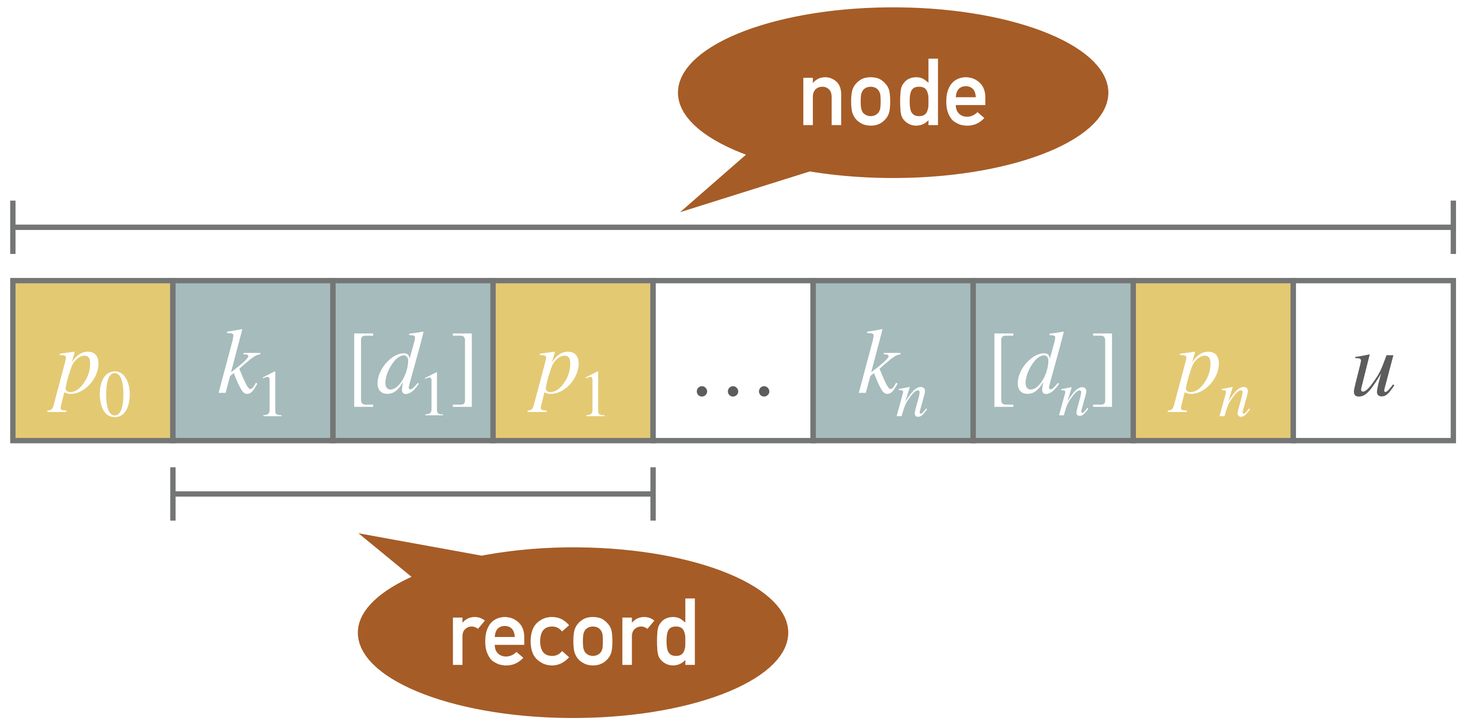
B-Trees

NDBI007: Practical class 5

B-Tree

- B-Tree of *degree* m is *balanced* m*-ary* tree where:
  - The *root* has at least 2 children unless it is a *leaf*
  - Every *inner node* have at least $\left\lceil \frac{m}{2} \right\rceil$ and at most m *children*
  - Every inner node contains at least $\left\lceil \frac{m}{2} \right\rceil - 1$ and at most $m - 1$ data entries (e.g., keys, pointers)
  - All the *paths* from the root to the leaf are of *the same length*

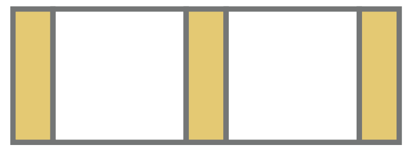
- The nodes have the structure $p_0, (k_1[,\ d_1],\ p_1), (k_2[,\ d_2],\ p_2), \ldots, (k_n[,\ d_n],\ p_n), u$
  - p_i - *pointers* to the children
  - k_i - *keys*
  - d_i - *data* or pointers to them
  - u - unused space
  - where $\left\lceil \frac{m}{2} \right\rceil - 1 \leq n \leq m - 1$



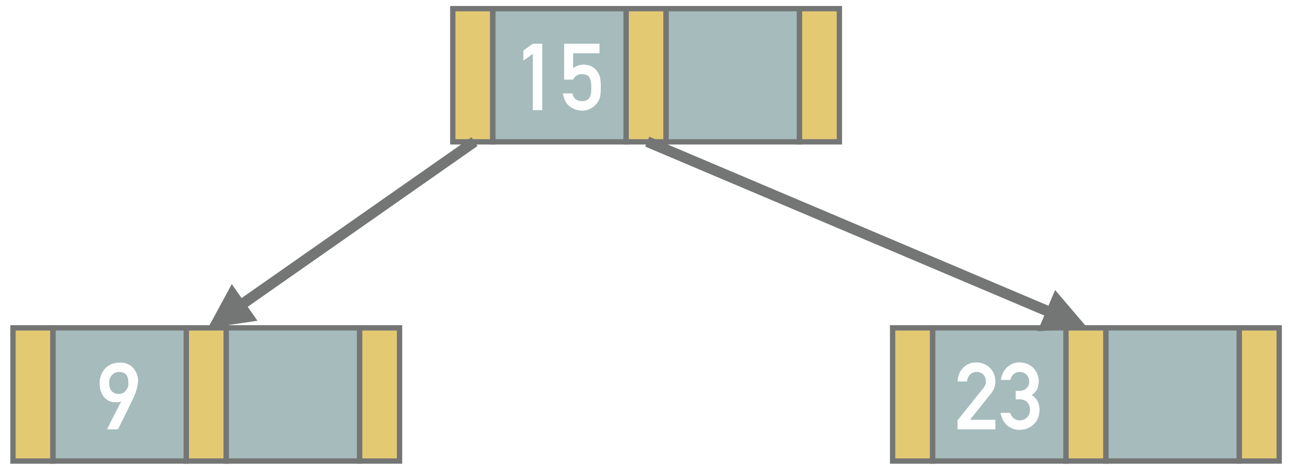
- Records $(k_i[,\ d_i],\ p_i)$ are *sorted* with respect to k_i
- Keys k_i in the subtree pointed by p_i are greater than or equal to k_i and less than i_{i+1}

Example 5.1: Insert (Splitting the Root)

- Insert entries with keys 15, 9, and 23 into an empty tree
  - Suppose a non-redundant B-tree of degree $m = 3$
    - The inner nodes have between $\lceil 3/2 \rceil$ and 3 children, i.e., they contain between 1 and 2 keys

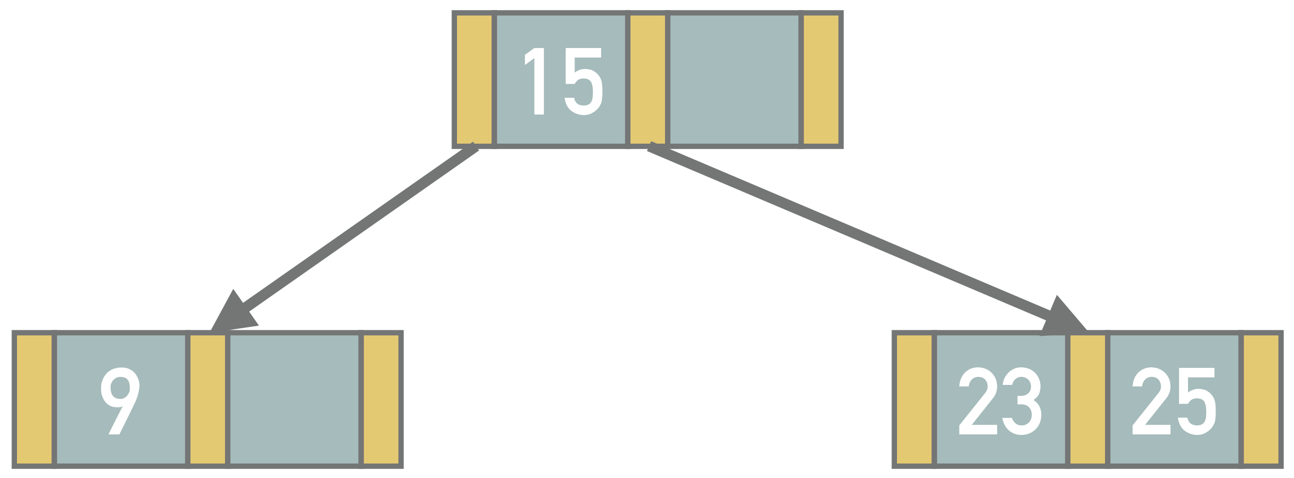


- The records with keys 15 and 9 fit into a single (root) node



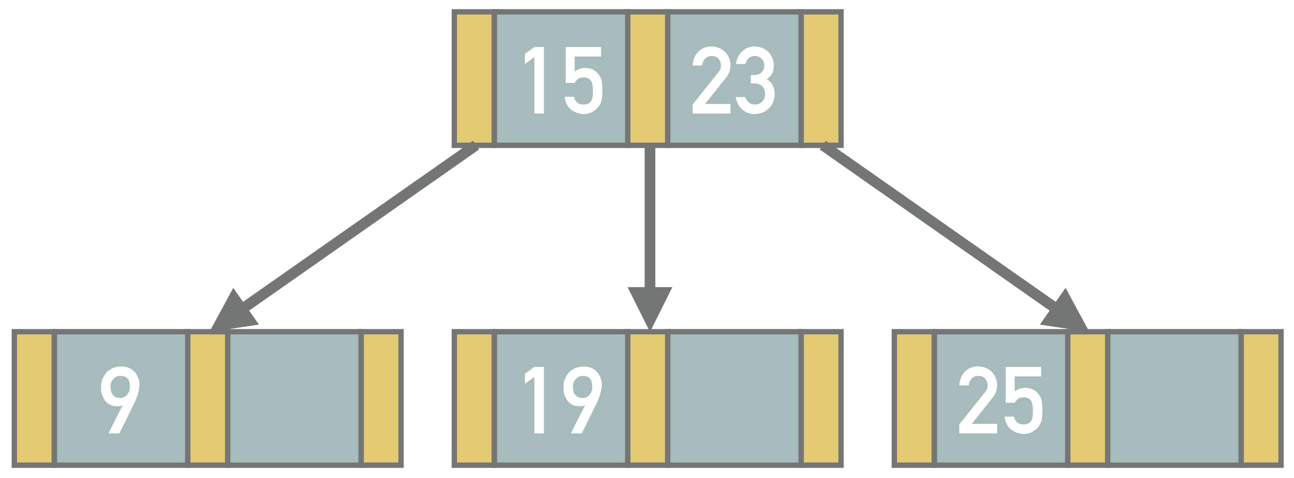
- The record with key 23 does not fit and causes splitting
  - First, we order the keys 15, 9, and 23 in ascending order, i.e., 9, 15, and 23
  - The middle key (i.e., 15) will divide the smaller keys (i.e., 9) in one node from the bigger keys (i.e., 23) in a new node
  - The dividing key will be placed into the parent node (i.e., new root node)

