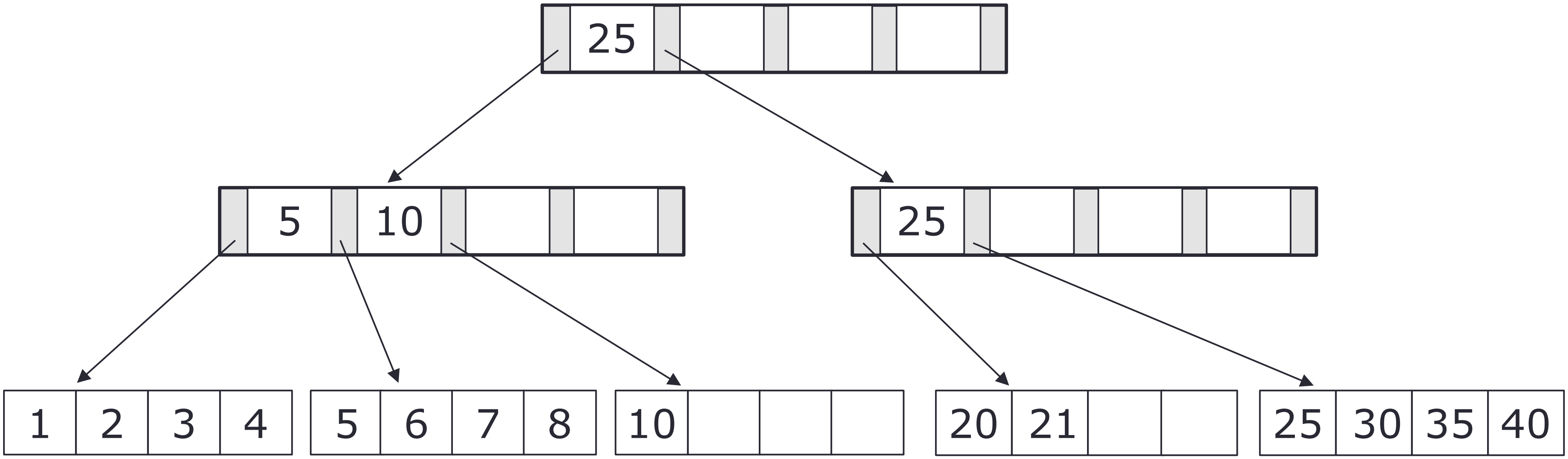
Indexing
B+ tree worksheet

CS 4750
Database Systems

Let’s Try: B+ Tree Constraints

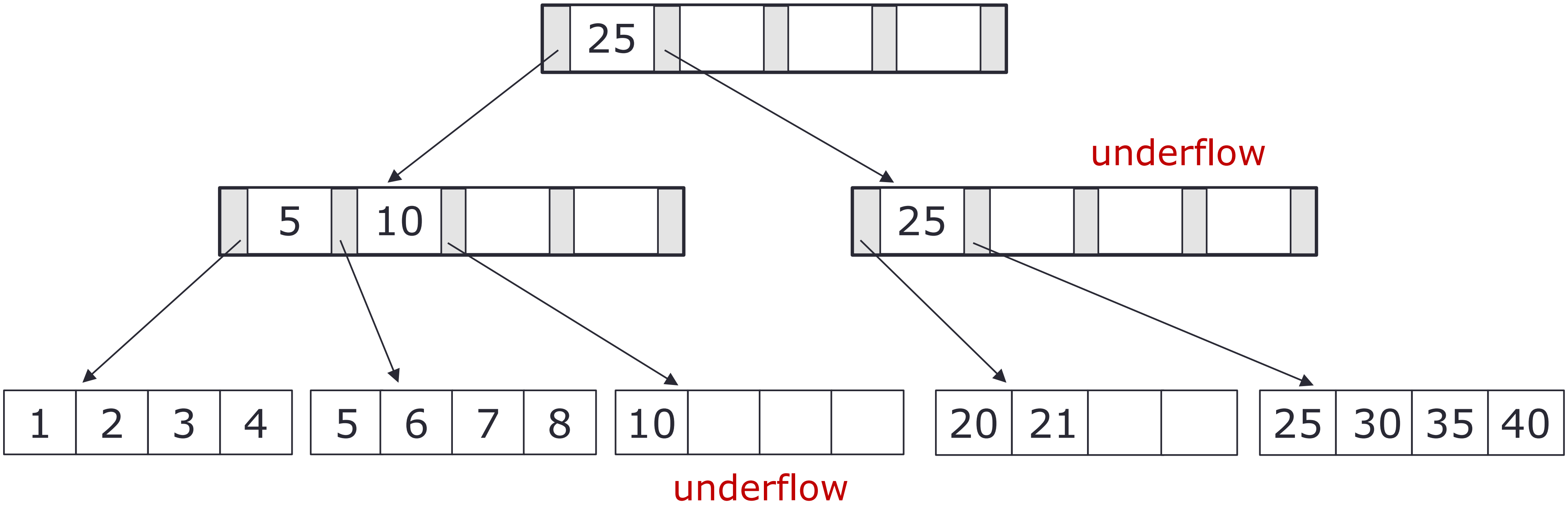
Consider the following tree, identify everything that violates the constraints



[Let’s try]

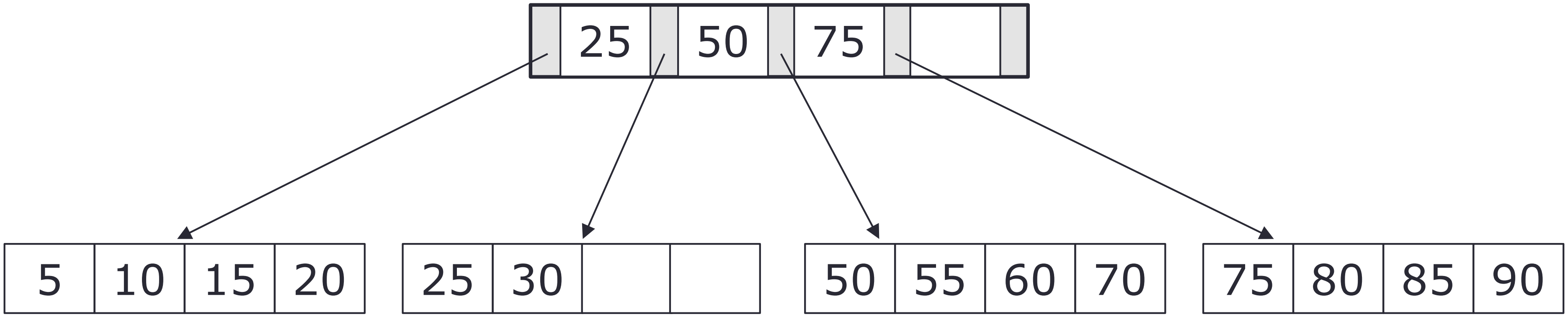
Let’s Try: B+ Tree Constraints

Consider the following tree, identify everything that violates the constraints



Let’s Try: Searching

What would happen if we want to find the key values 45 and 15?



25 50 75

5 10 15 20

25 30

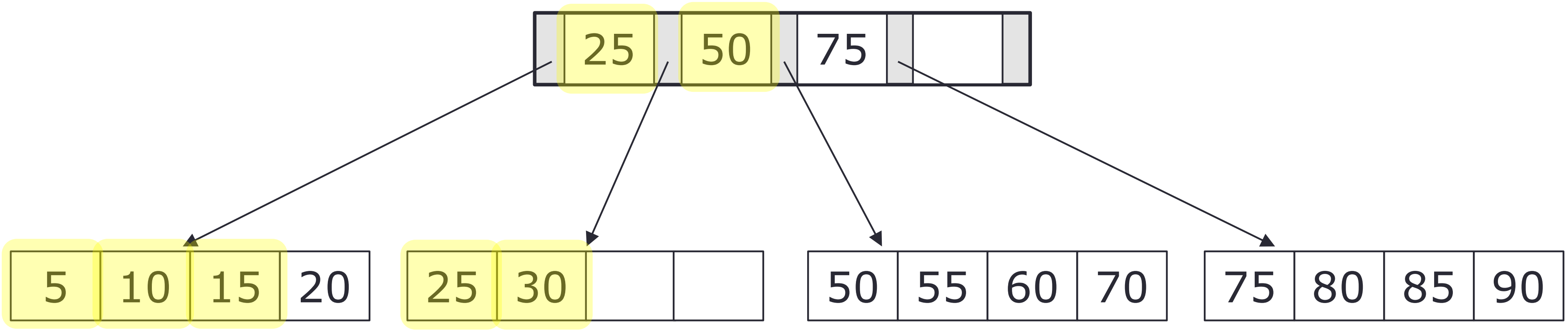
50 55 60 70

75 80 85 90

[Let’s try]

Let’s Try: Searching

What would happen if we want to find the key values 45 and 15?

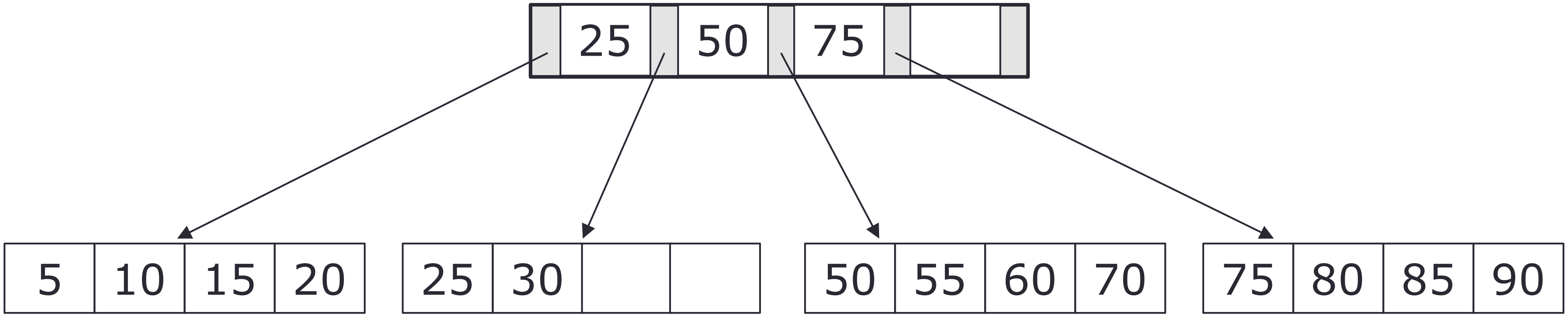


Result:

- For 45, the key value is not found
- For 15, return the position where the pointer located

Let’s Try: Insertion (1)

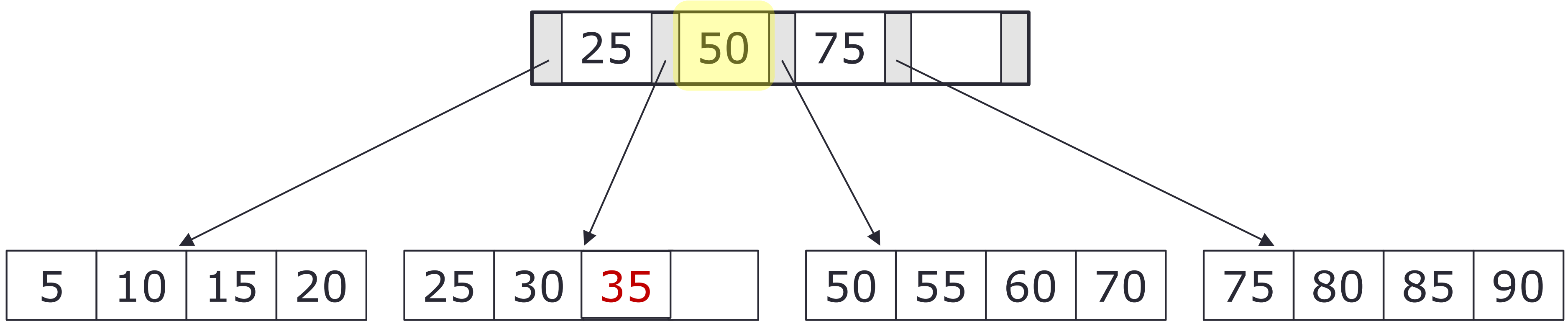
Insert key value (35, val) into the tree



[Let’s try]

Let’s Try: Insertion (1)

Insert key value (35, val) into the tree



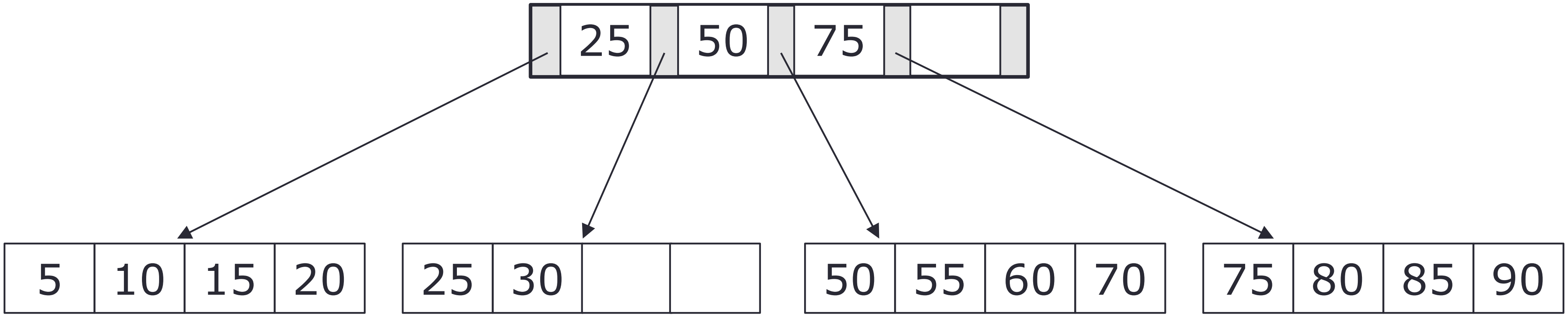
Data page (leaf node) – not full

Index page – not full

Place the record in sorted position in the appropriate leaf page

Let’s Try: Insertion (2)

