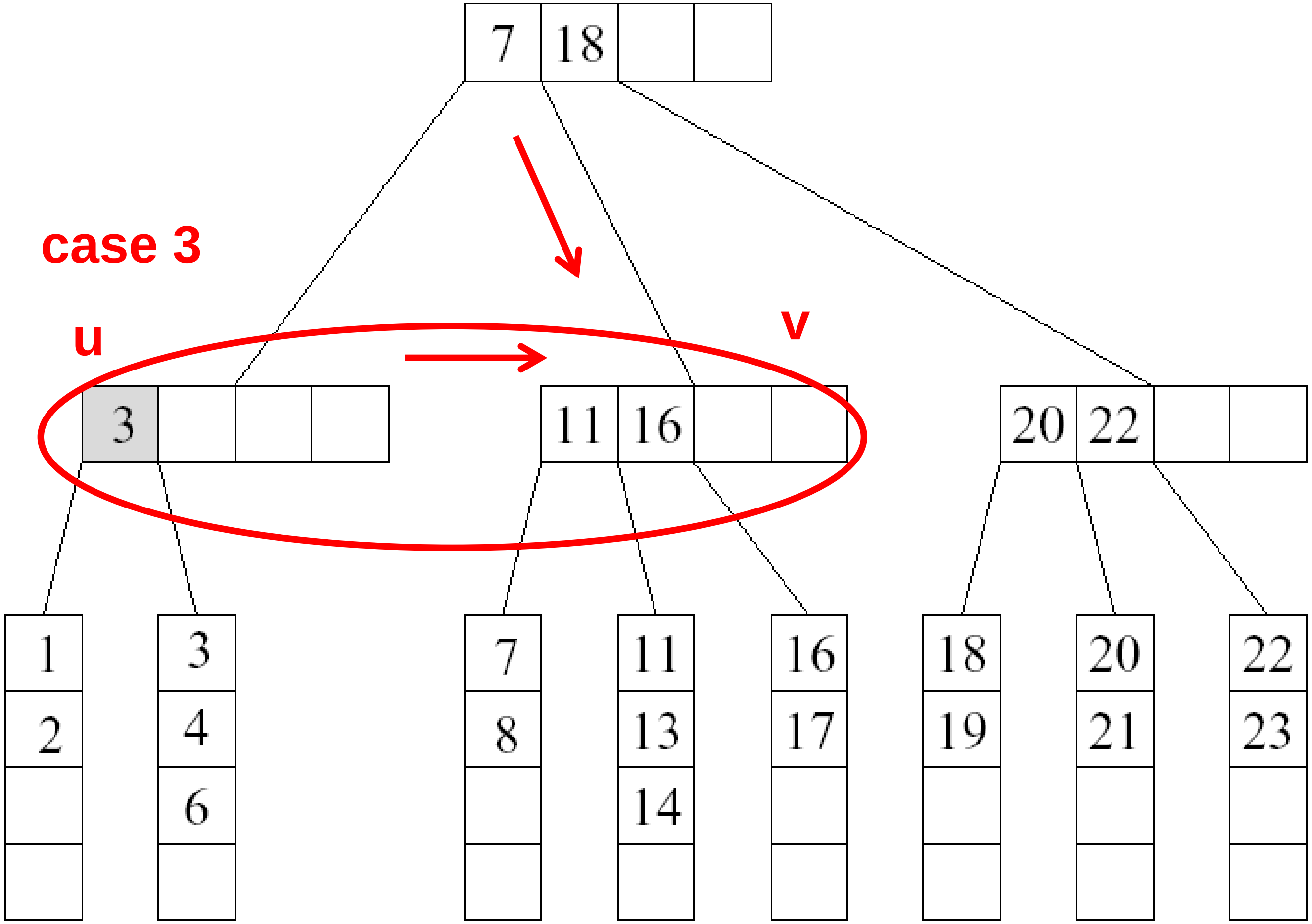
5 deleted, step 1

# Example (Cont’d)



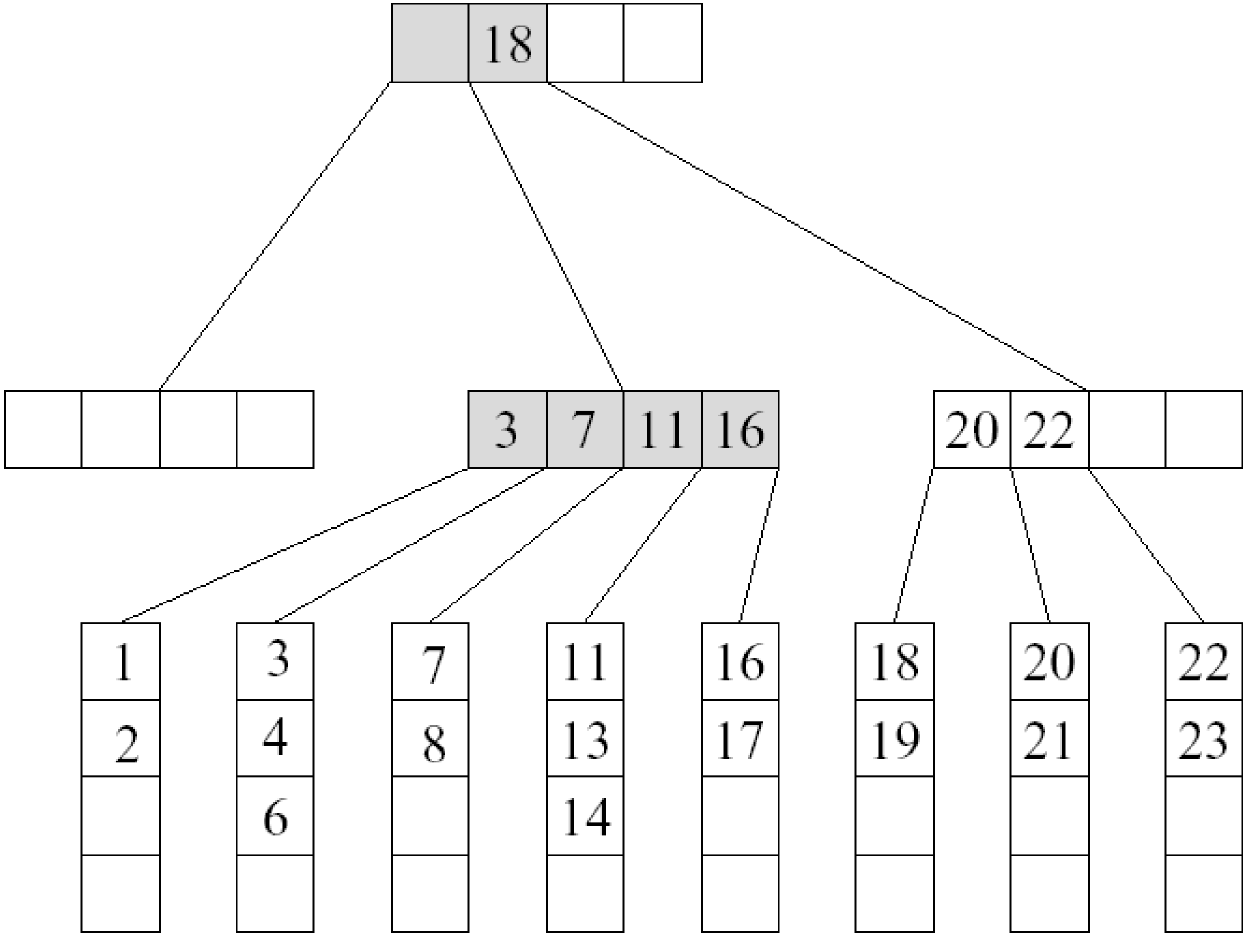