Example 5.2: Additional Inserts

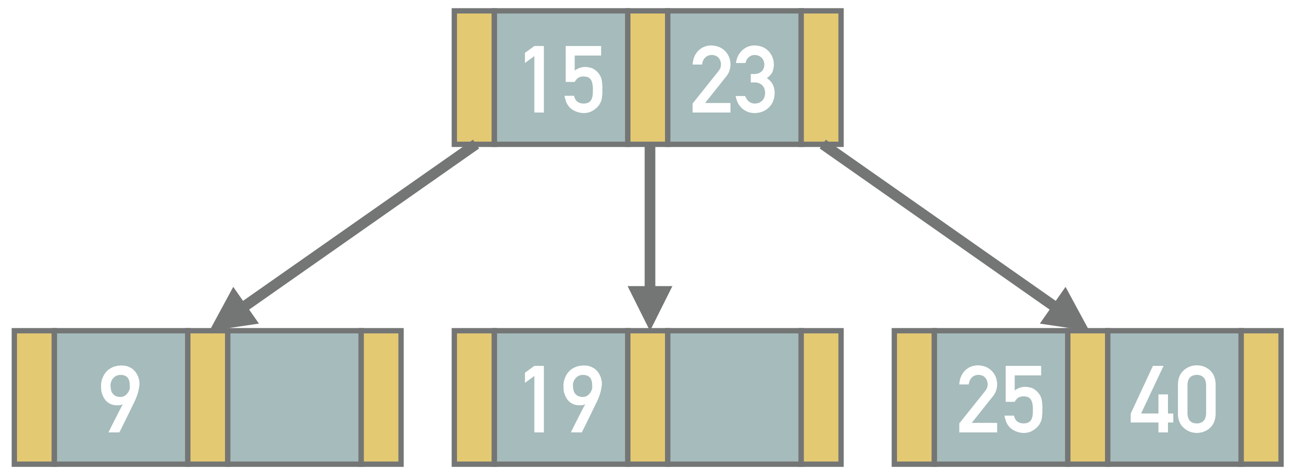


- Insert records with keys 25, 19, and 40 into B-tree from previous example

- The record with key 25 fits into the (right) leaf



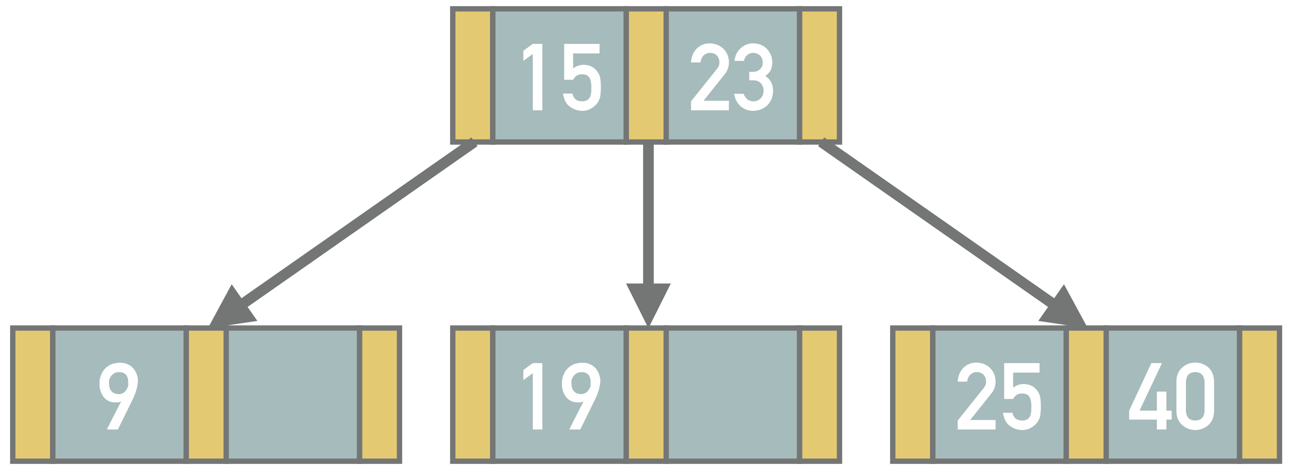
- The record with key 19 will split the (right) node into two nodes, i.e., (19) and (25) with (23) being the dividing record
  - The dividing record (23) finds its place in the parent node



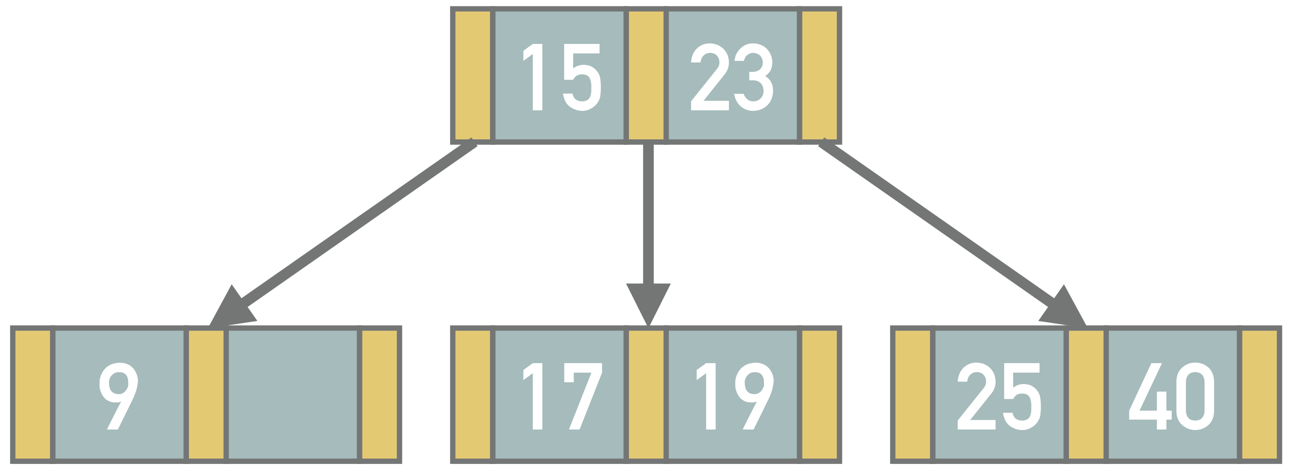
- The record 40 will fall into the right node

Example 5.3: Insert (Propagation)

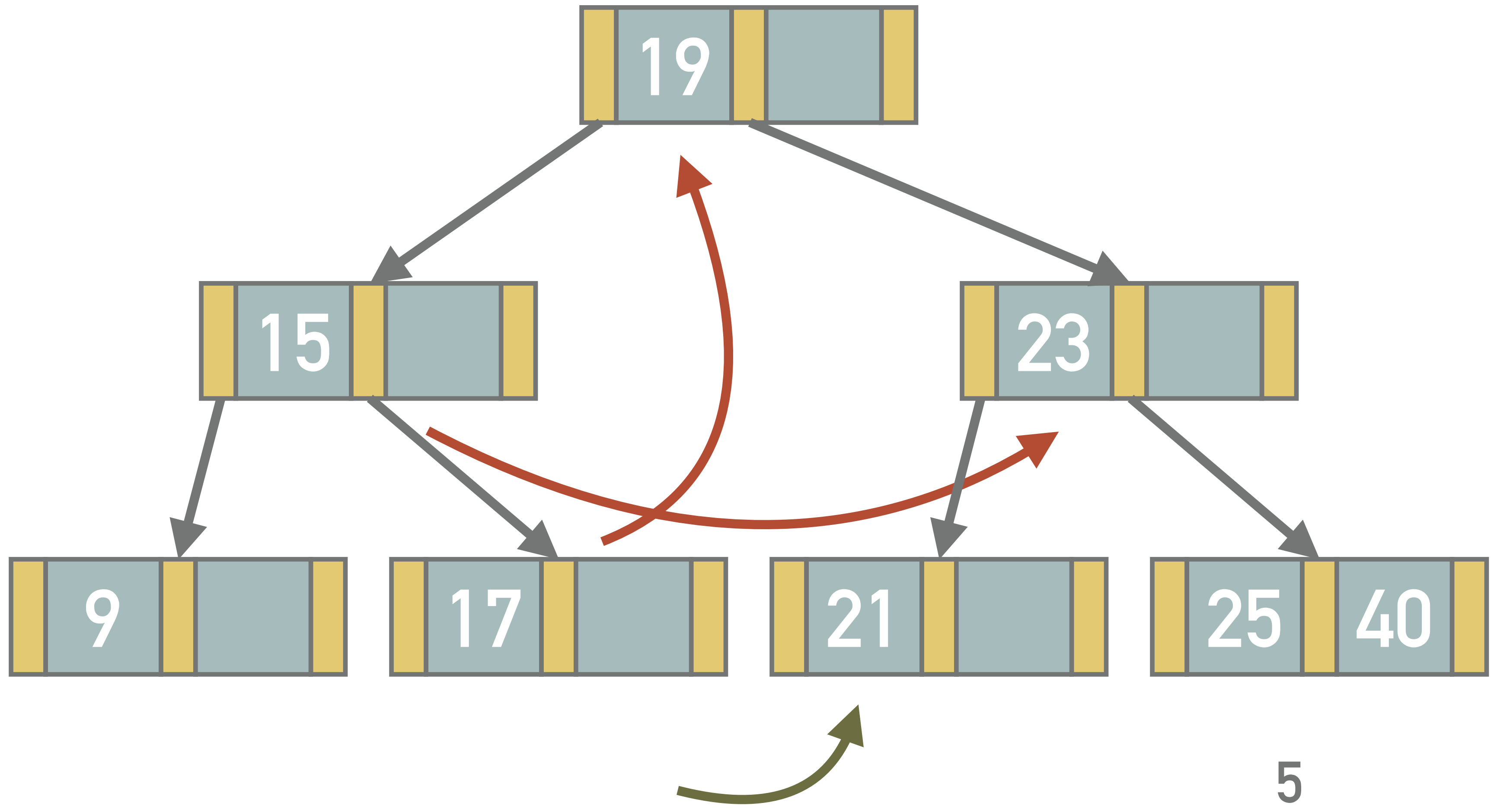
- Insert records with keys 17 and 21 into the B-tree from previous example