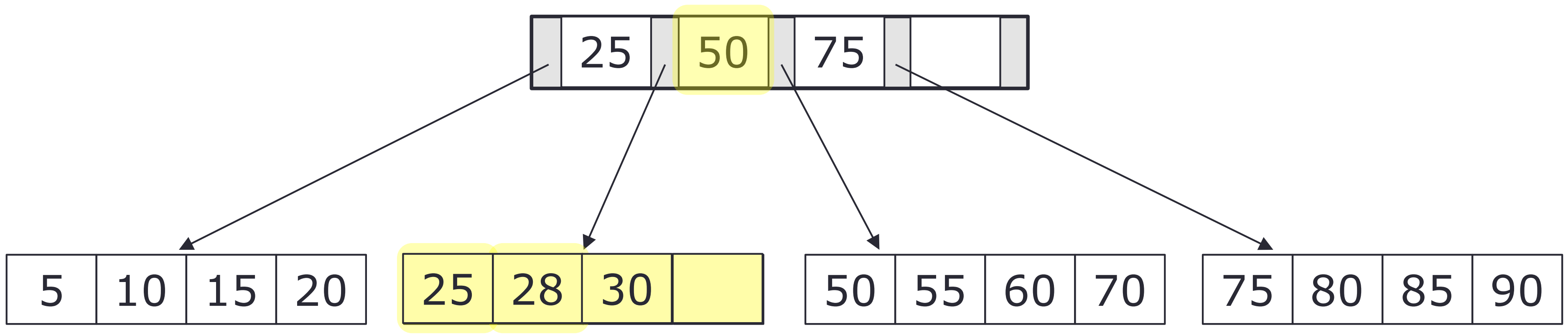
Insert key value (28, val) into the tree



[Let’s try]

Let's Try: Insertion (2)

Insert key value (28, val) into the tree



Preserve the ordering of keys

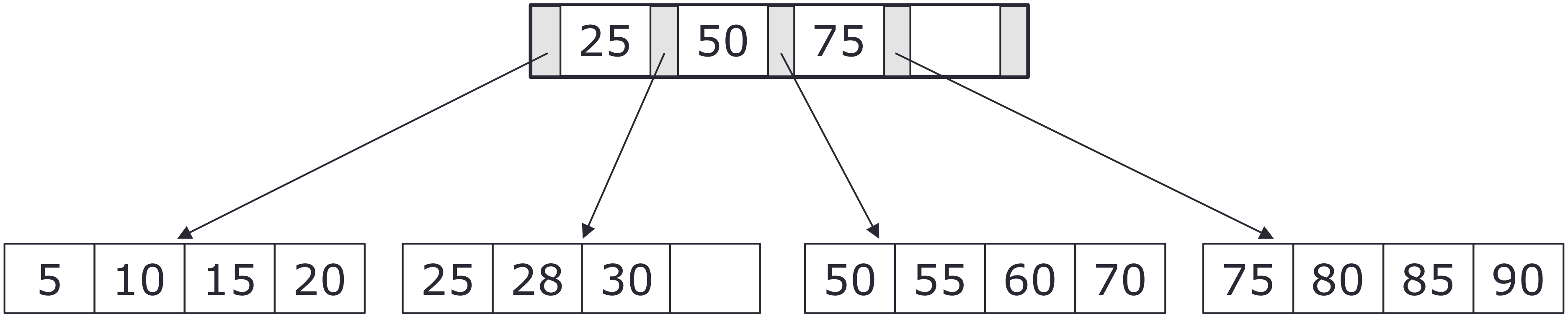
Find the place to insert

Leaf is not full

Insert (28, val) into the leaf node, in sorted order

Let’s Try: Insertion (3)

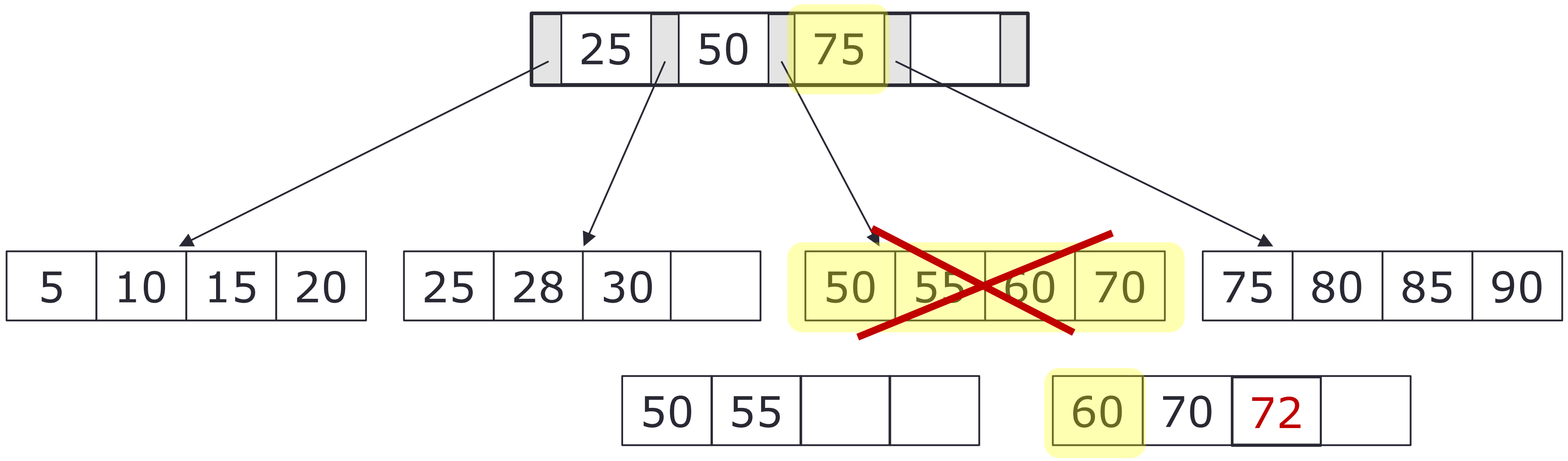
Insert key value (72, val) into the tree



[Let’s try]

Let’s Try: Insertion (3)

Insert key value (72, val) into the tree



Find the place to insert

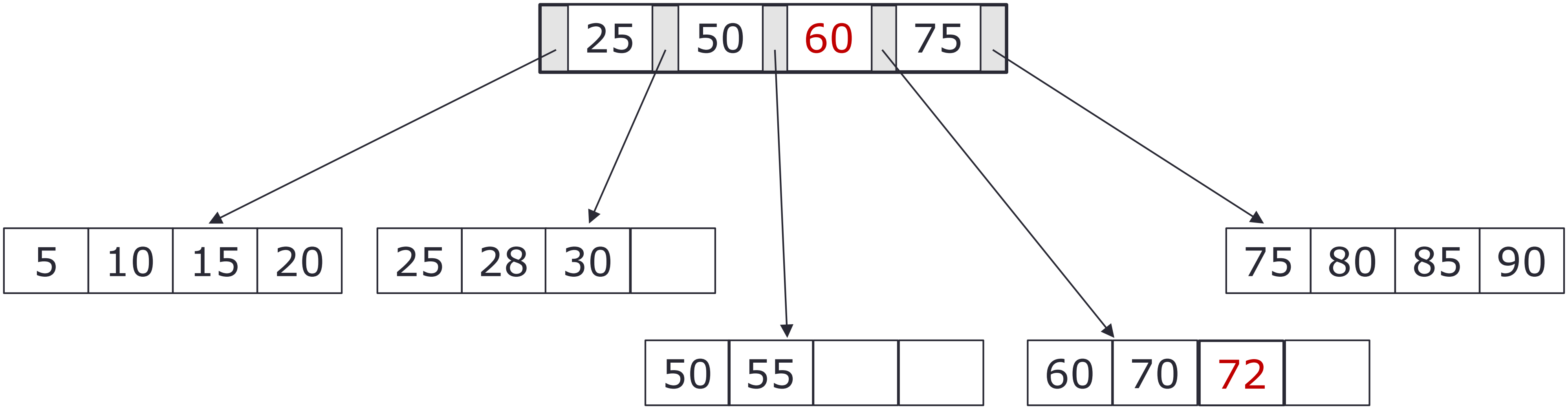
Leaf is full, root is not full

Split leaf, insert (72, val) into the leaf node in sorted order

Update keys in root such that it points to the new node

Let's Try: Insertion (3)

Insert key value (72, val) into the tree



Find the place to insert

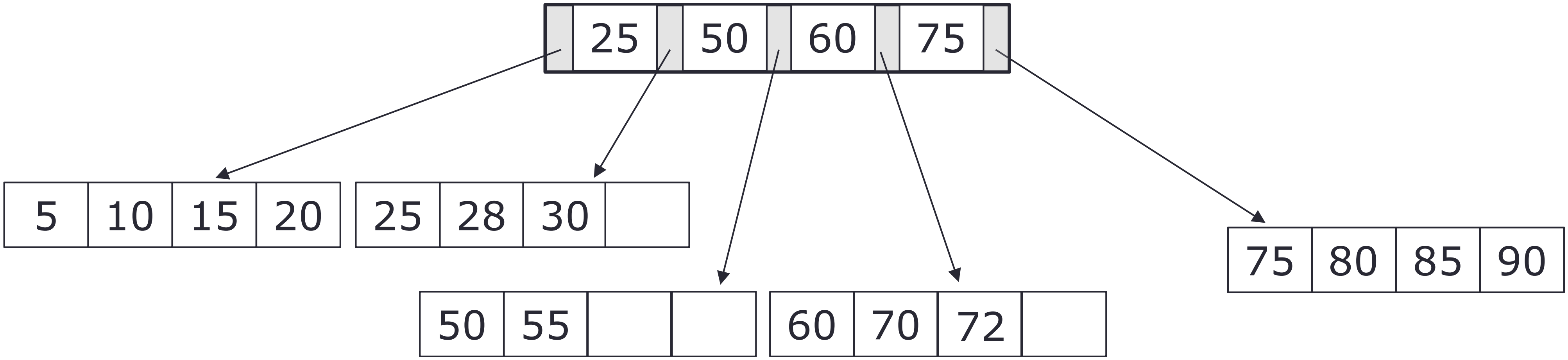
Leaf is full, root is not full

Split leaf, insert (72, val) into the leaf node in sorted order

Update keys in root such that it points to the new node

Let’s Try: Insertion (4)

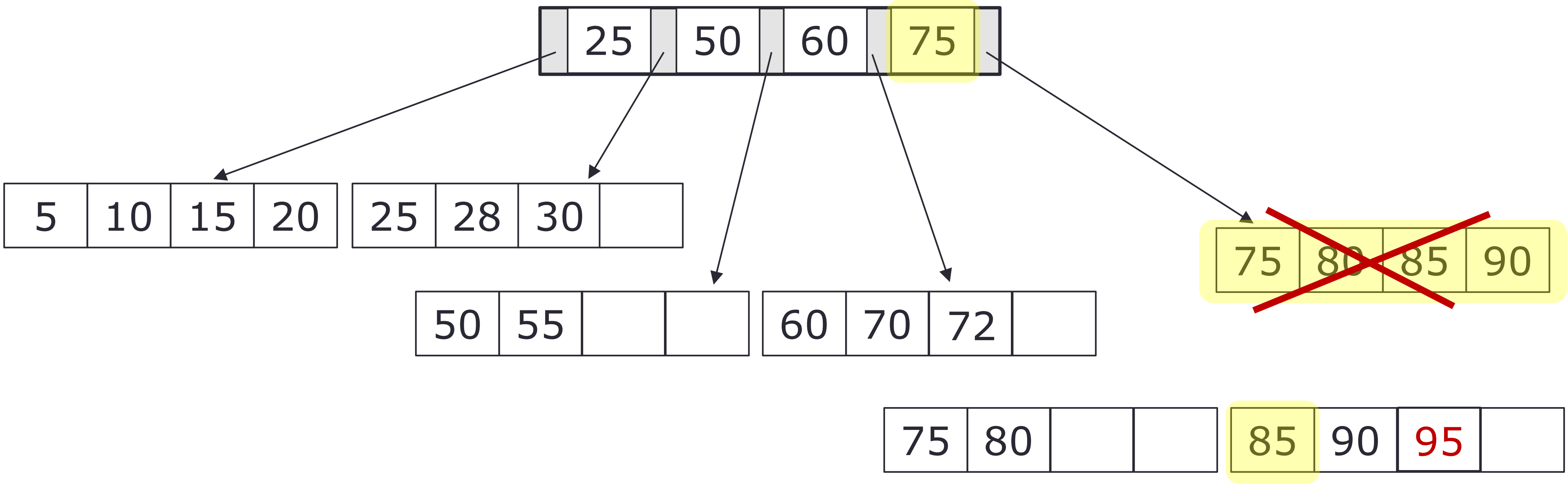
Insert key value (95, val) into the tree



[Let’s try]

Let’s Try: Insertion (4)

Insert key value (95, val) into the tree

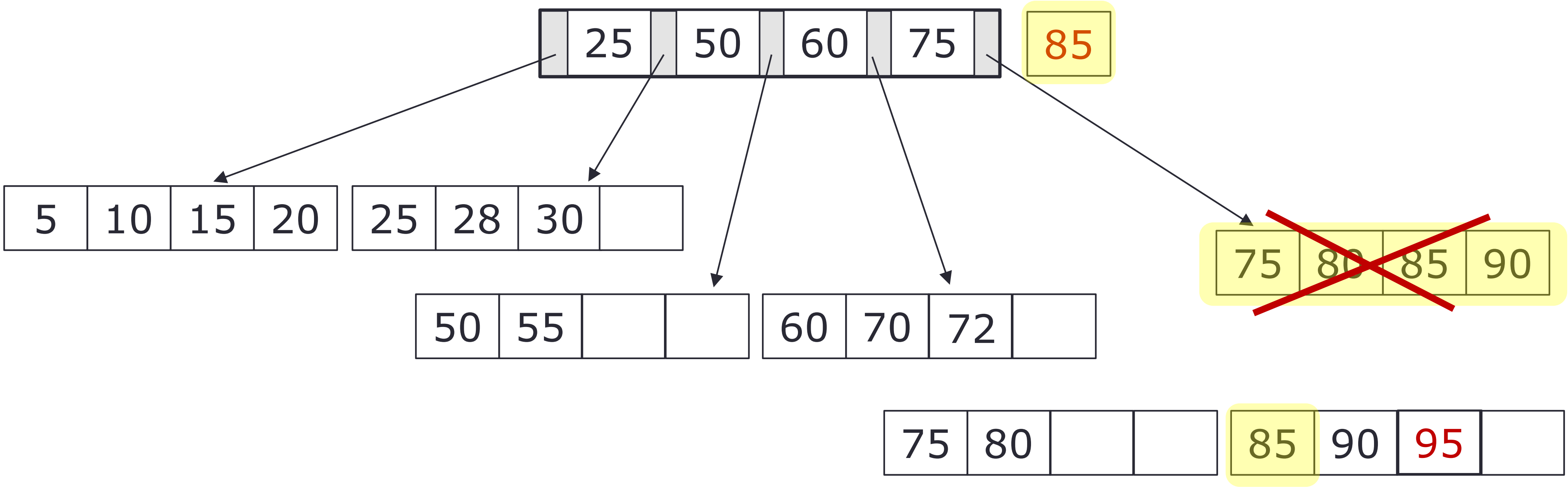


Find the place to insert.