5 deleted, merge with left sibling

# Example (Cont'd)



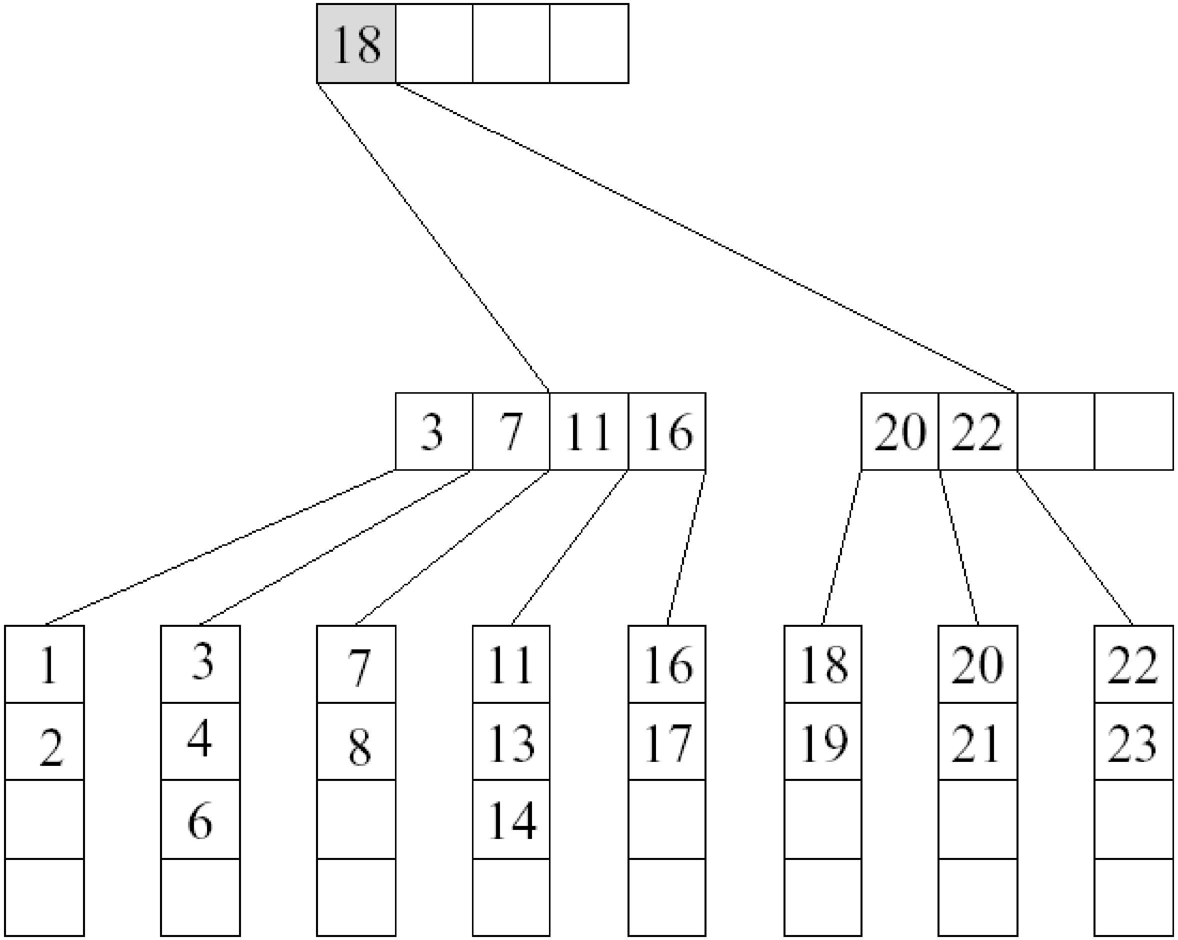
5 deleted, delete the empty leaf and the separating key 6

# Example (Cont’d)



5 deleted, borrow from parent and merge with right sibling

# Example (Cont'd)



5 deleted, delete empty internal node