- The record 17 falls into the middle leaf

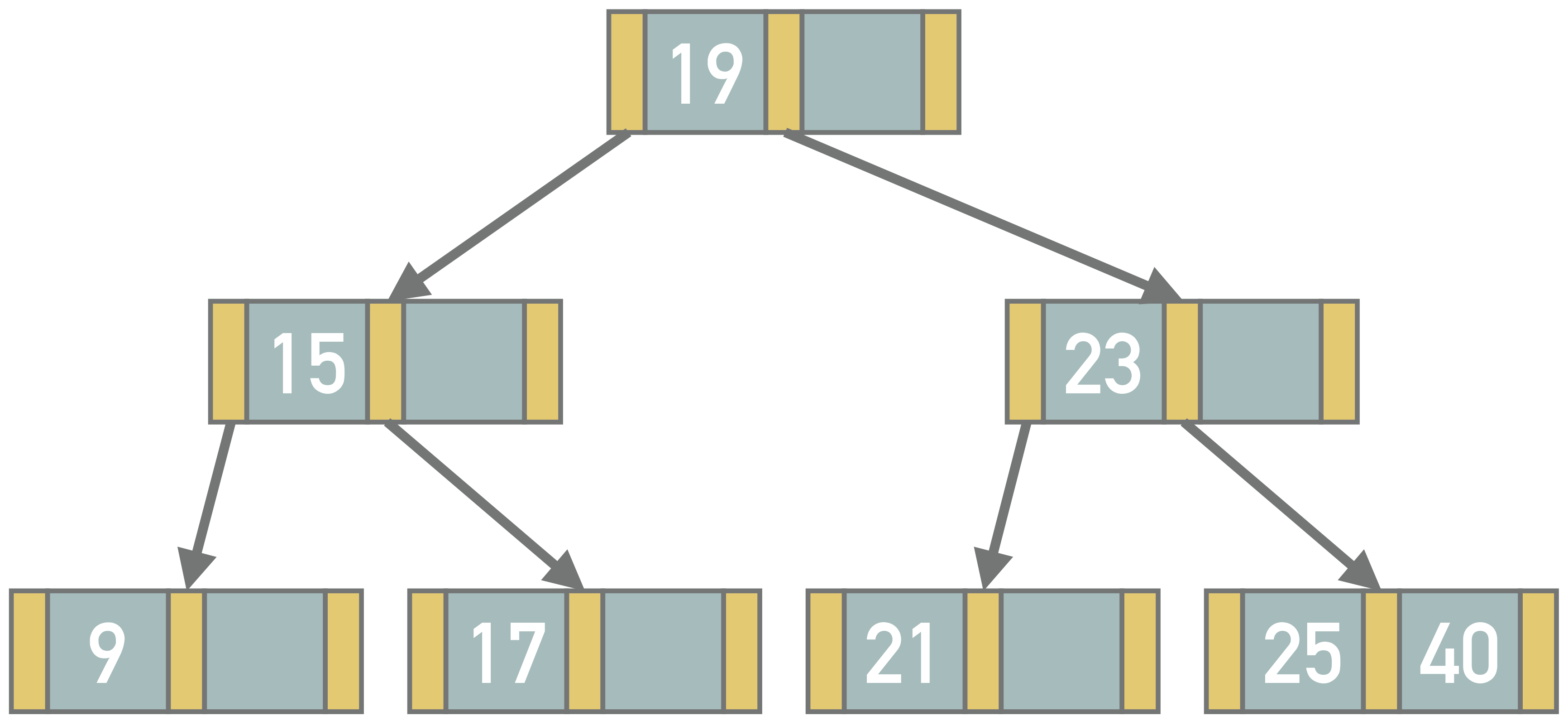


- The record 21 causes splitting of the middle leaf (17, 19, 21) and propagation of the record (19) to the parent
    - However, there is no more space in the parent node (root)
    - Thus, the parent node (15, 19, 23) needs to be split as well which increases the tree height



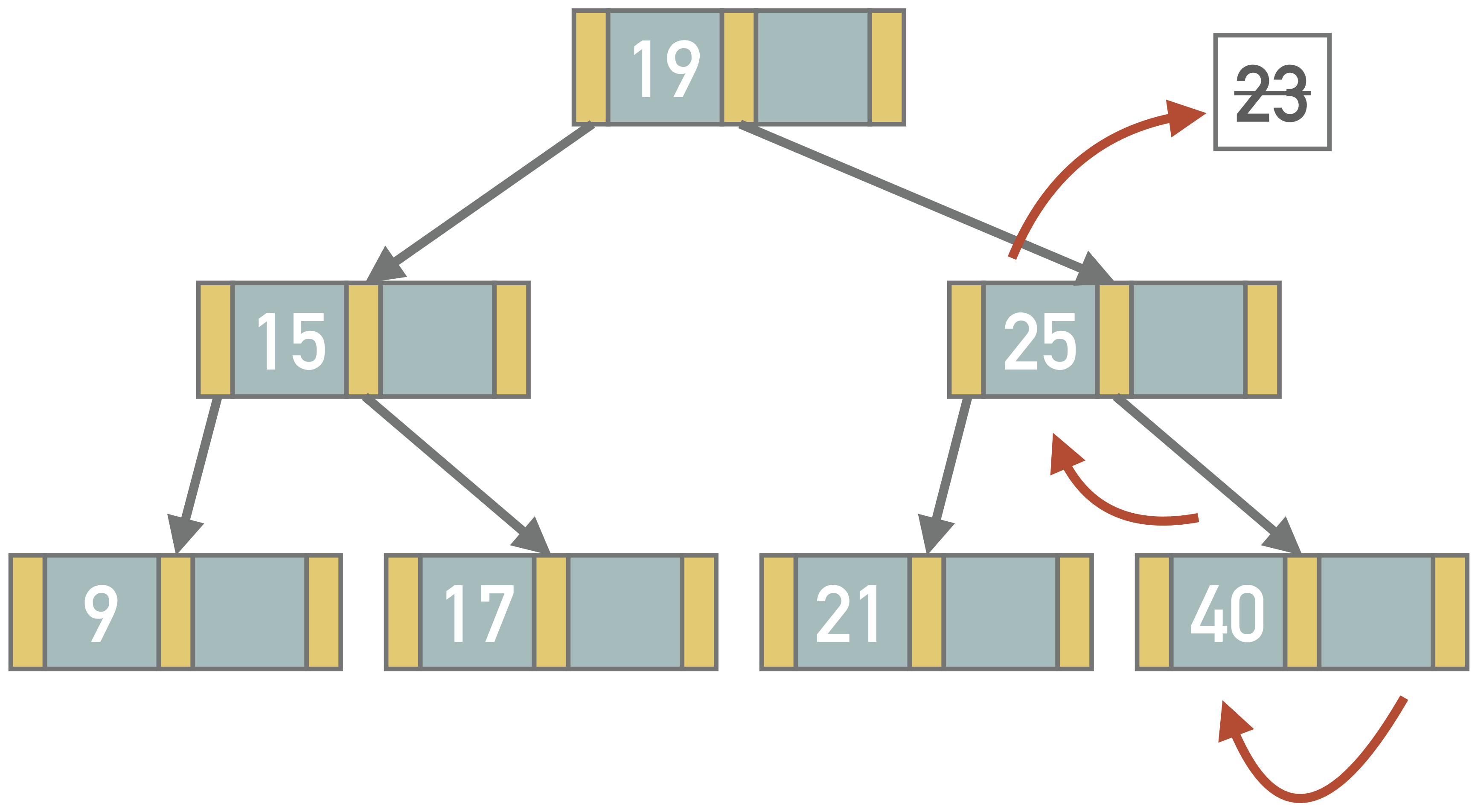
Example 5.4 Delete

❖ Remove record with key 23 from the non-redundant B-tree of degree 3 (see the upper figure)



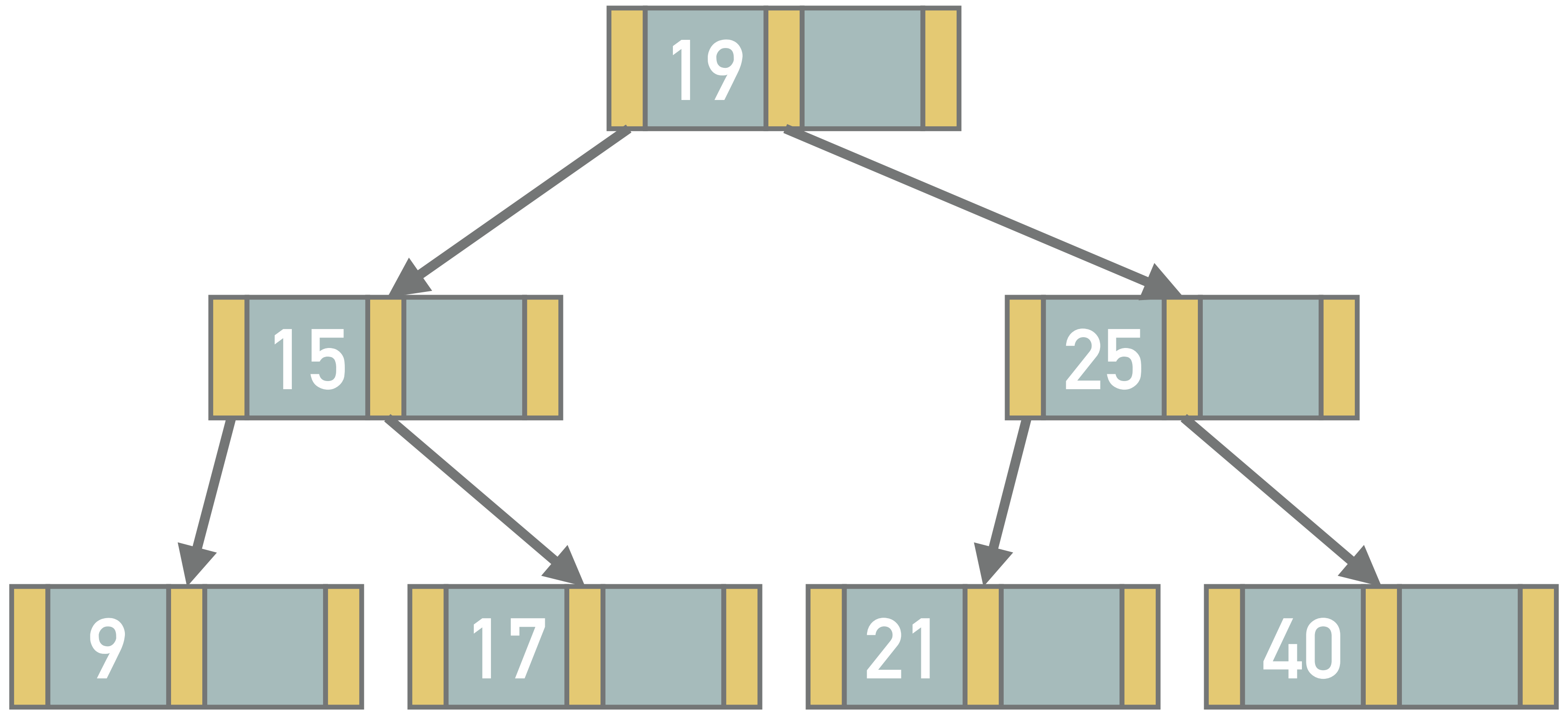
❖ The deletion of a data entry from an inner node leads to its replacement with the most left descendant entry from the right subtree or the most right entry from its left subtree