Leaf is full, root is full.

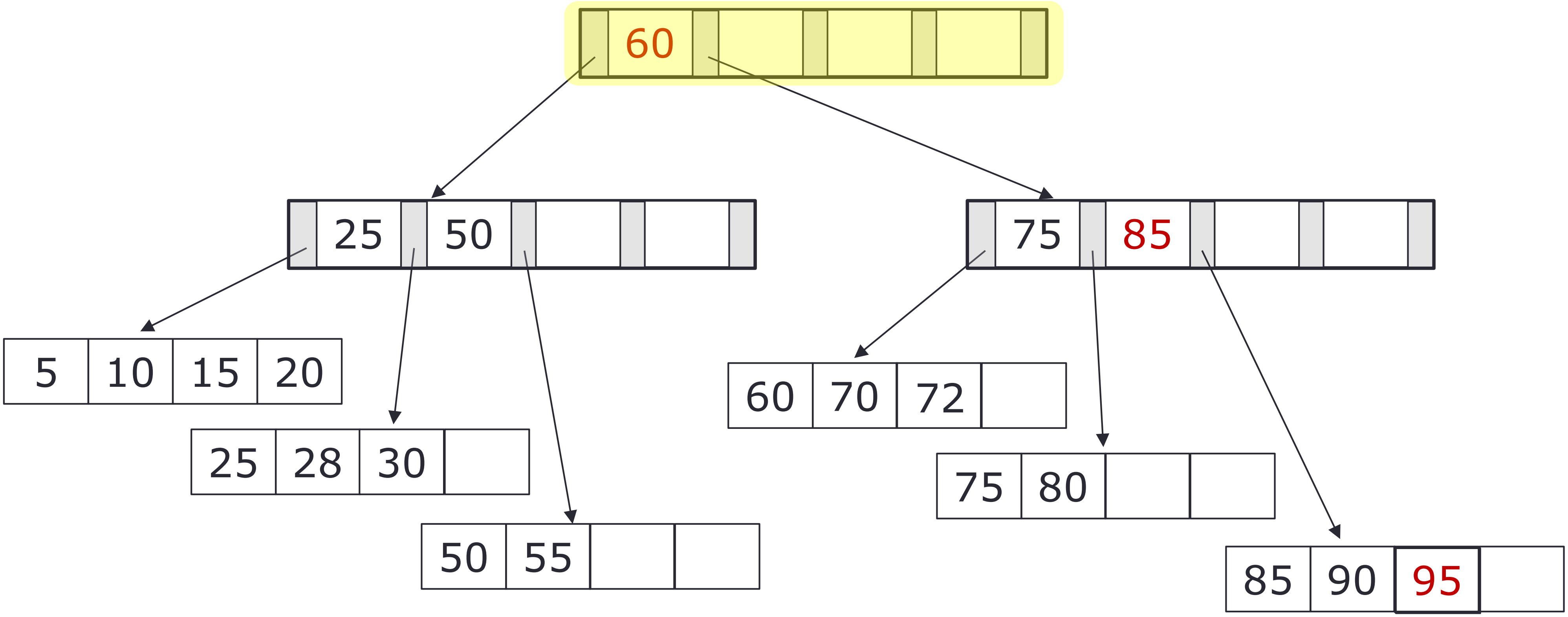
Split leaf, insert (95, val) into the leaf node in sorted order.

Let's Try: Insertion (4)



Since 85 is a new key, need to add 85 to the parent node.
The parent node is full.
There are 5 keys: 25, 50, 60, 75, 85.
Split the parent node and move the middle key (60) up one level.

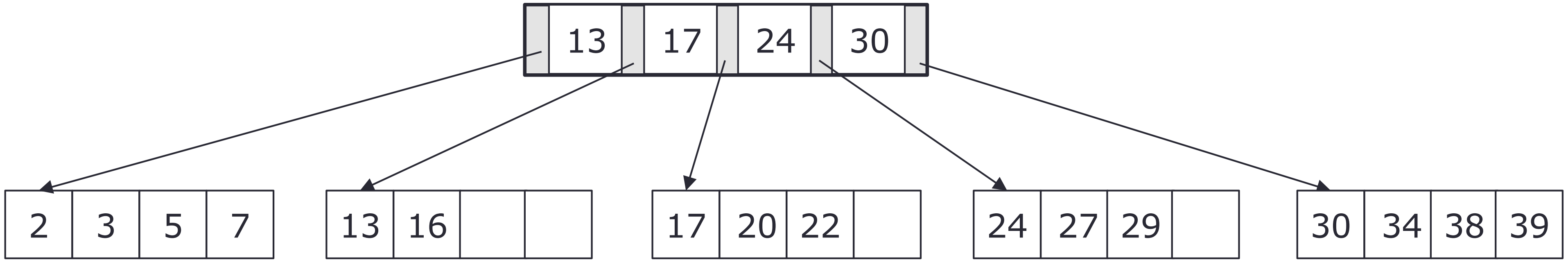
Let’s Try: Insertion (4)



Split the parent node and move the middle key (60) up one level. Update keys as necessary at all levels (root and internal nodes).

Recap 1: Insertion

Insert key value (23, val) into the tree



13 | 17 | 24 | 30

2 | 3 | 5 | 7

13 | 16

17 | 20 | 22

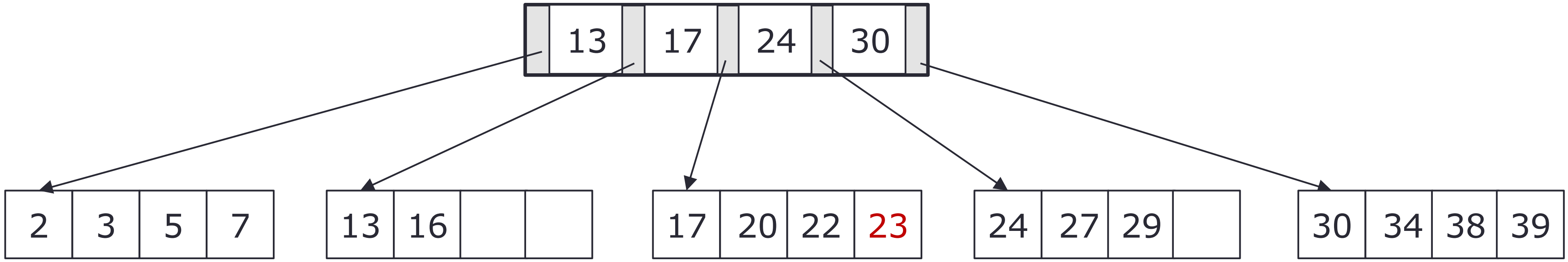
24 | 27 | 29

30 | 34 | 38 | 39

[Let's try]

Recap 1: Insertion

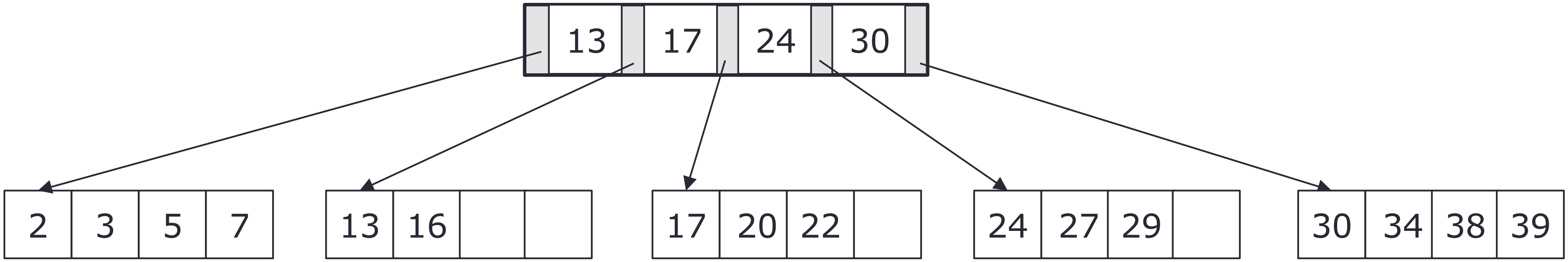
Insert key value (23, val) into the tree



No split needed

Recap 2: Insertion

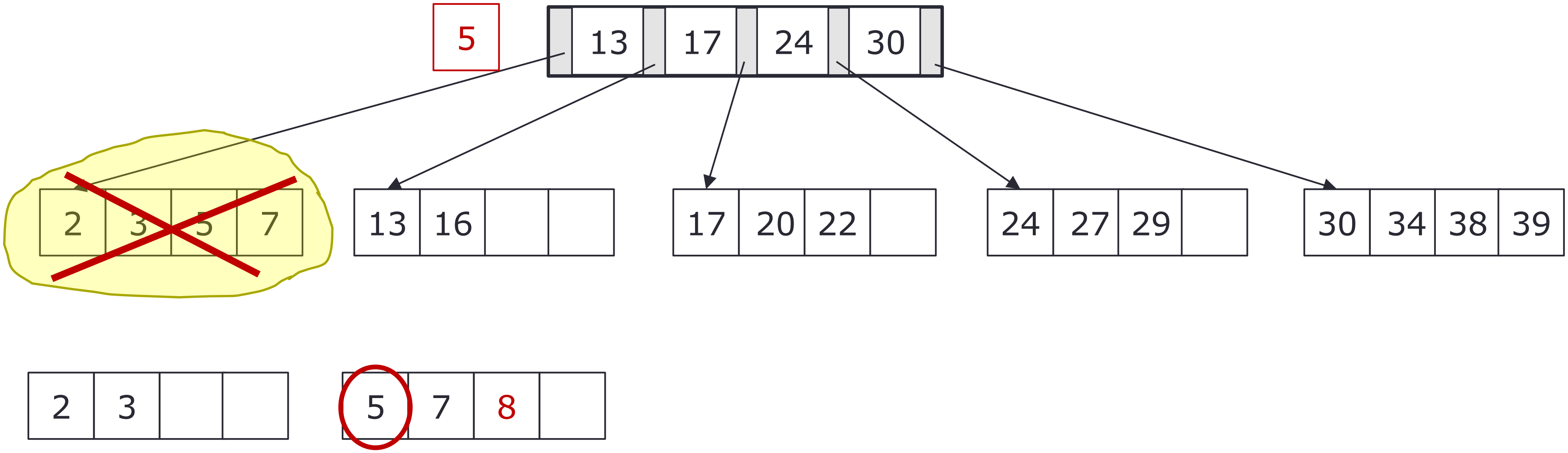
Insert key value (8, val) into the tree



[Let's try]

Recap 2: Insertion

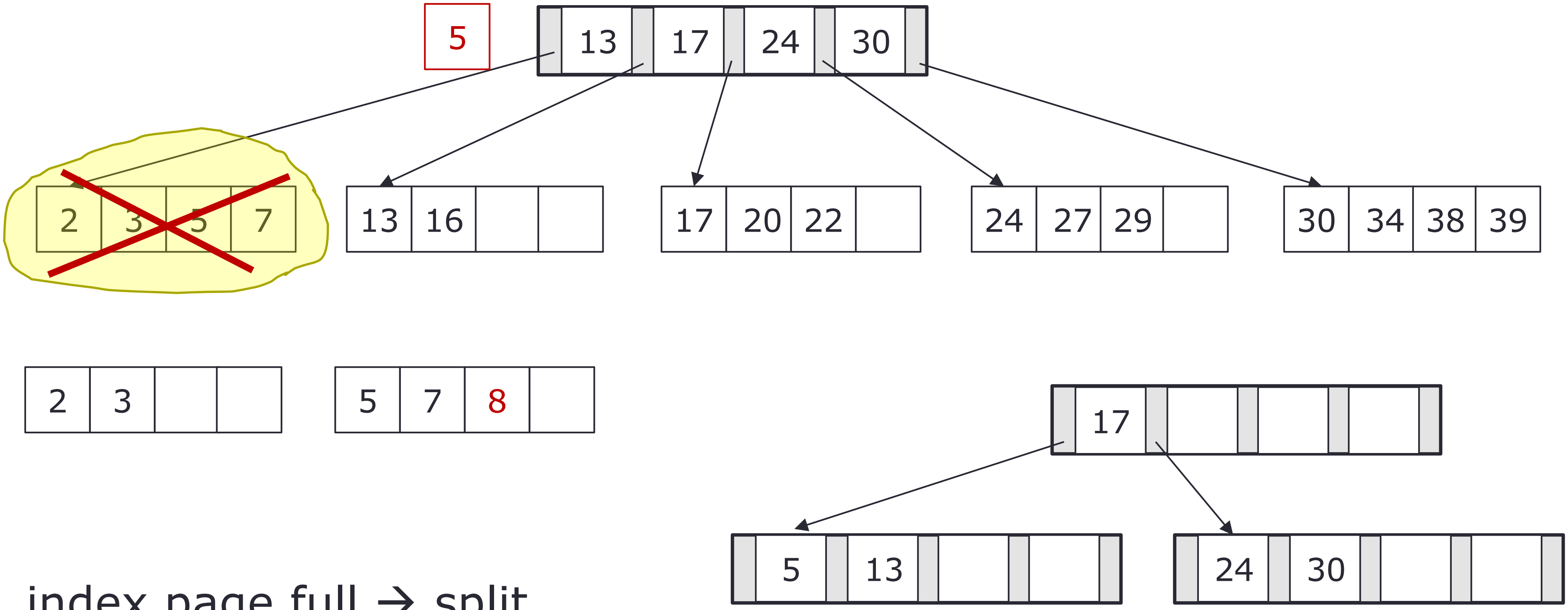
Insert key value (8, val) into the tree



data page full → split
insert 8
copy key (first value) up the level

Recap 2: Insertion

Insert key value (8, val) into the tree

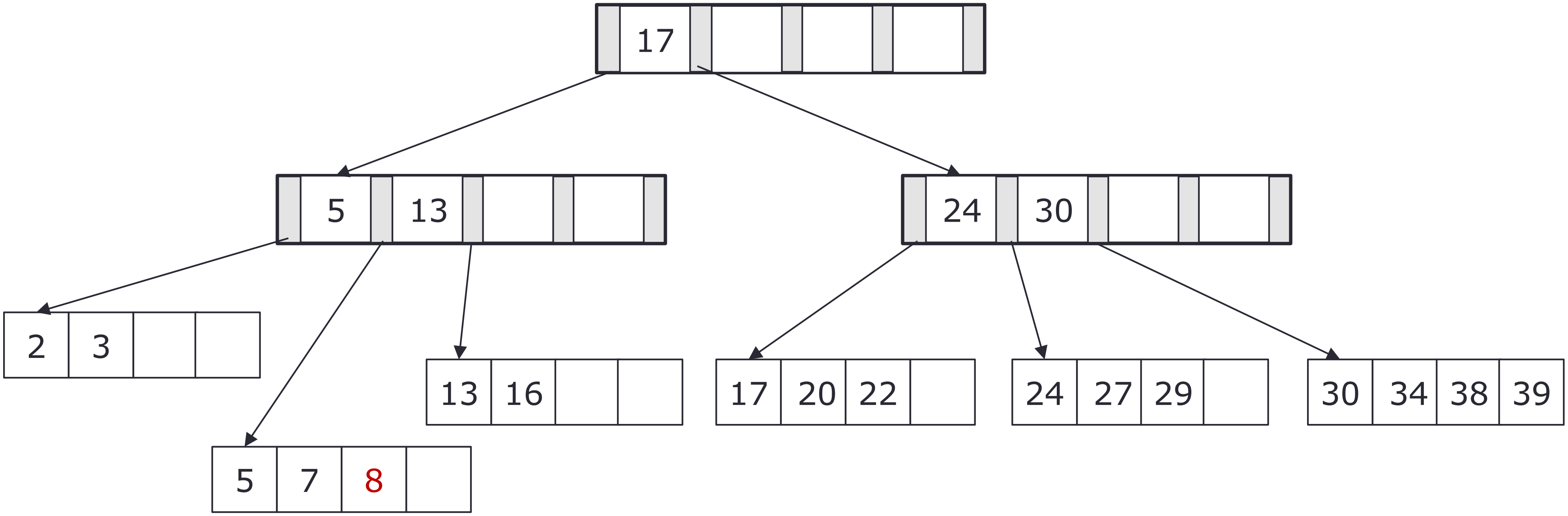


index page full → split

push key (middle value) up the level

Recap 2: Insertion

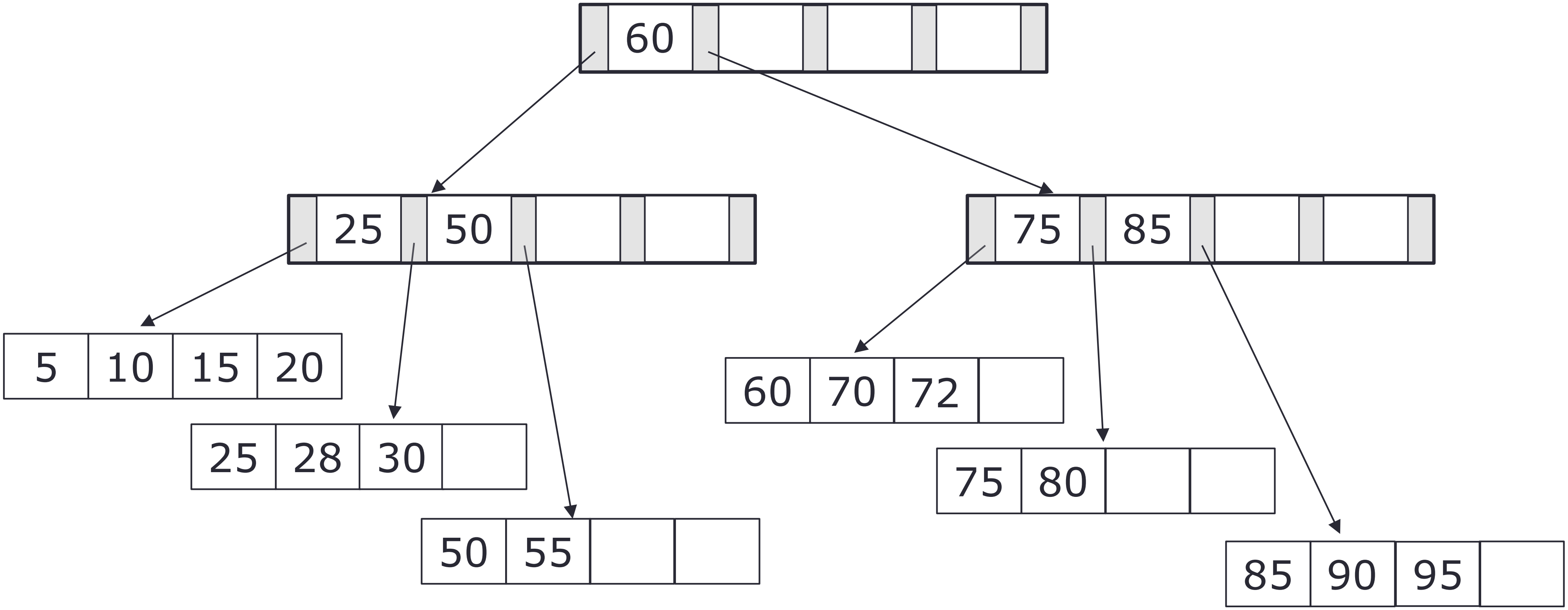
Insert key value (8, val) into the tree



Re-distribute the tree, update the pointers

Let's Try: Deletion (1)

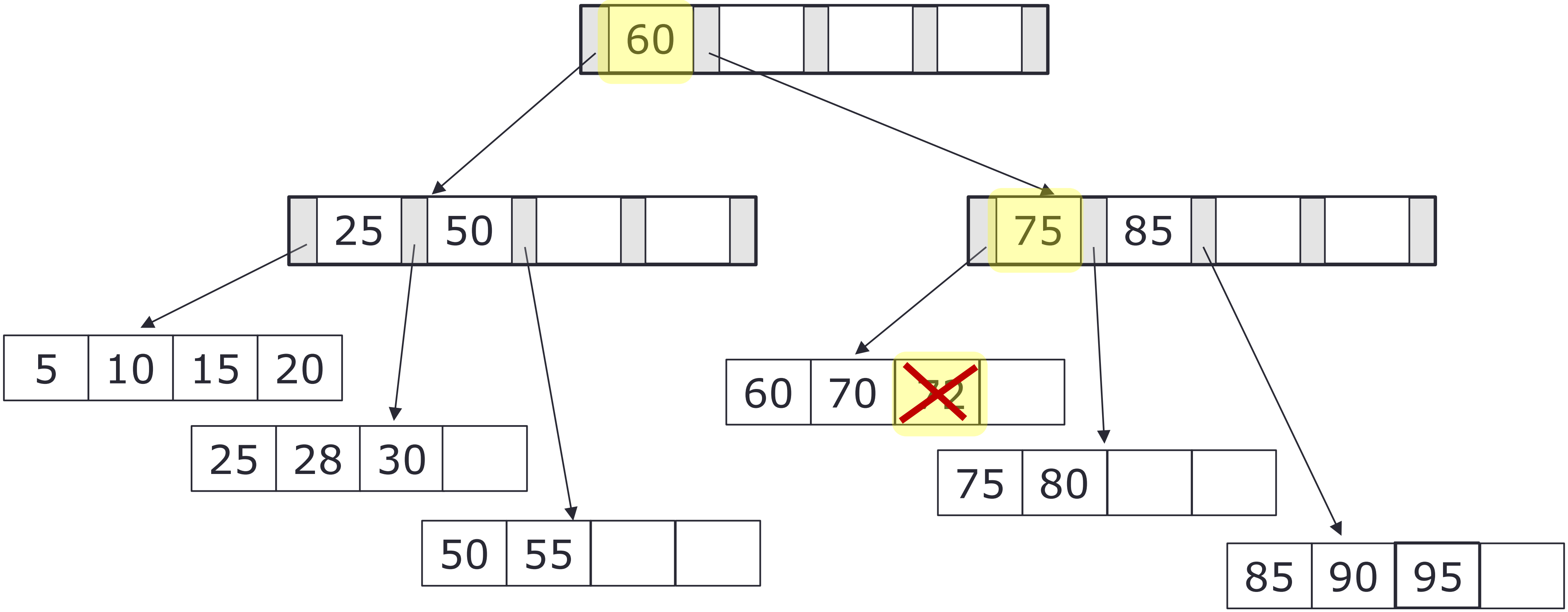
Delete key value 72 from the tree



[Let's try]

Let's Try: Deletion (1)

Delete key value 72 from the tree



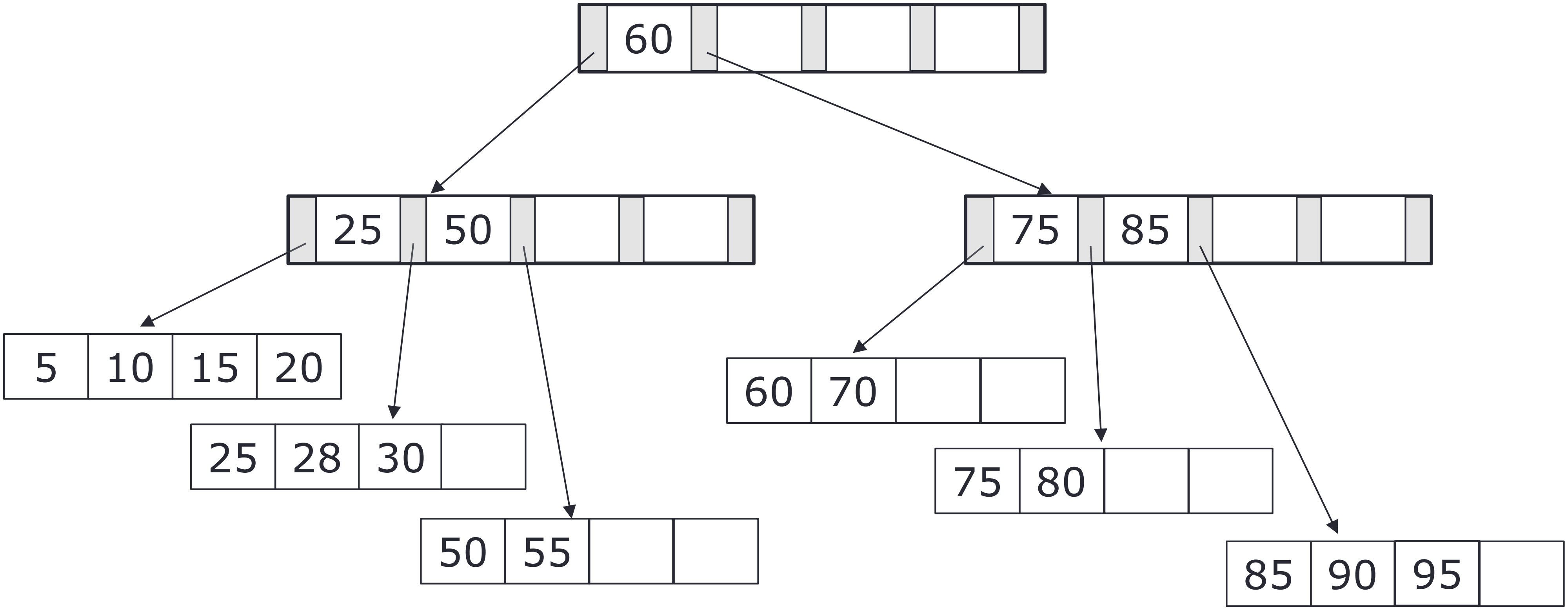
Find the place to delete.

Remove the key from the tree.

If the deletion does not violate B+ tree constraints, no need to update/rebuild the tree.