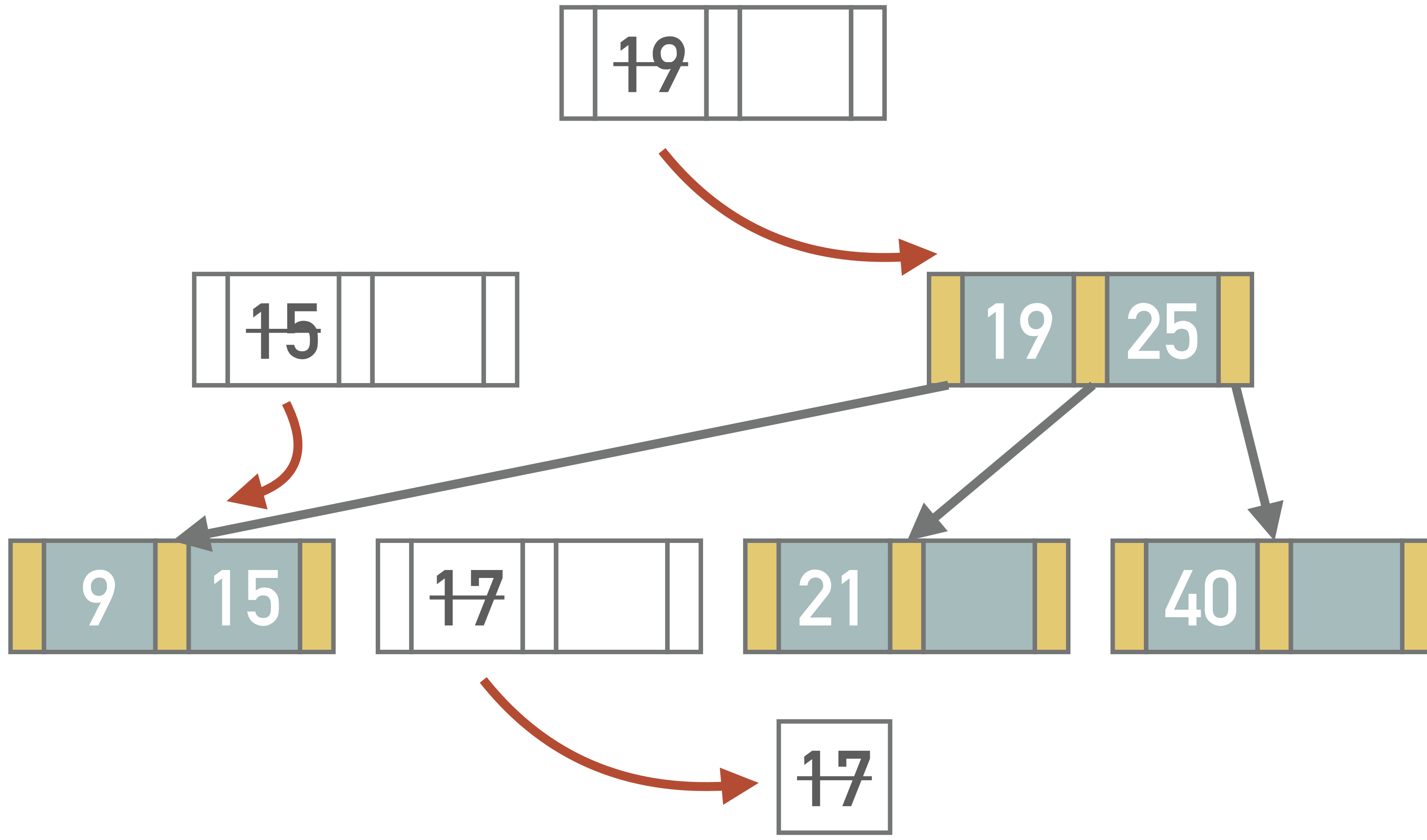
- ❖ If we delete 23 from the tree above, we can replace it with entry 25 from the bottom node (leaf)
- ❖ Moving the entry 25 from the leaf (25, 40) is safe since it still has the minimum number of entries



Example 5.5: Delete (Merging)

- Remove record with key 17 from the non-redundant B-tree of degree 3 (see the upper figure)
- We cannot borrow an entry from the neighbor (9) since it also contains the minimal number of entries
- Therefore we have to merge nodes (9), (empty), and 15
  - The entries of the current node (none left after removing 17), those from the neighboring node (9) and the dividing node will be moved into a single node (9, 15)
    - Thus, the entry 15 needs to be removed from the parent node which causes underflow of that node
- We have to merge nodes (empty parent node), (19) and (25)
  - Once again, we cannot borrow an entry from the neighbor node (25)
  - The empty node (empty) is merged with the node (25) and dividing entry (19) from the root node, resulting in the node (19, 25)
  - Having entry 19 removed from the root (empty), the height of the tree decreases



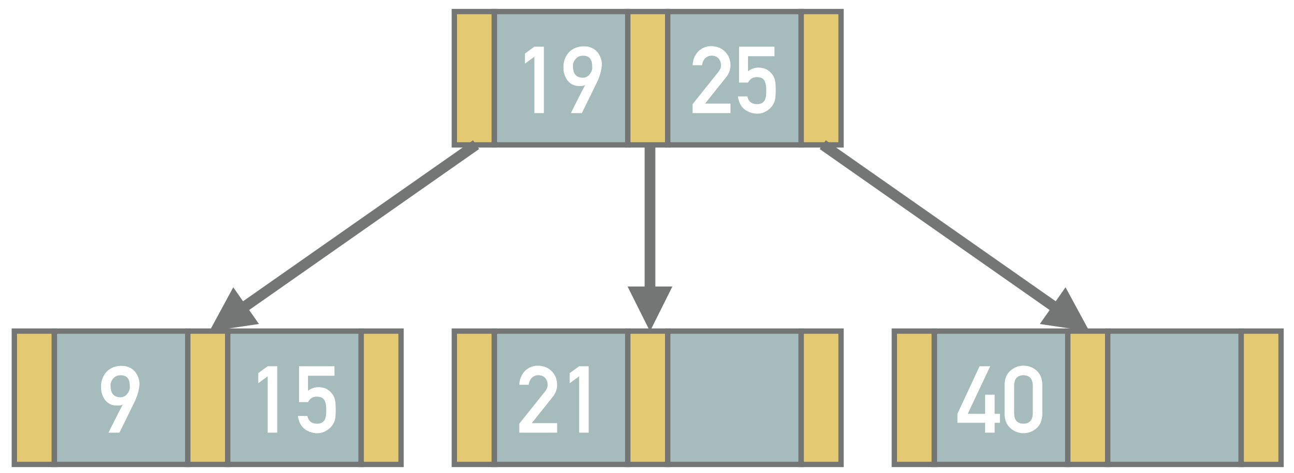


Exercise 5.6

- Suppose a non-redundant B-tree of degree $m = 3$ (see the figure)

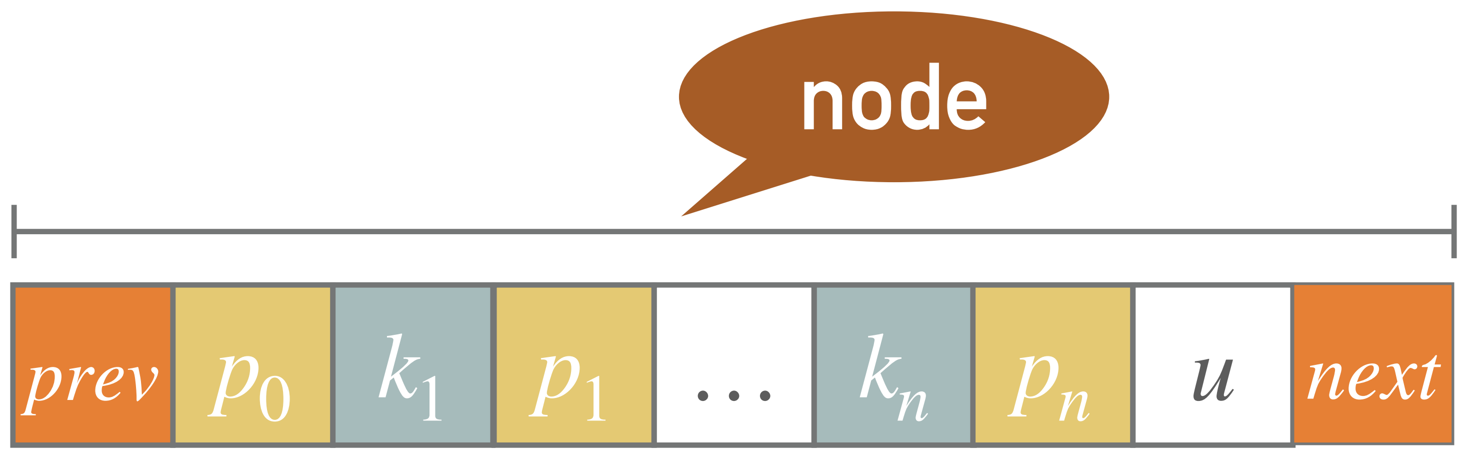
- First, illustrate the B-tree after insertion of records with keys 11, 18, and 14

- Second, illustrate the B-tree after deletion of records with keys 40, and 14



B+-Tree

- B+-Tree differs from the original B-tree by:
  - It is *always redundant*, i.e., the data are stored or pointed to from the leaf nodes
  - The *leaf nodes are chained* using pointers in a linked list which simplifies range queries
    - In reality, often all the levels are linked (not just the leaf level)
  - The inner nodes contain only the values using which the tree can be traversed

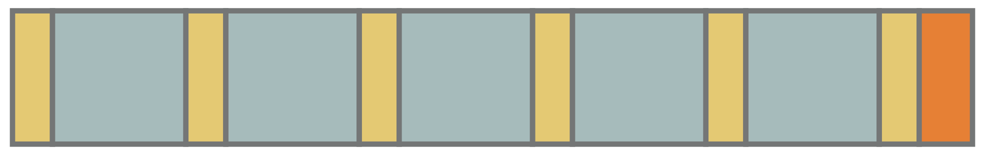


- The nodes have the structure $[prev,]\ p_0, (k_1, p_1), \dots, (k_n, p_n), u\ [,next]$
- p_i - pointers to the children or data
- k_i - keys

- Keys k_j in the subtree pointed by p_i are greater than or equal to k_i and less than k_{i+1}, if k_{i+1} exists
- The minimum number of children can be raised to $\lceil (m+1)/2 \rceil$