Let's Try: Deletion (2)

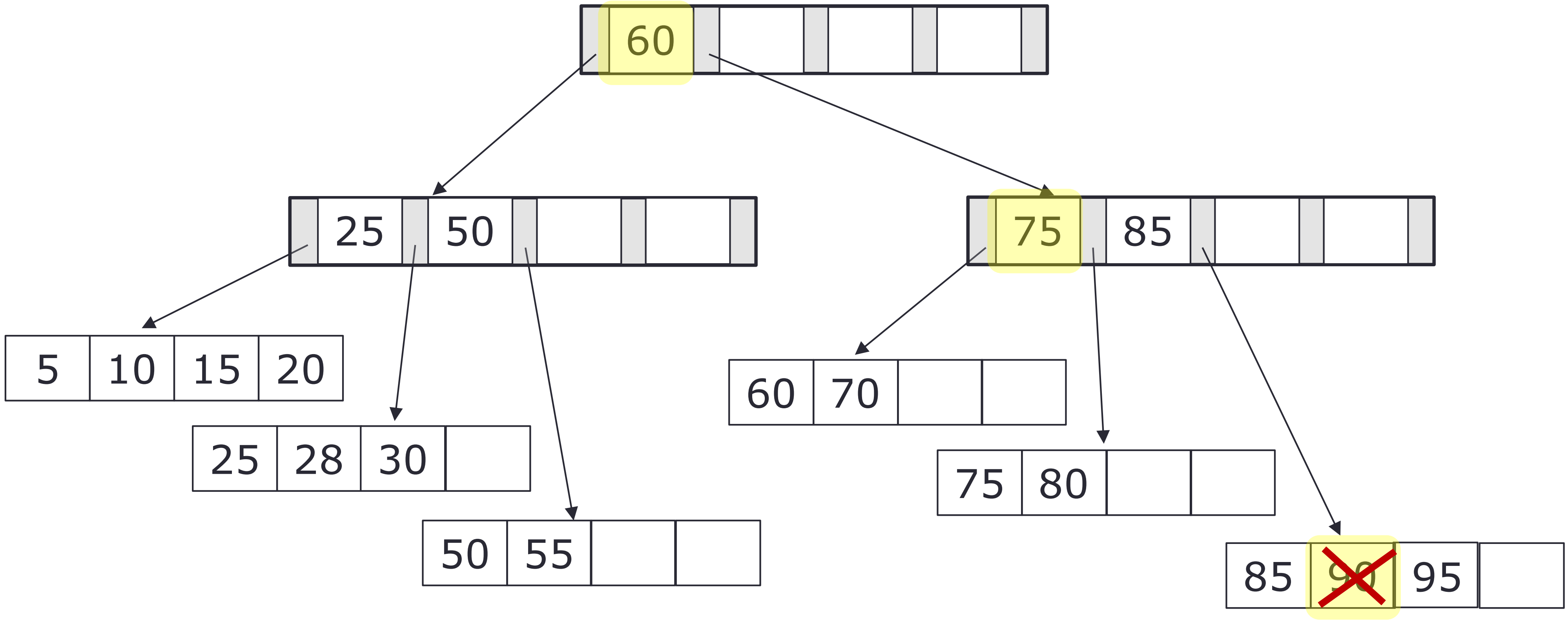
Delete key value 90 from the tree



[Let's try]

Let’s Try: Deletion (2)

Delete key value 90 from the tree

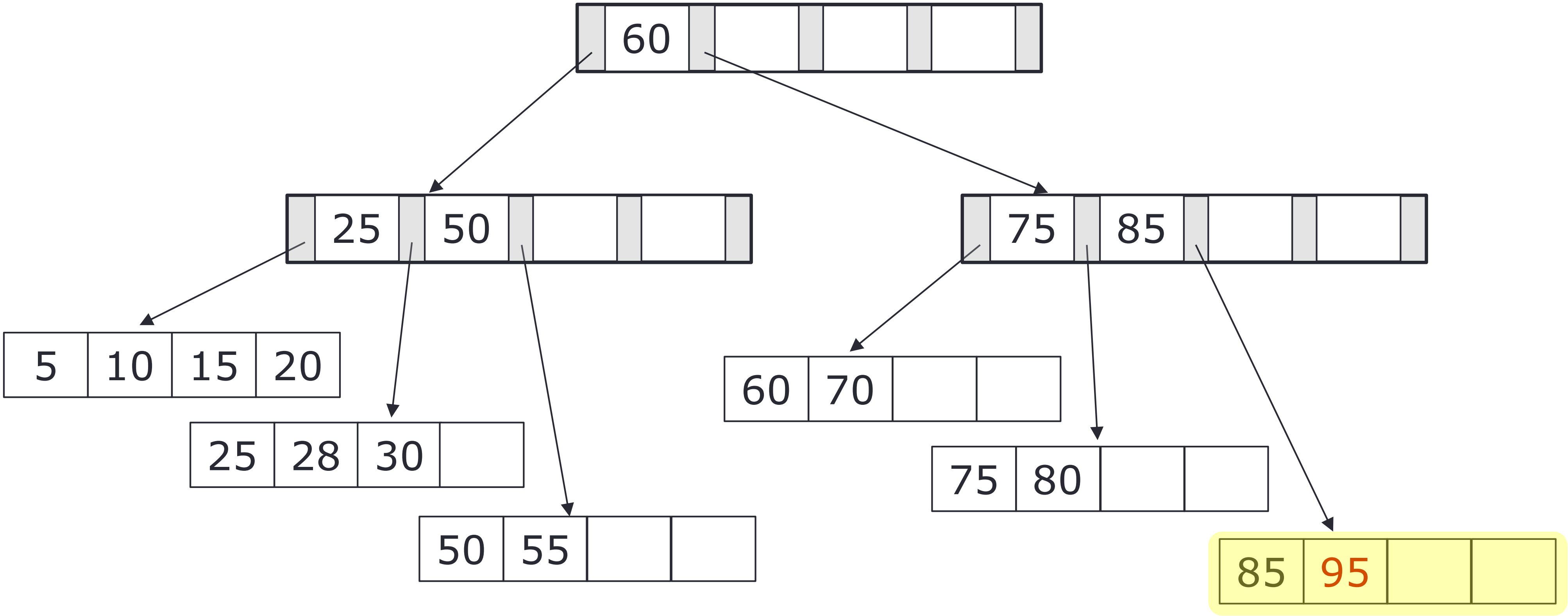


Find the place to delete.

Remove the key from the tree.

Let’s Try: Deletion (2)

Delete key value 90 from the tree

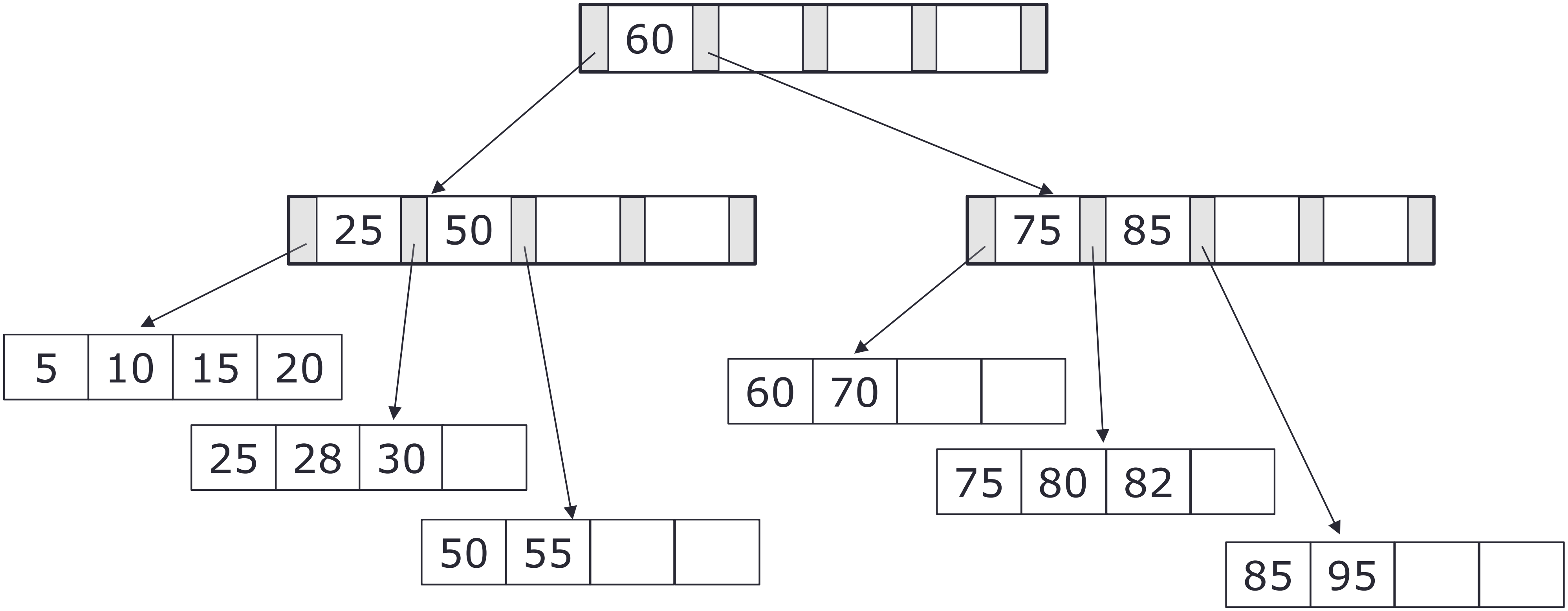


Update the node

If the deletion does not violate B+ tree constraints, no need to update/rebuild the tree.

Let's Try: Deletion (3)

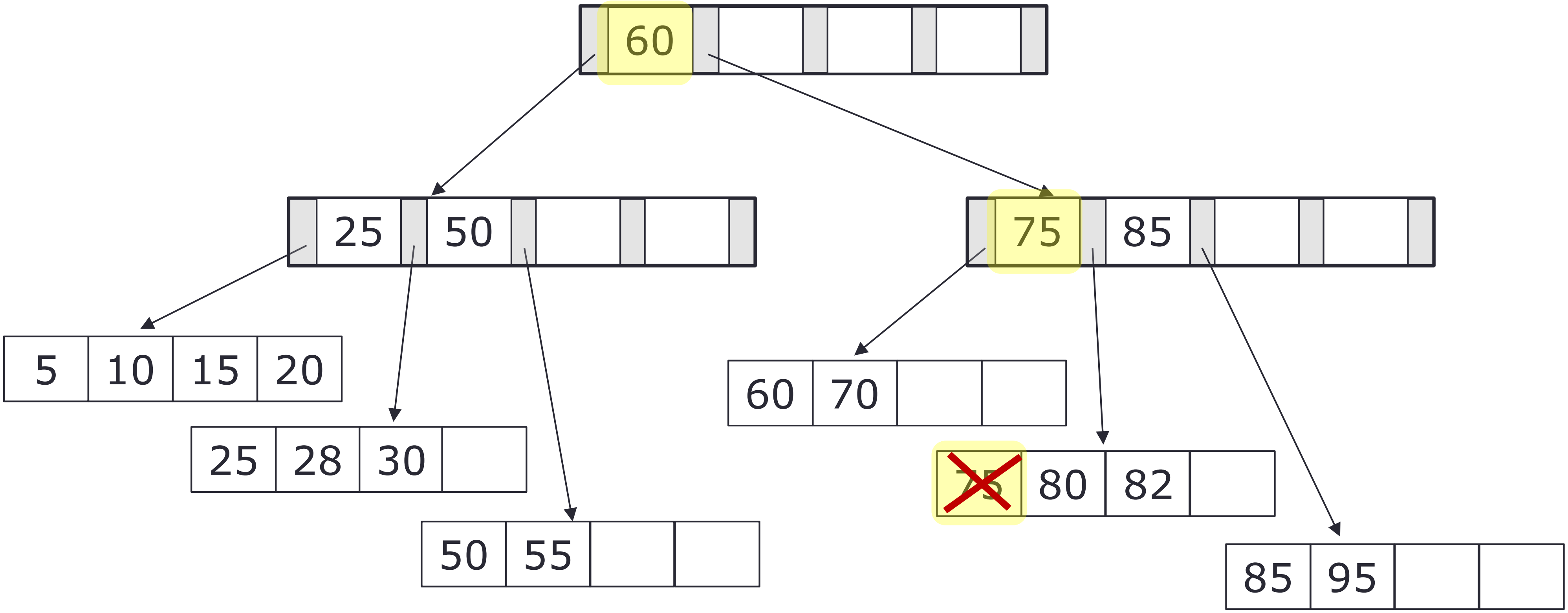
Delete key value 75 from the tree



[Let's try]

Let’s Try: Deletion (3)

Delete key value 75 from the tree

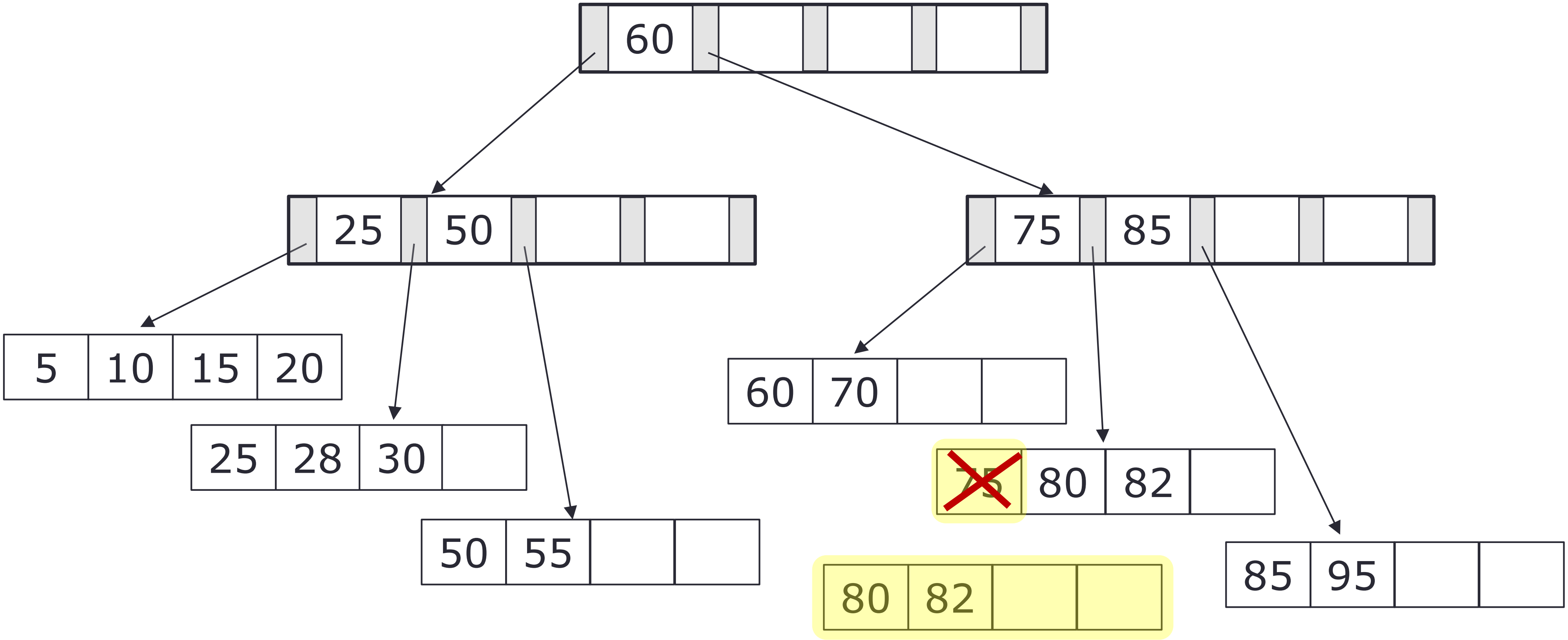


Find the place to delete.

Remove the key from the tree.

Let’s Try: Deletion (3)

Delete key value 75 from the tree

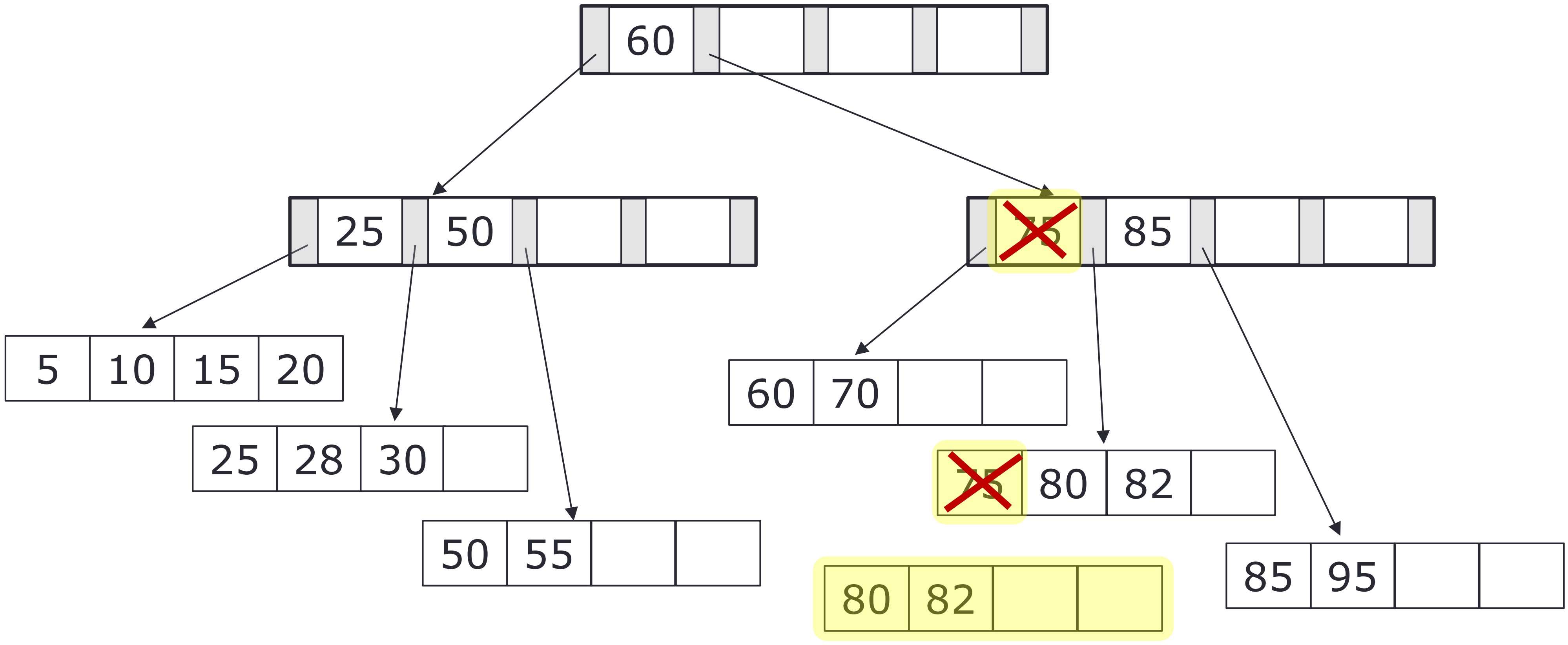


Update the node.

A leaf node with 2 element values does not violate B+ tree property.

Let’s Try: Deletion (3)

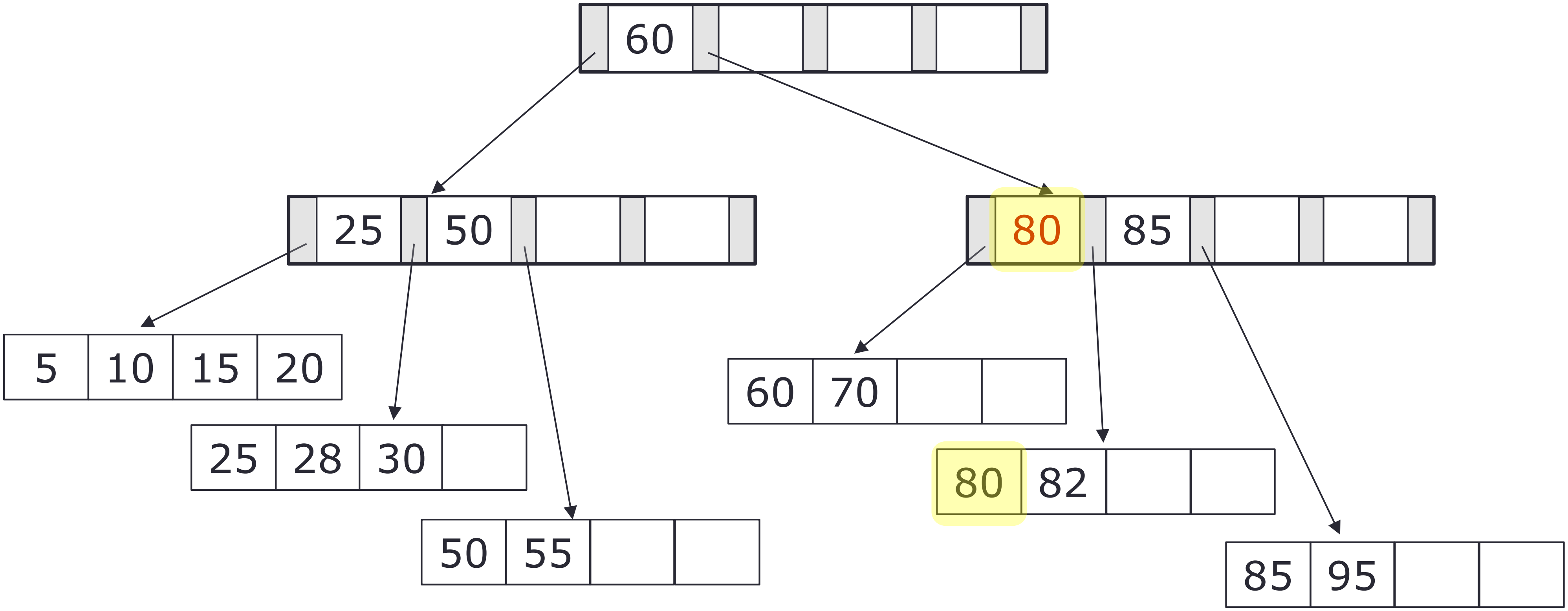
Delete key value 75 from the tree



Since 75 was a key, appearing in the internal node (~index page), need to update the internal node with a new key (80)

Let’s Try: Deletion (3)

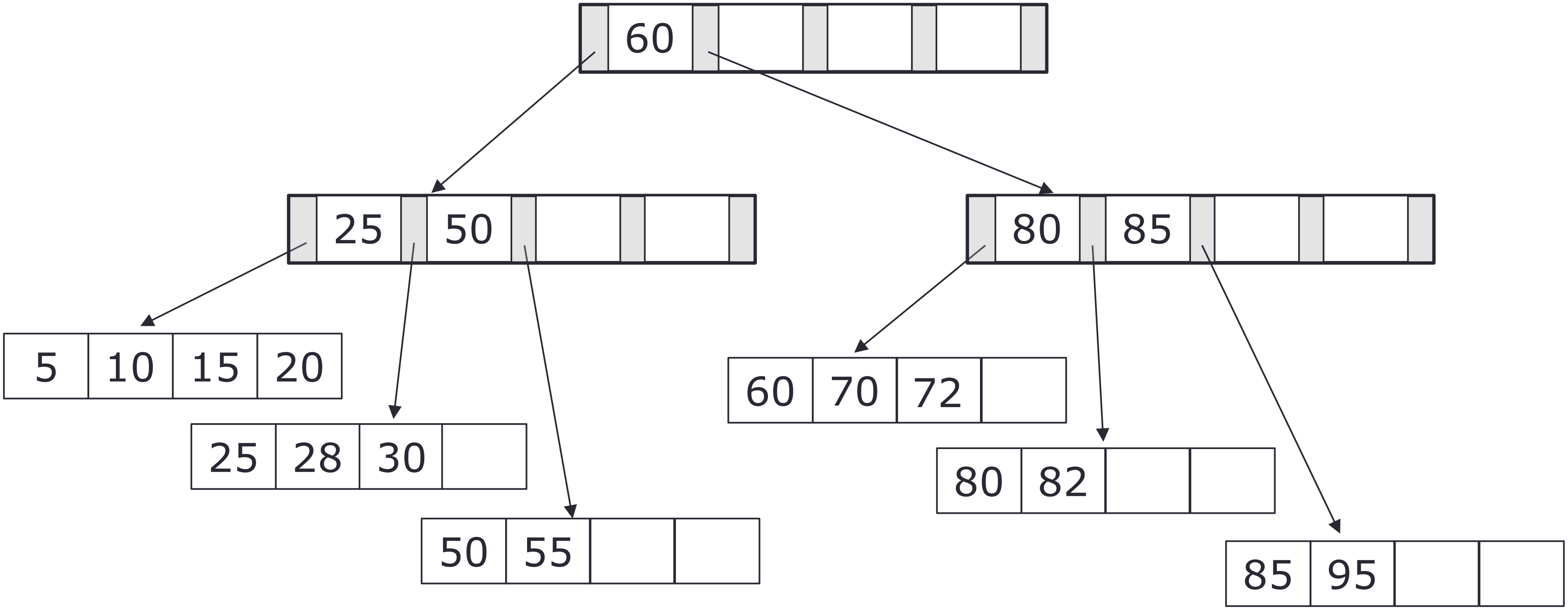
Delete key value 75 from the tree



Since 75 was a key, appearing in the internal node (~index page), need to update the internal node with a new key (80)

Let’s Try: Deletion (4)

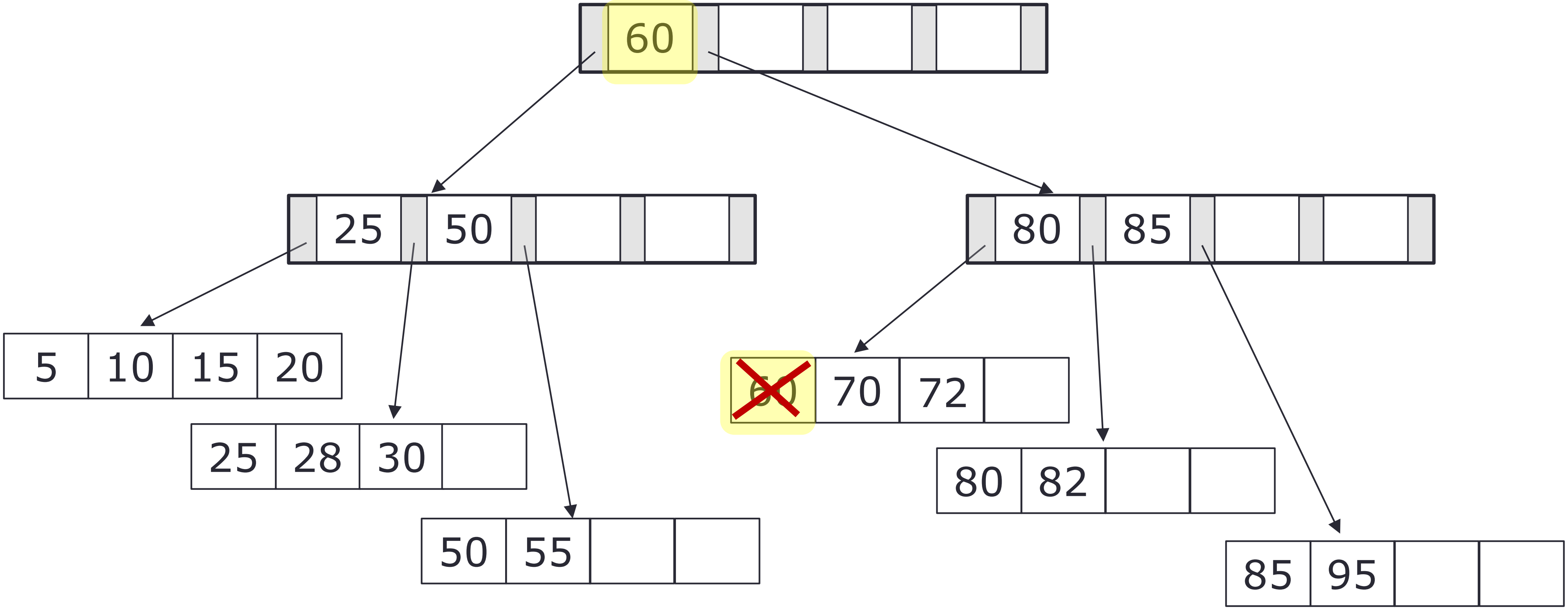
Delete key value 60 from the tree



[Let’s try]

Let’s Try: Deletion (4)

Delete key value 60 from the tree

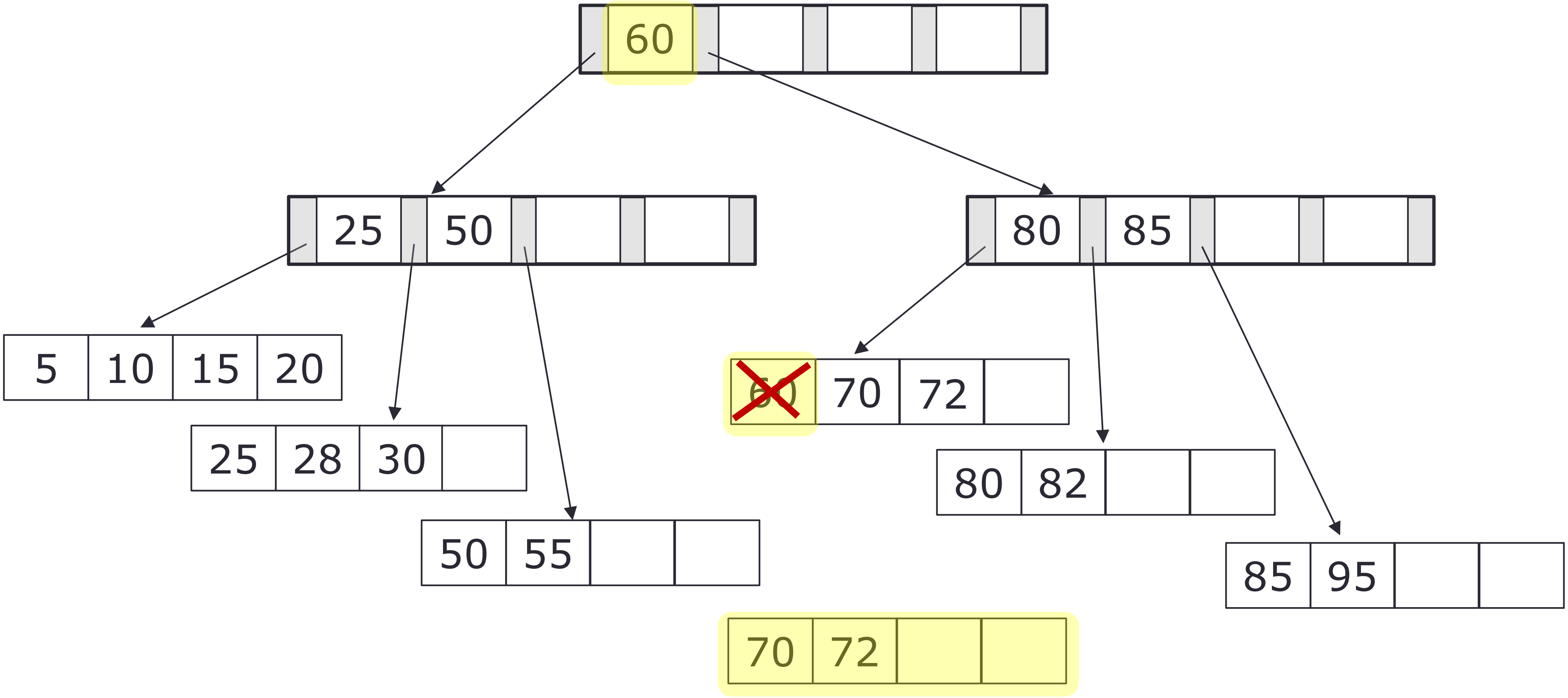


Find the place to delete.