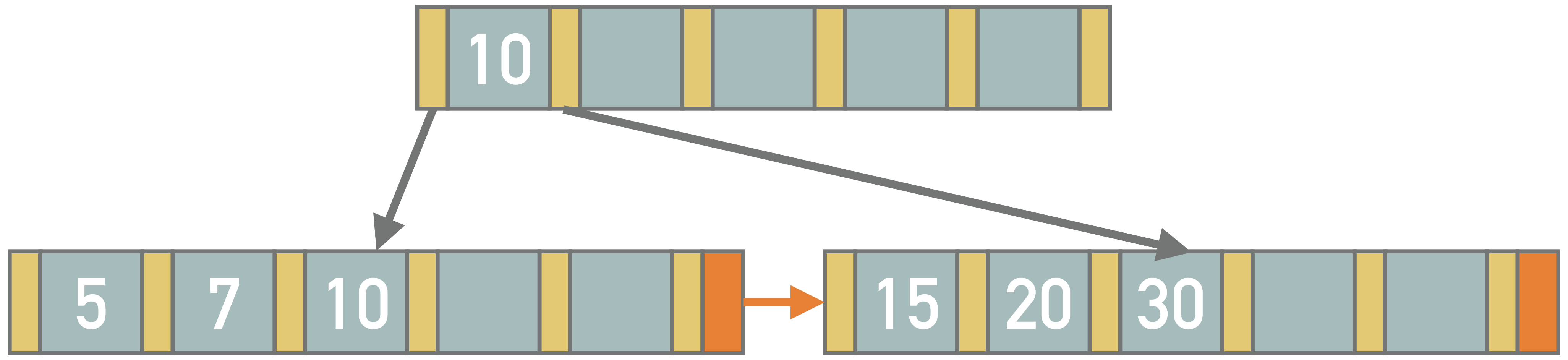
Example 5.7: Insert

- Insert records with keys 10, 7, 15, 5, 30, and 20 into an empty B^+-tree
  - Suppose a B^+-tree of degree $m = 6$
    - Hence, the minimum number of children is 3+1 (modified)



- Insertion of keys 10, 7, 15, 5, and 30 is trivial, all belong to the root node

- Insertion of key 20 leads to a page split
  - A half of the records, i.e., (5, 7, 10), stays in the original page while the rest, i.e., (15, 20, 30), moves into a new page
  - The maximal key value in the left node, i.e., 10, is propagated into the higher level (new root note)
    - However, any value $10 \leq value \leq 14$ would work



Example 5.8: Additional Inserts

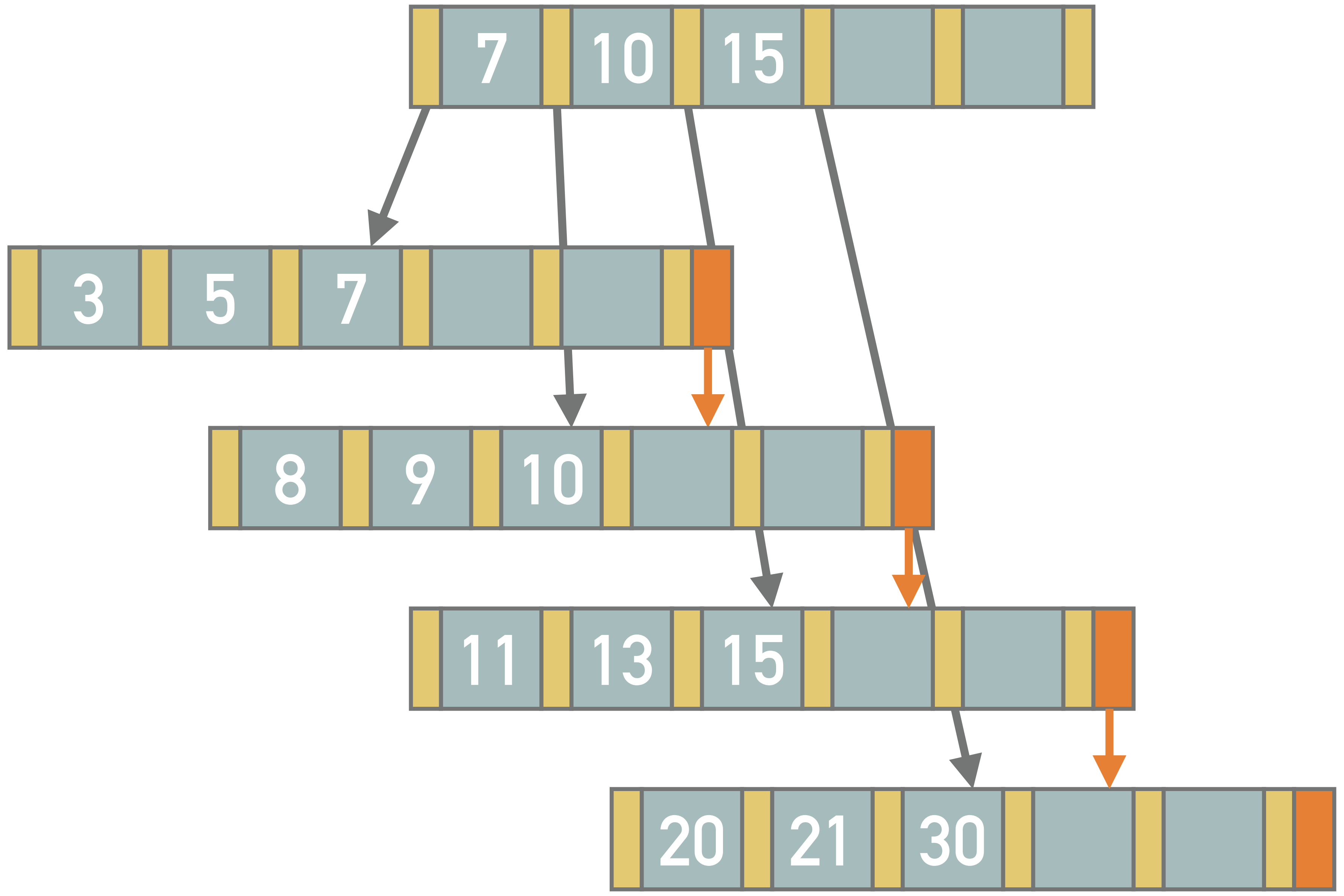
- Insert additional records with keys 13, 3, 11, 21, 8, and 9 into the B+-Tree from the previous example

- The insertion of records with keys 13, 3, and 11 is trivial
- The insertion of a record with key 21 splits the right leaf node into nodes (11, 13, 15) and (20, 21, 30)
  - The separating value (i.e., 15) is inserted into the parent node where there is enough space so it does not lead to another split
- Inserting of records with keys 8 and 9 leads to the split of the leaf into (3, 5, 7) and (8, 9, 10)
  - The separation value 7 is inserted into the parent node

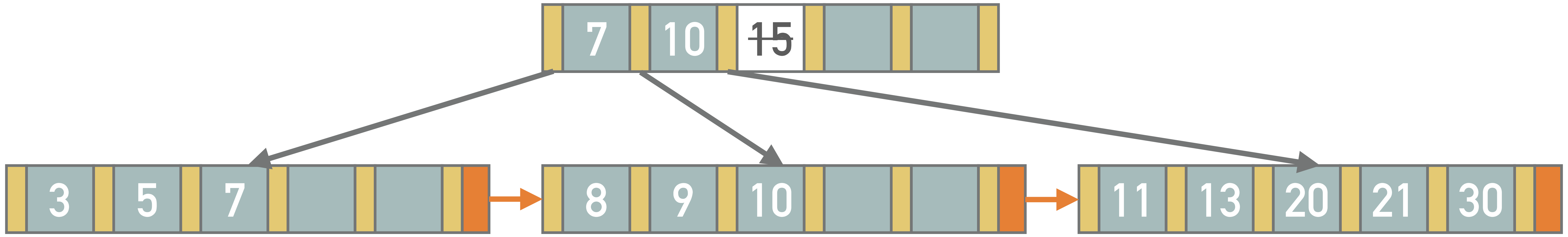
Example 5.9: Delete (and Merge Nodes)

- Remove the record with key 15 from the B+-Tree

- When removing keys from a B+-Tree, the given key is simply removed from the leaf unless the corresponding leaf underflows
  - In such case, the tree tries to borrow a key from a sibling leaf and to change the splitting value
  - If also the neighbors have the minimum number of entries, it is necessary to merge two nodes into one and remove the splitting value from the parent
    - Which can lead to the merge cascade up to the root

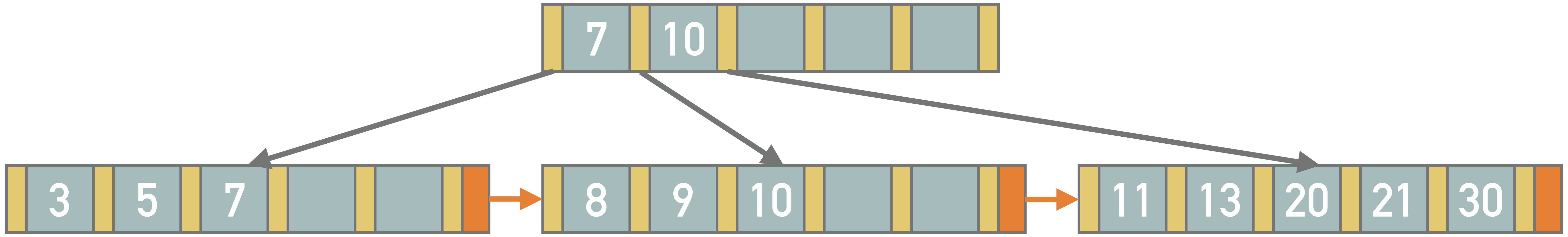


- In our example, every node (except the root) needs to include at least three keys
  - Removing the key 15, the condition is violated and sibling leaves cannot lose any entry either
  - Hence we merge node (11,13) with (20, 23, 30) and remove the splitting value 15 from the parent

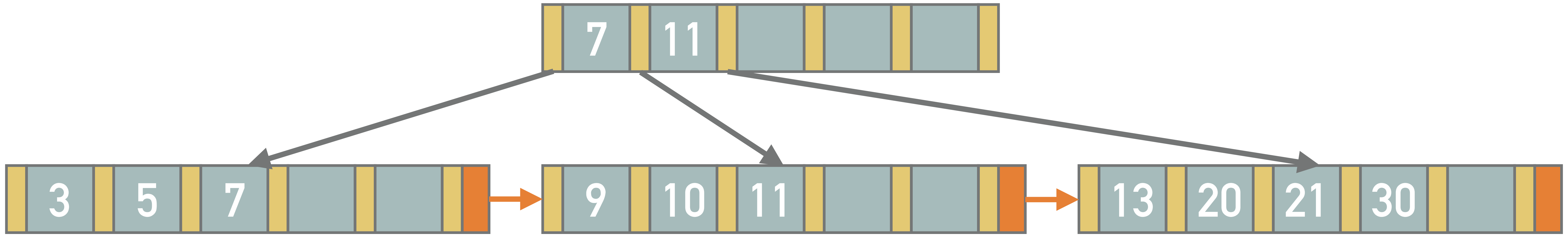


Example 5.10: Delete (Borrow Key)