Remove the key from the tree.

Let’s Try: Deletion (4)

Delete key value 60 from the tree

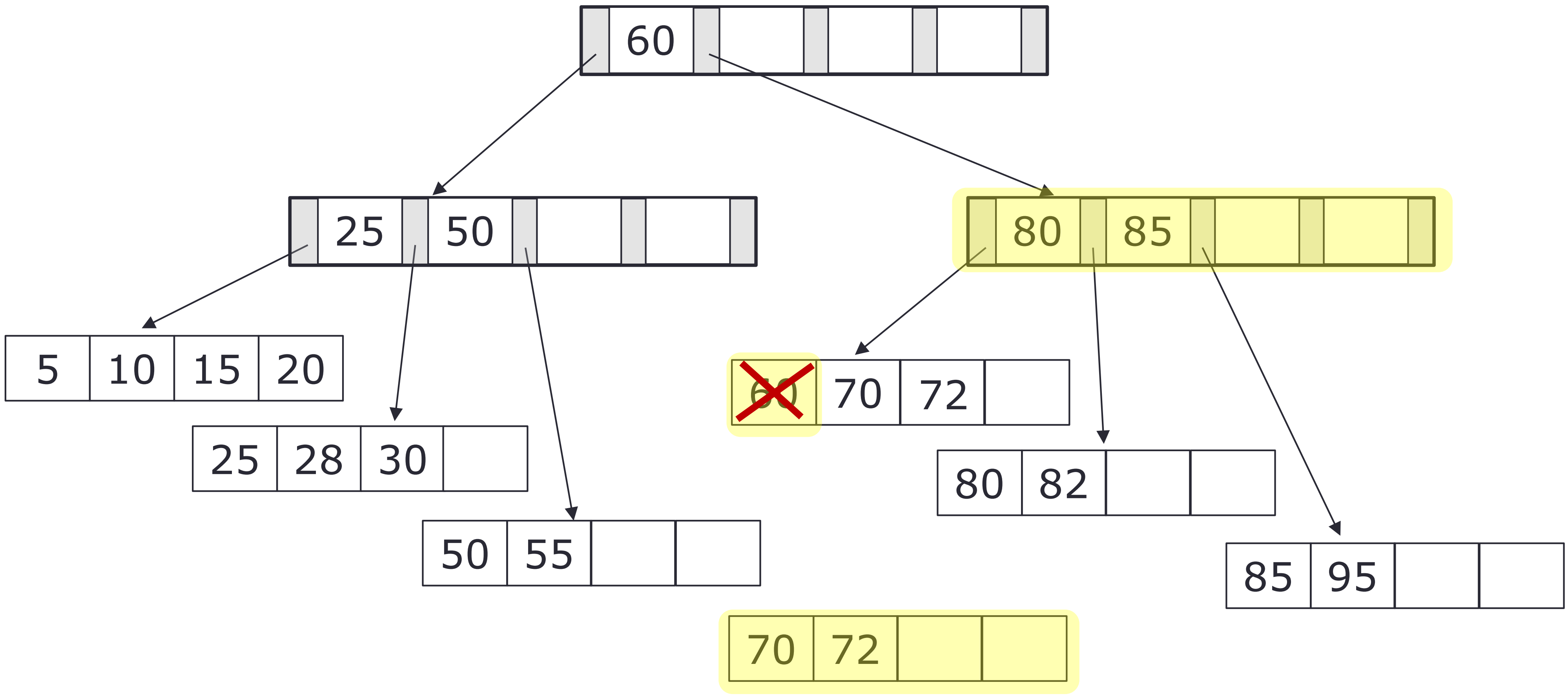


Update the node.

A leaf node with 2 element values does not violate B+ tree property.

Let’s Try: Deletion (4)

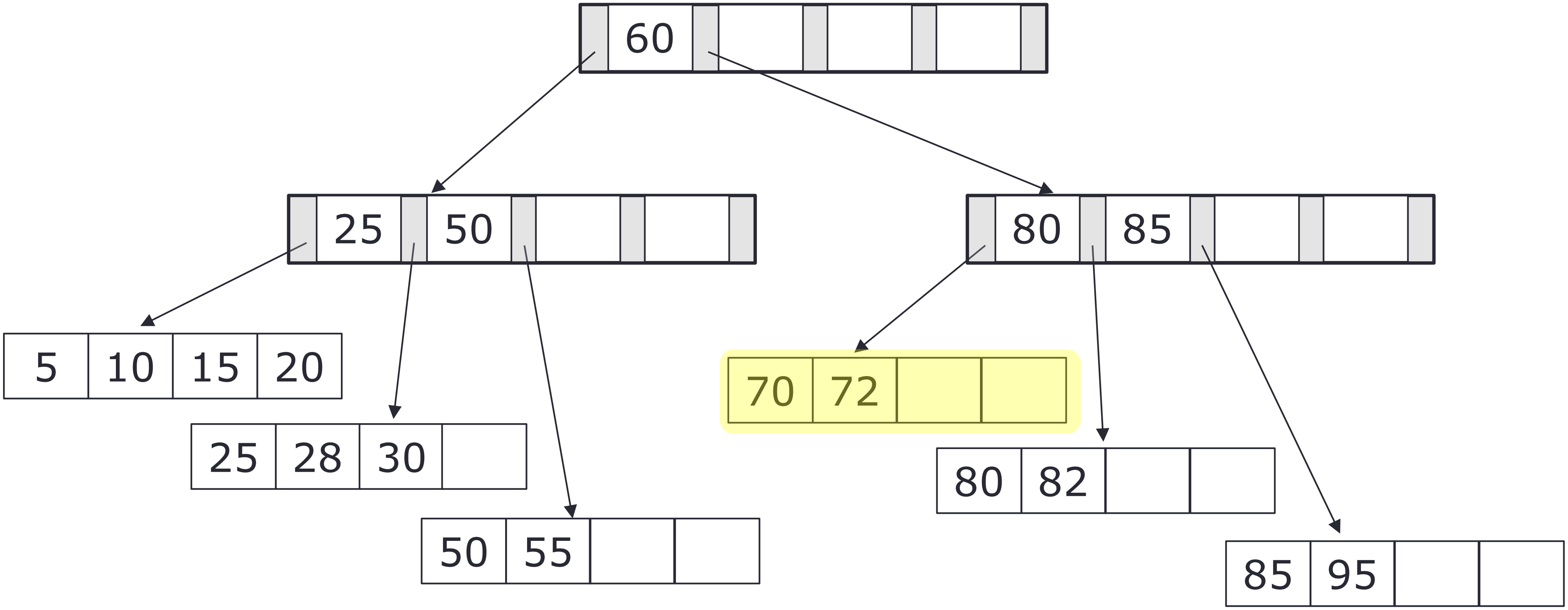
Delete key value 60 from the tree



Since 60 does not appear as a key in its parent (internal node), no need to update the index page (internal node or root node)

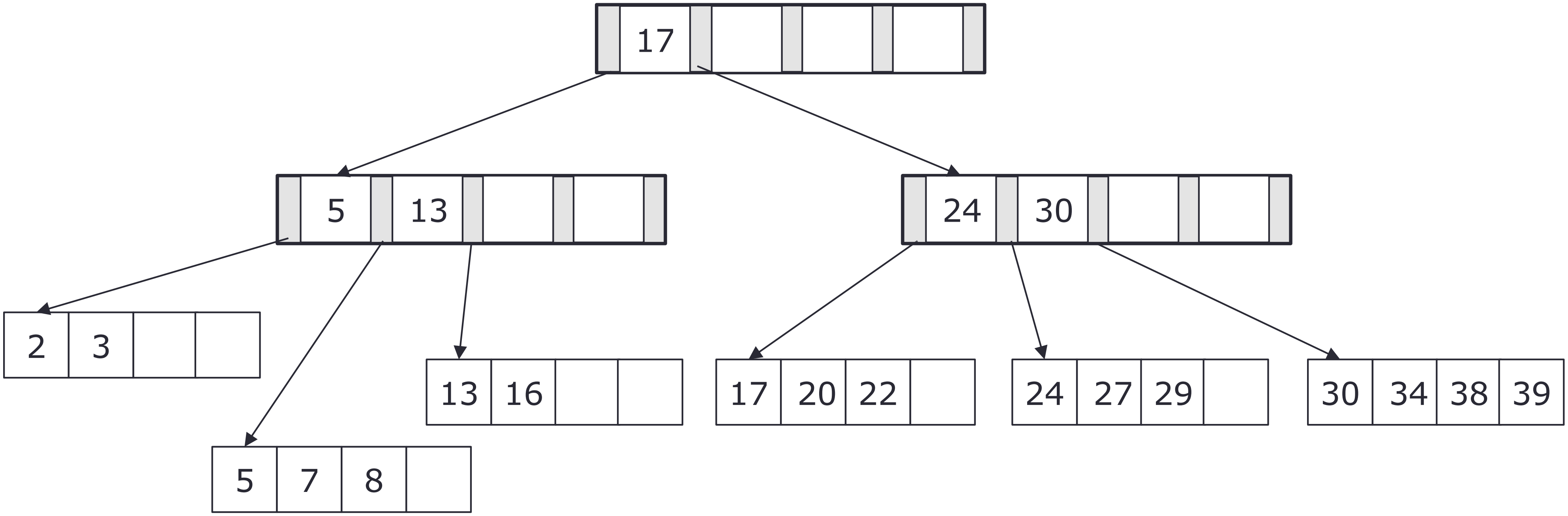
Let’s Try: Deletion (4)

Delete key value 60 from the tree



Recap 3: Deletion

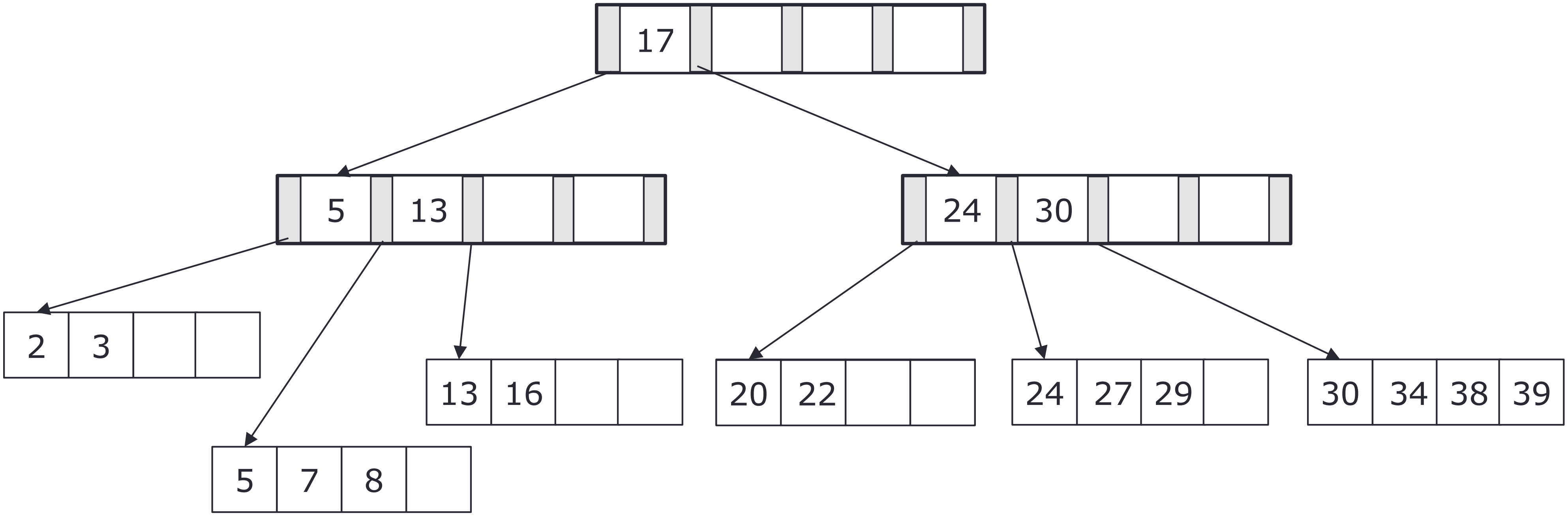
Delete key value 17 from the tree



[Let's try]