- Remove the entry with key 8 from the B+-tree



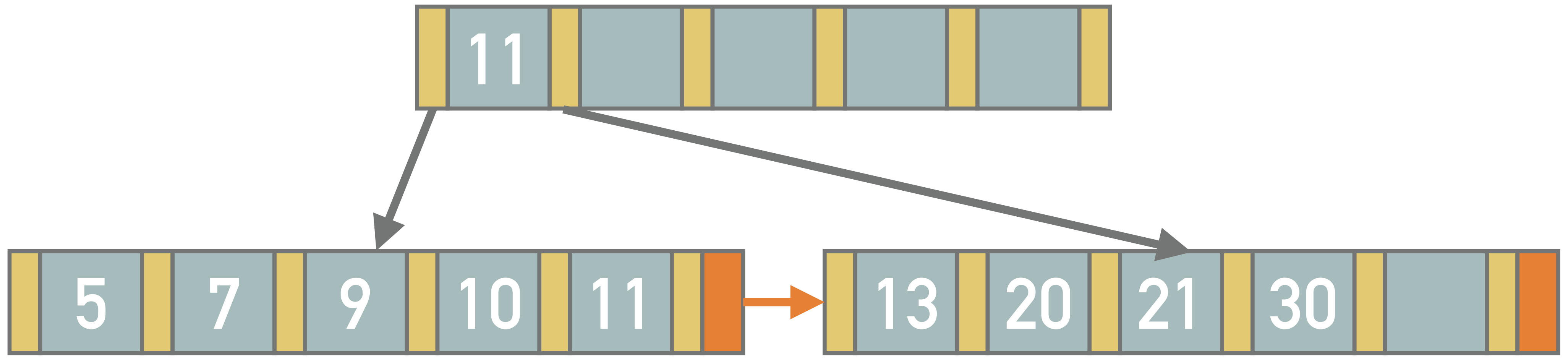
- To remove the entry 8 we need to move the entry with key 11 from the neighboring node to keep the condition of minimum number of entries in every node
  - It is necessary to change the splitting value in the parent from 10 to 11



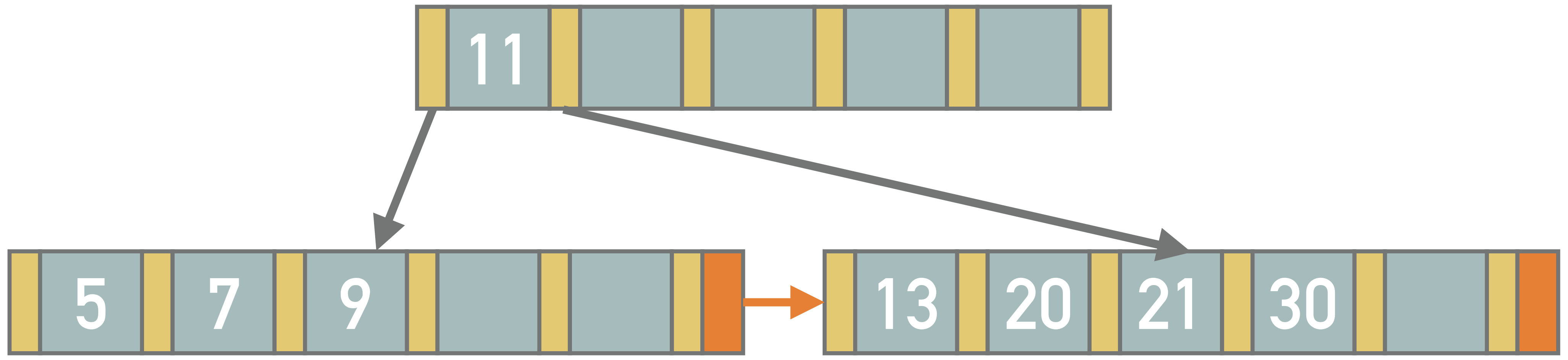
Example 5.11: Delete

- Remove records with keys 3, 10, and 11 from the B+-tree (see the previous page)

- Removing the key 3
  - After the removal, the number of records in the node (5, 7) falls under minimum and the neighboring node, i.e., (9, 10, 11), cannot provide any record
  - The nodes (5, 7) and (9, 10, 11) are merged
  - Finally, the splitting value 7 is removed from the parent

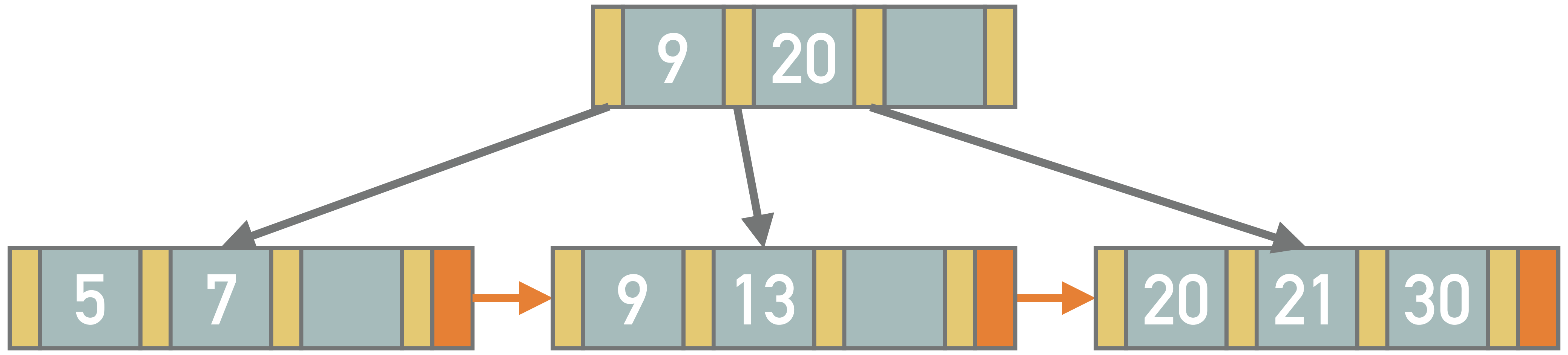


- Removing the keys 10, 11
  - It is sufficient to remove the keys from the node, no modifying of splitting value is required



Exercise 5.12

- Suppose a B+-tree of degree $m = 4$ (see the figure)
  - Minimum modified number of children of a node is 3, i.e., $\lceil (4+1)/2 \rceil$
- Illustrate the B+-tree after the insertion of keys 40, 50, and 60

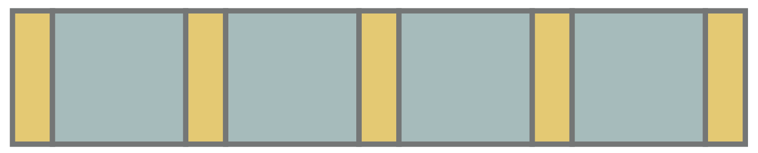


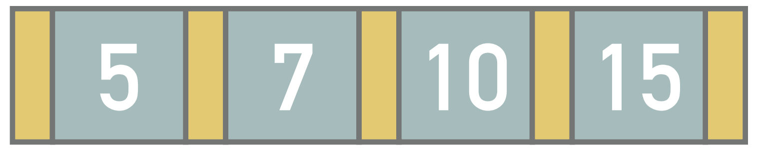
B*-Tree

- B*-tree differ from the standard B-tree by:
  - The non-root nodes have at least $\lceil (2m-1)/3 \rceil$ children
    - If the tree contains few records (i.e., after splitting the root node), the only two leafs can contain less records (about half)
  - If a node has too few items, or overflows, it is balanced using both of its neighbors
  - If a node and its neighbor are full, they are split (together with the new record) into three nodes being 2/3 filled

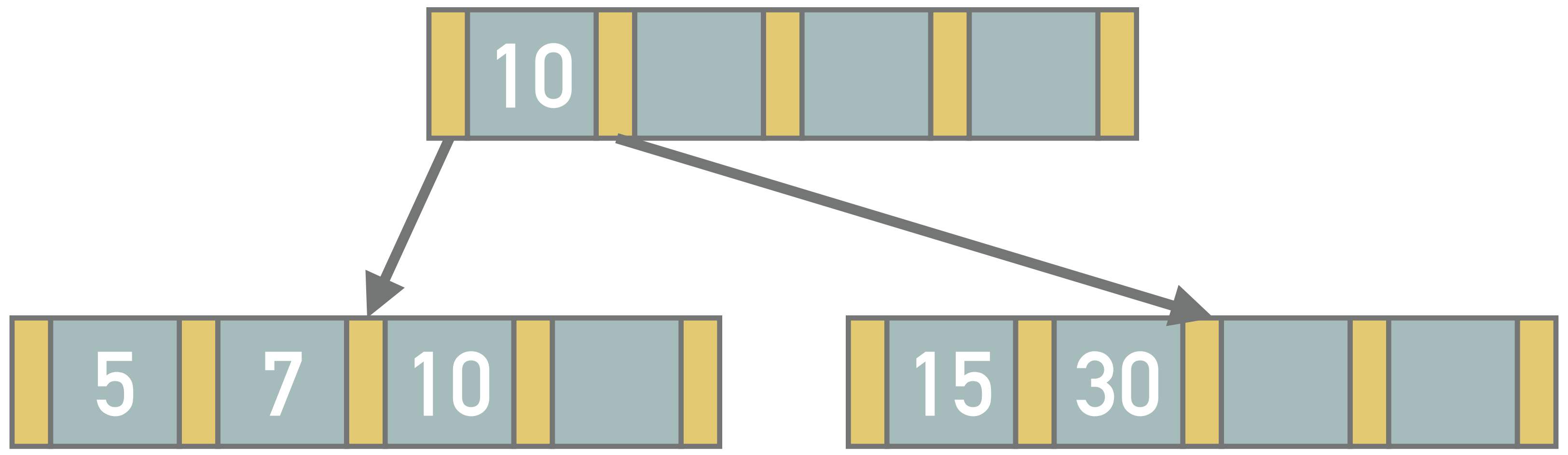
Example 5.13: Insert

- Insert records with keys 10, 7, 15, 5, 30, 20, and 13 into an empty redundant B*-tree
  - Suppose an empty B*-tree of degree $m = 5$
  - Minimum number of children is 3 and minimum number of keys is 2

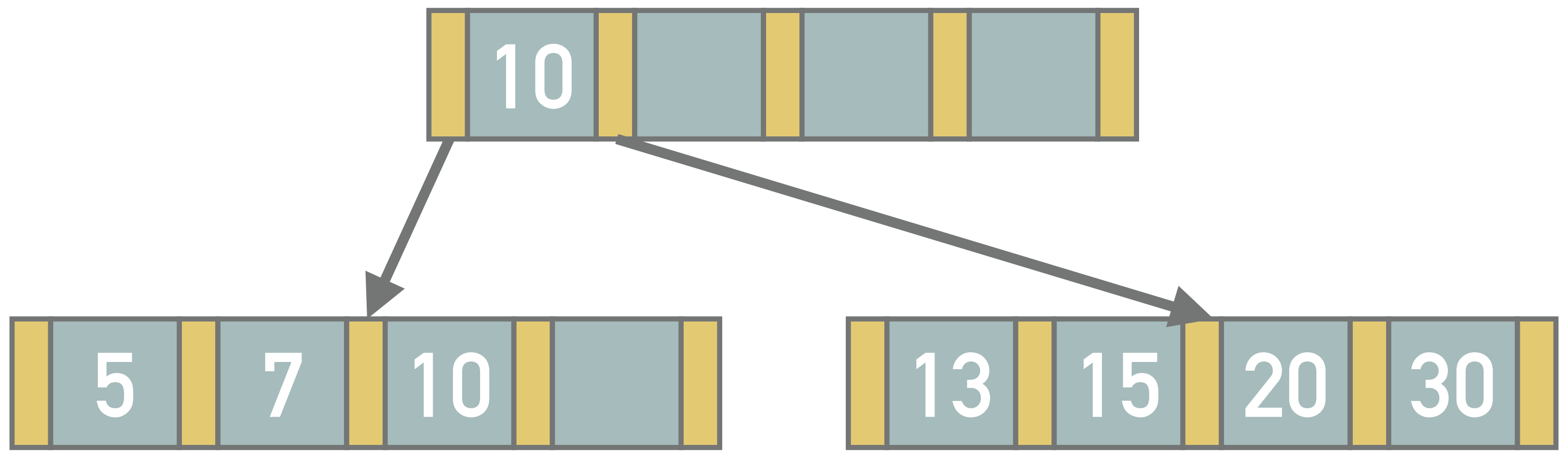




- Insertion of records with keys 10, 7, 15, and 5 is trivial, all goes to the root node



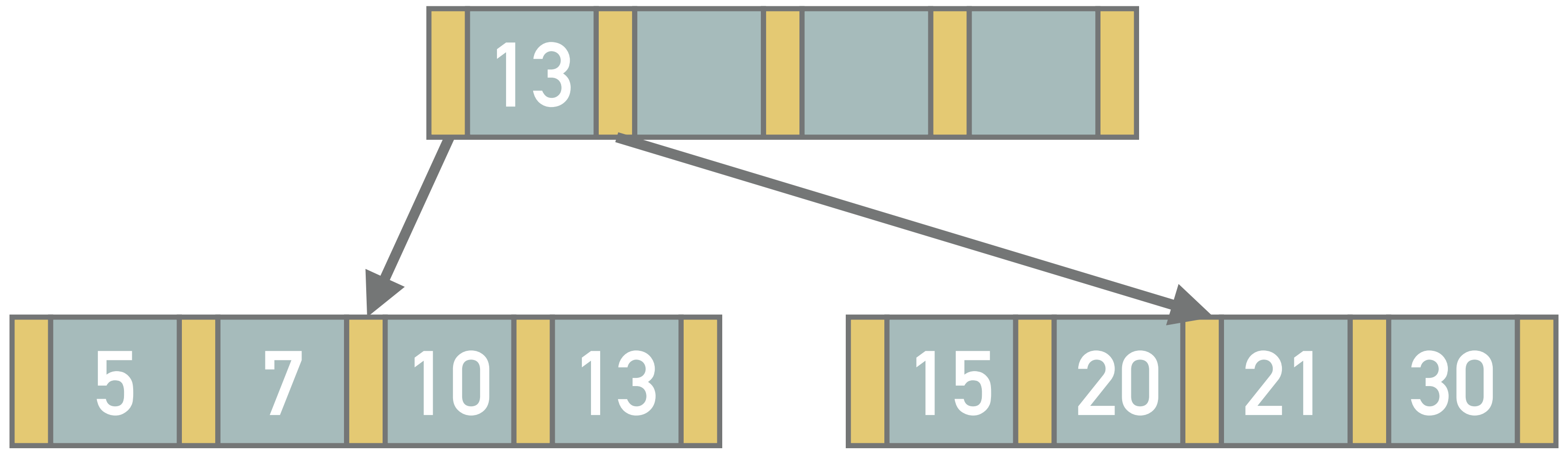
- Inserting a record with key 30 leads to root node split
  - Split nodes are (5, 7, 10) and (15, 30)
  - The dividing value 10 is inserted into the new parent (i.e., new root)



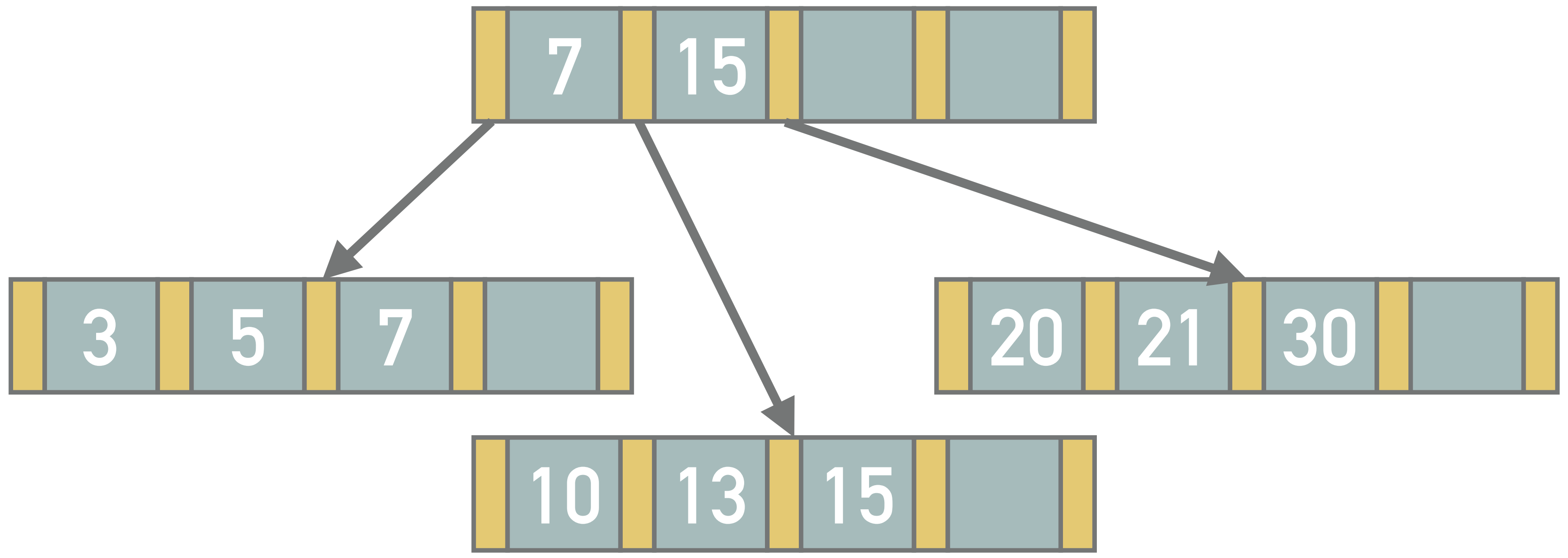
- A record with key 20 can be inserted into the right leaf, as well as a record with a key 13

Example 5.14: Additional Inserts

- Continue with the previous example and insert records with keys 21 and 3 into the redundant B*-tree



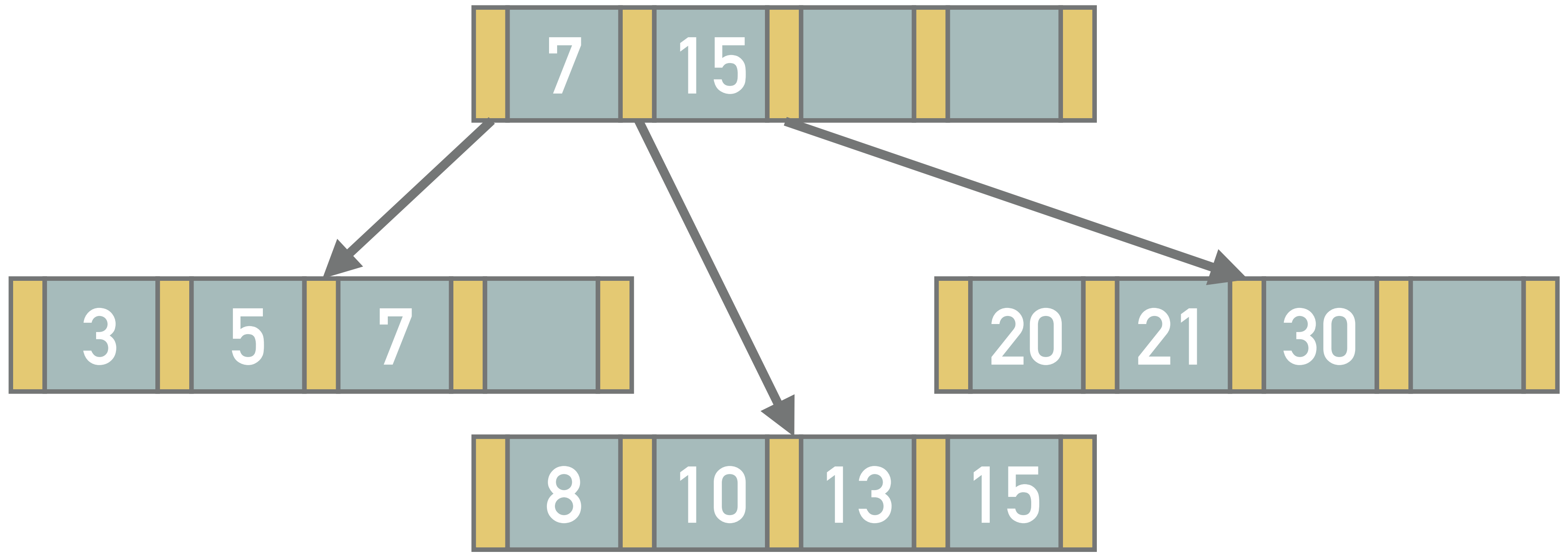
- Inserting the key 21
  - We cannot insert the key 21 into the full node (13, 15, 20, 30), but the record with key 13 can be moved to the neighboring and not yet filled node
  - The splitting value in the parent needs to be modified



- Inserting the key 3
  - The key 3 cannot be inserted into the node (5, 7, 10, 13) and the neighbor is full as well
  - The records in both nodes, together with record 3, will be split into three nodes (3, 5, 7), (10, 13, 15) and (20, 21, 30)
  - Splitting values 7 and 15 need to be inserted into the parent node instead of the existing splitting value 13

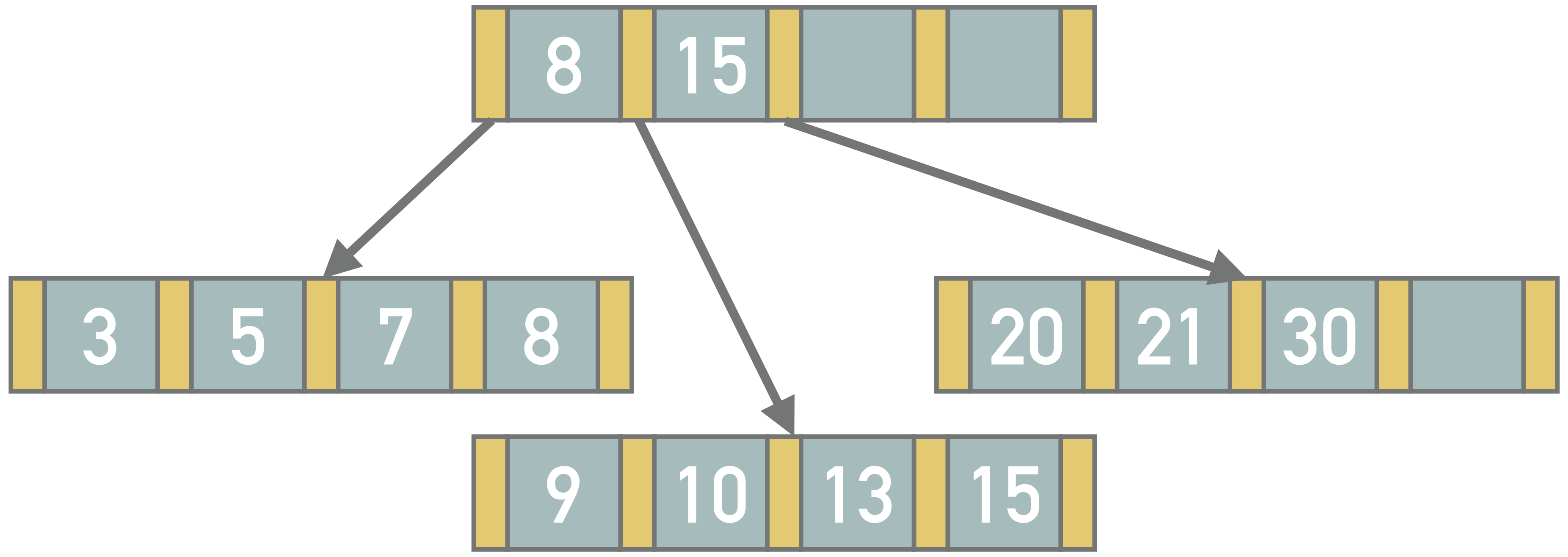
Example 5.15: Additional Inserts

- Continue with the previous example and insert records with keys 8, 9, and 11 into the redundant B*-tree

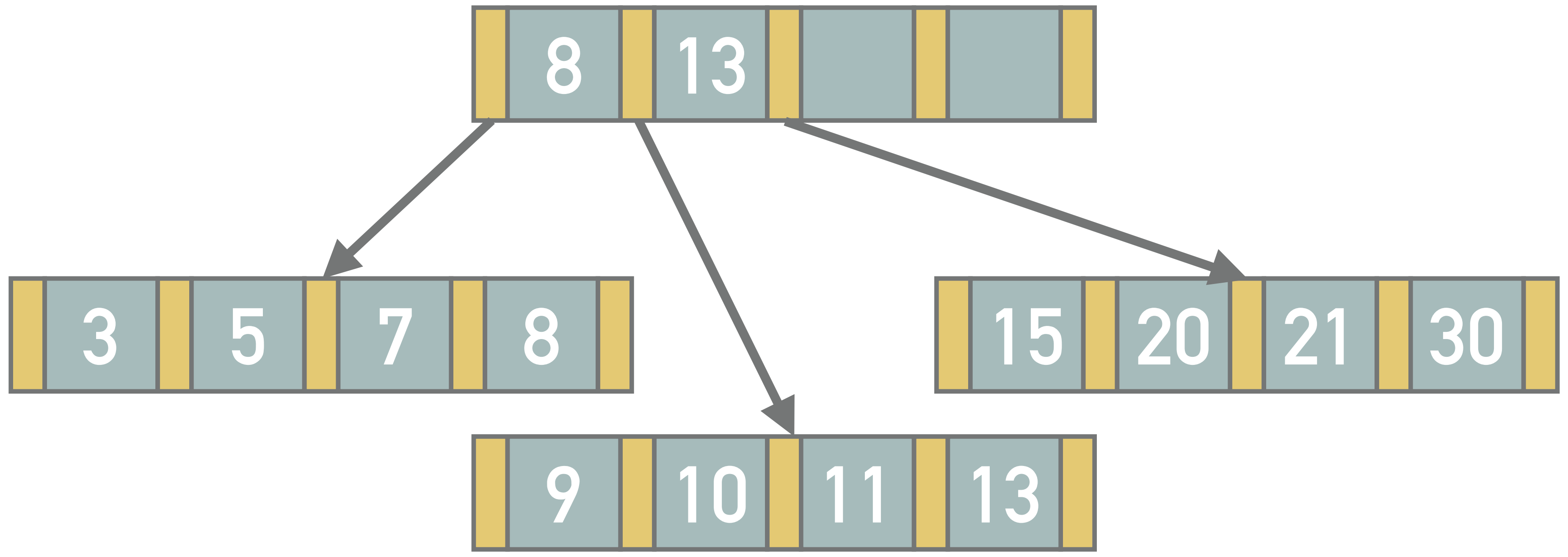


- The record 8 fits into the middle leaf

- The record 9 causes redistribution of the record 8 to the leaf and change of the splitting value from 7 to 8

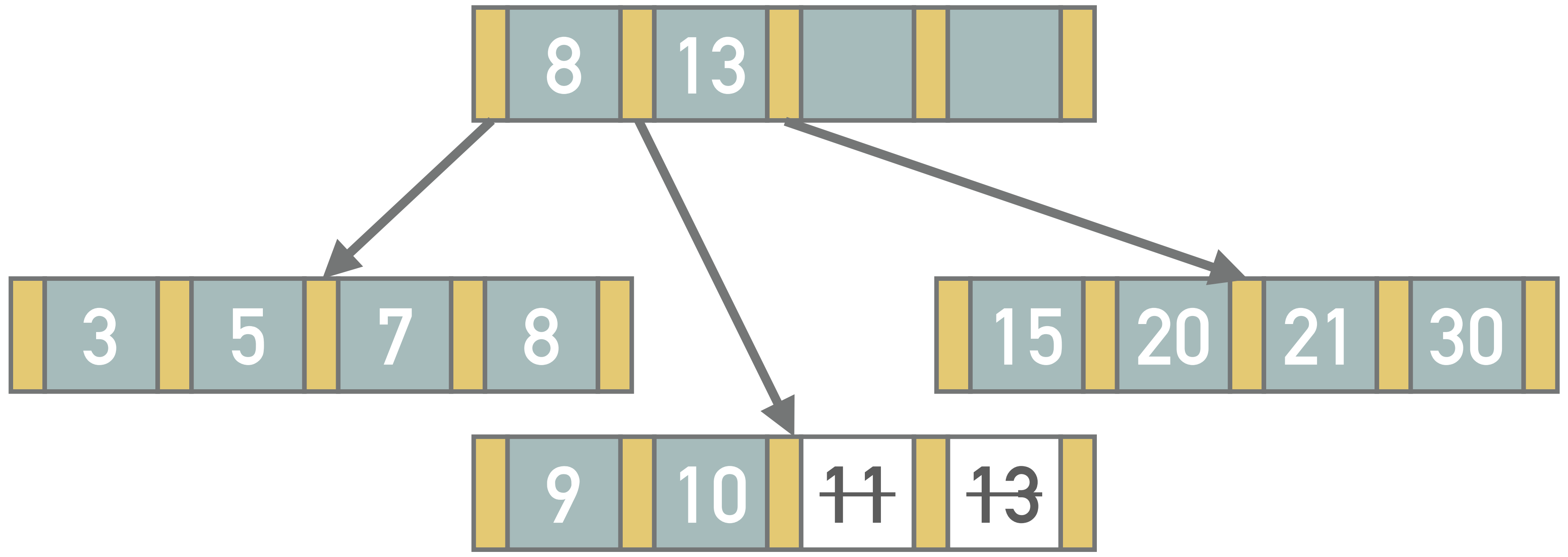


- The record with a key 11 will cause one of two possibilities:
    - **The redistribution of the record with key 15 to the right and modification of the splitting value in the parent from 15 to 13**
    - Split of nodes (3, 5, 7, 8) and (9, 10, 13, 15) into three nodes (3, 5, 7), (8, 9, 10) and (11, 13, 15)
        - The splitting value 8 would be replaced by a pair 7 and 10

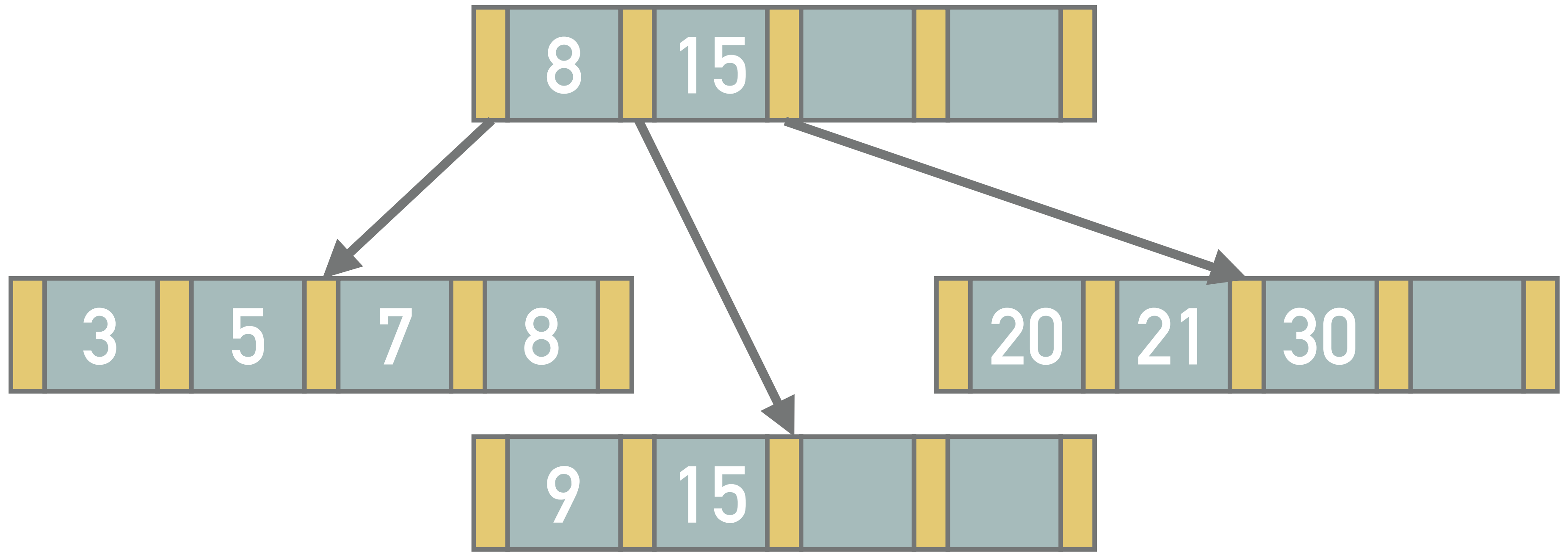


Example 5.16: Delete

- Continue with previous example and delete the records with keys 13, 11, and 10 from redundant B*-tree



- The record with key 13 can be easily deleted from the middle leaf
- The same holds for the record with key 11
- The record with key 10 cannot be deleted directly
  - The number of entries in a node would decrease under the threshold
  - Therefore it is necessary to move there the record with key 15 from the neighboring node
  - The splitting value in the parent changes from 13 to 15



Exercise 5.17

- Continue with previous example and delete records with keys 15, 9, and 8 from redundant B*-tree (see the figure)
- Finally, remove (single) additional key of your choice from the B*-tree
  - Illustrate and comment the removals step by step