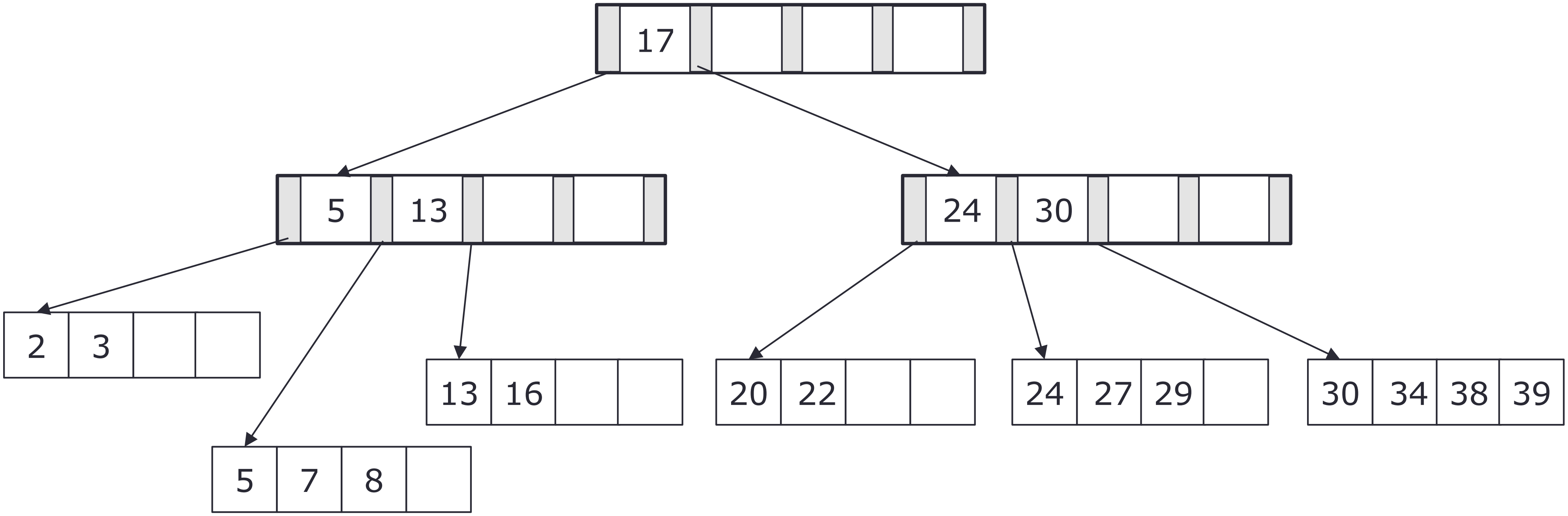
Recap 3: Deletion

Delete key value 17 from the tree



Recap 4: Deletion

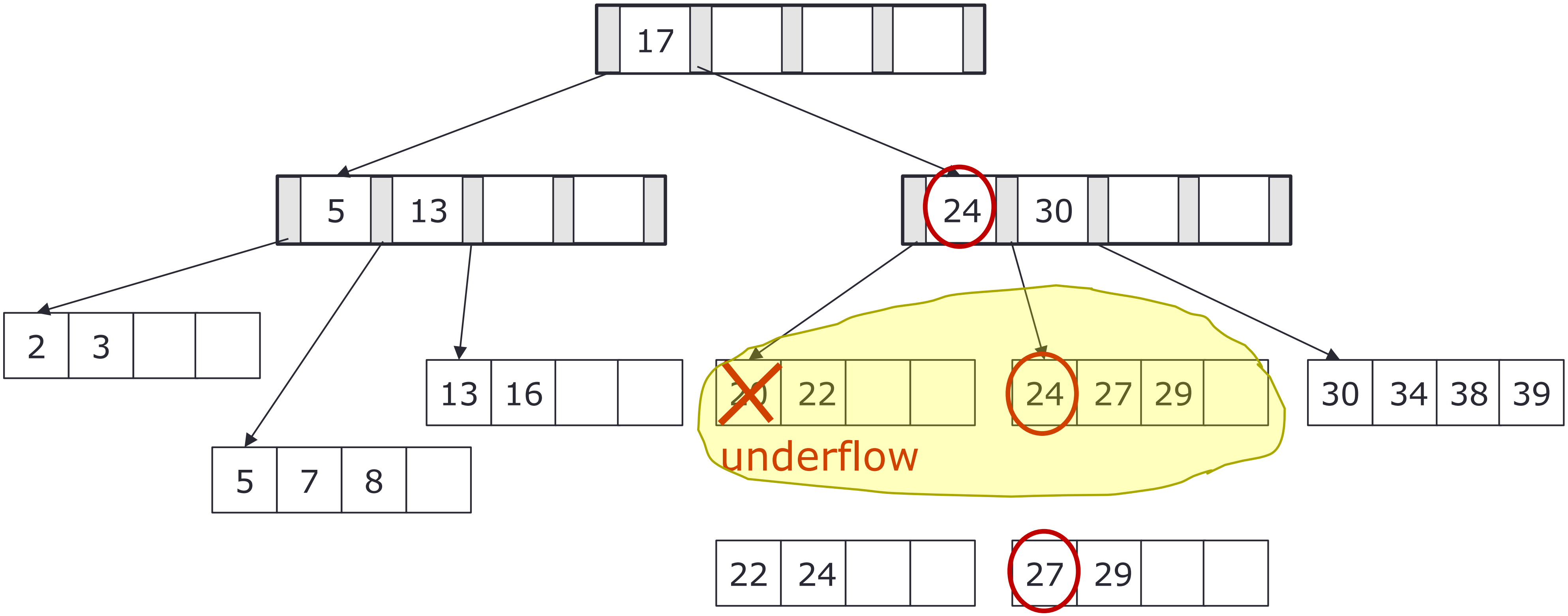
Delete key value 20 from the tree



[Let's try]

Recap 4: Deletion

Delete key value 20 from the tree



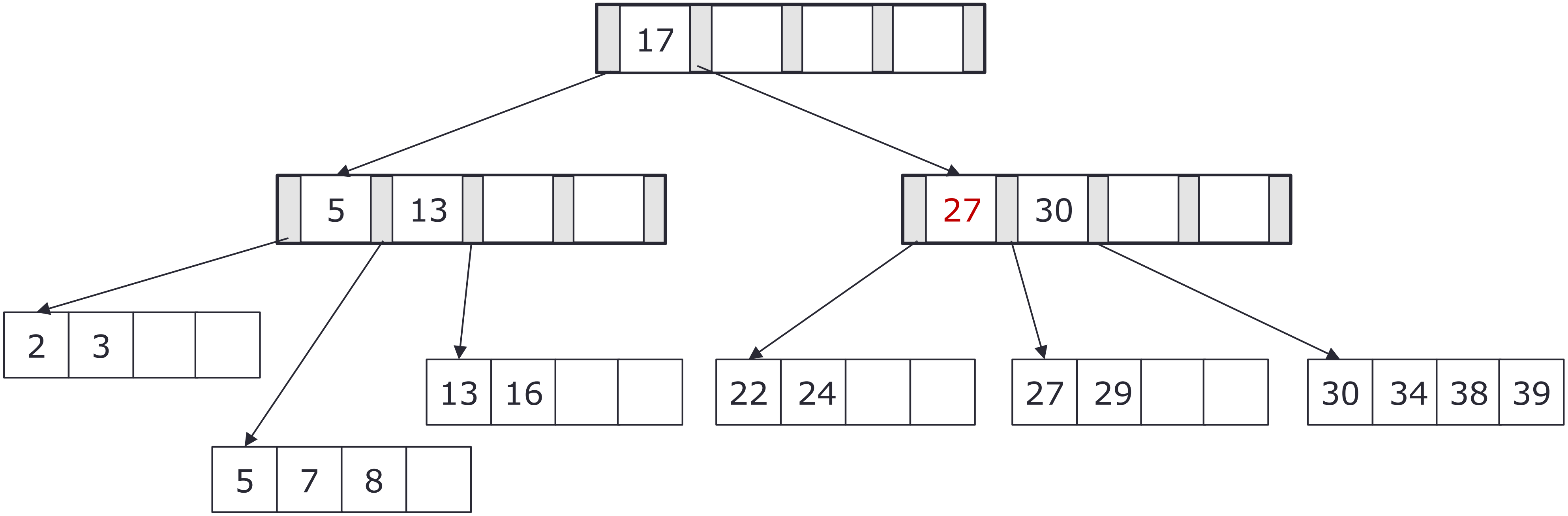
Remove key 20

Leaf node becomes underflow → borrow right sibling

Key 24 in parent results in unordered tree, update parent with 27

Recap 4: Deletion

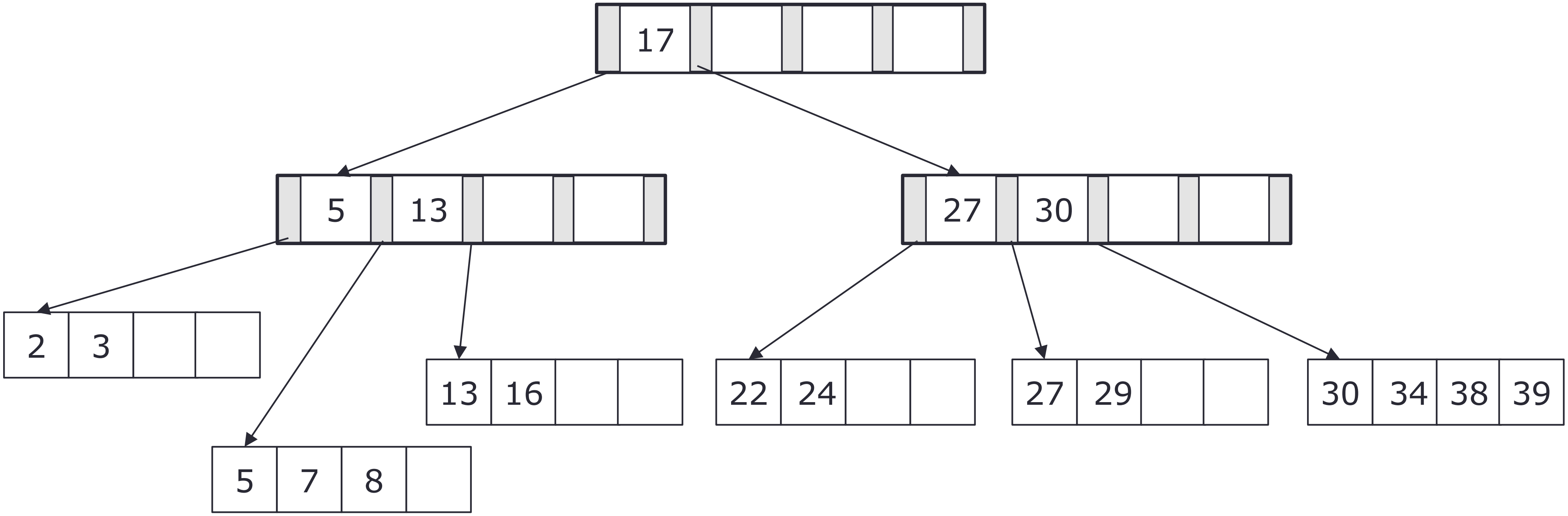
Delete key value 20 from the tree



Re-distribute the tree, update the pointers

Recap 5: Deletion

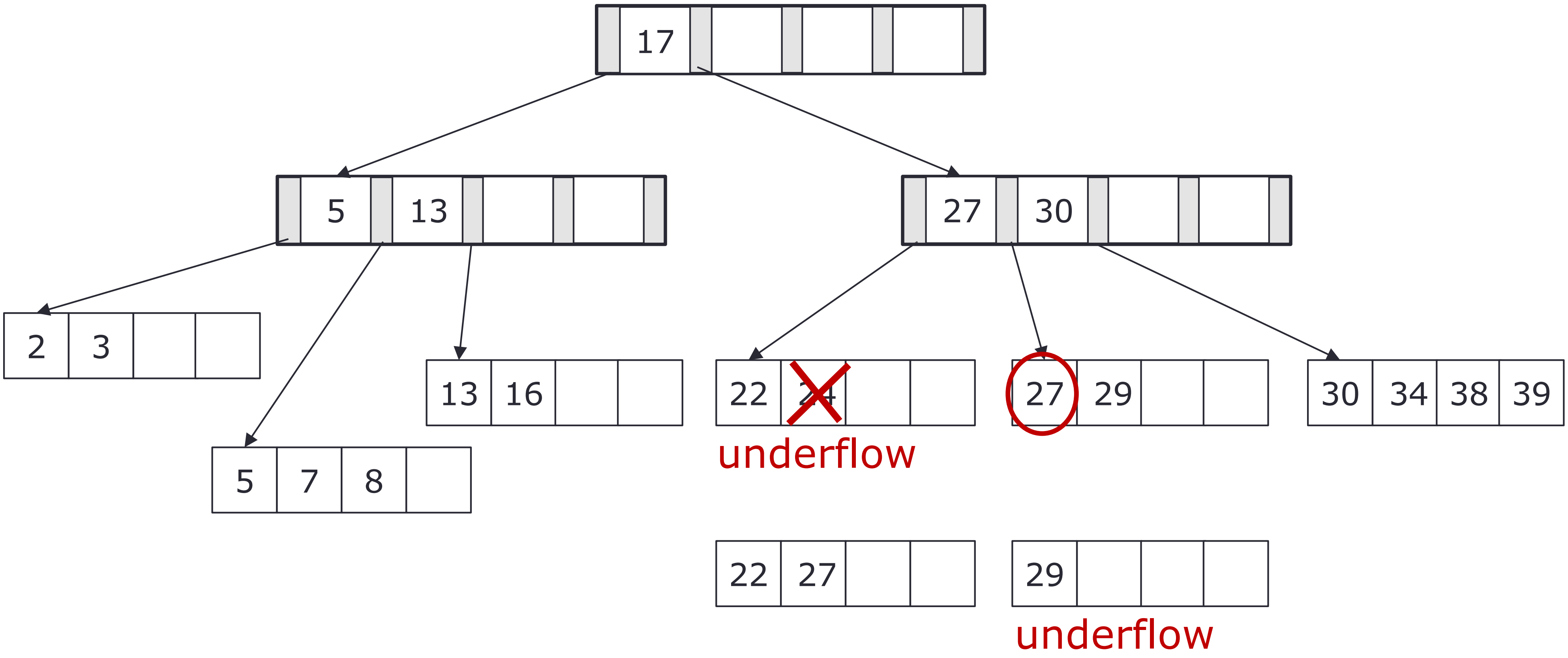
Delete key value 24 from the tree



[Let's try]

Recap 5: Deletion

Delete key value 24 from the tree



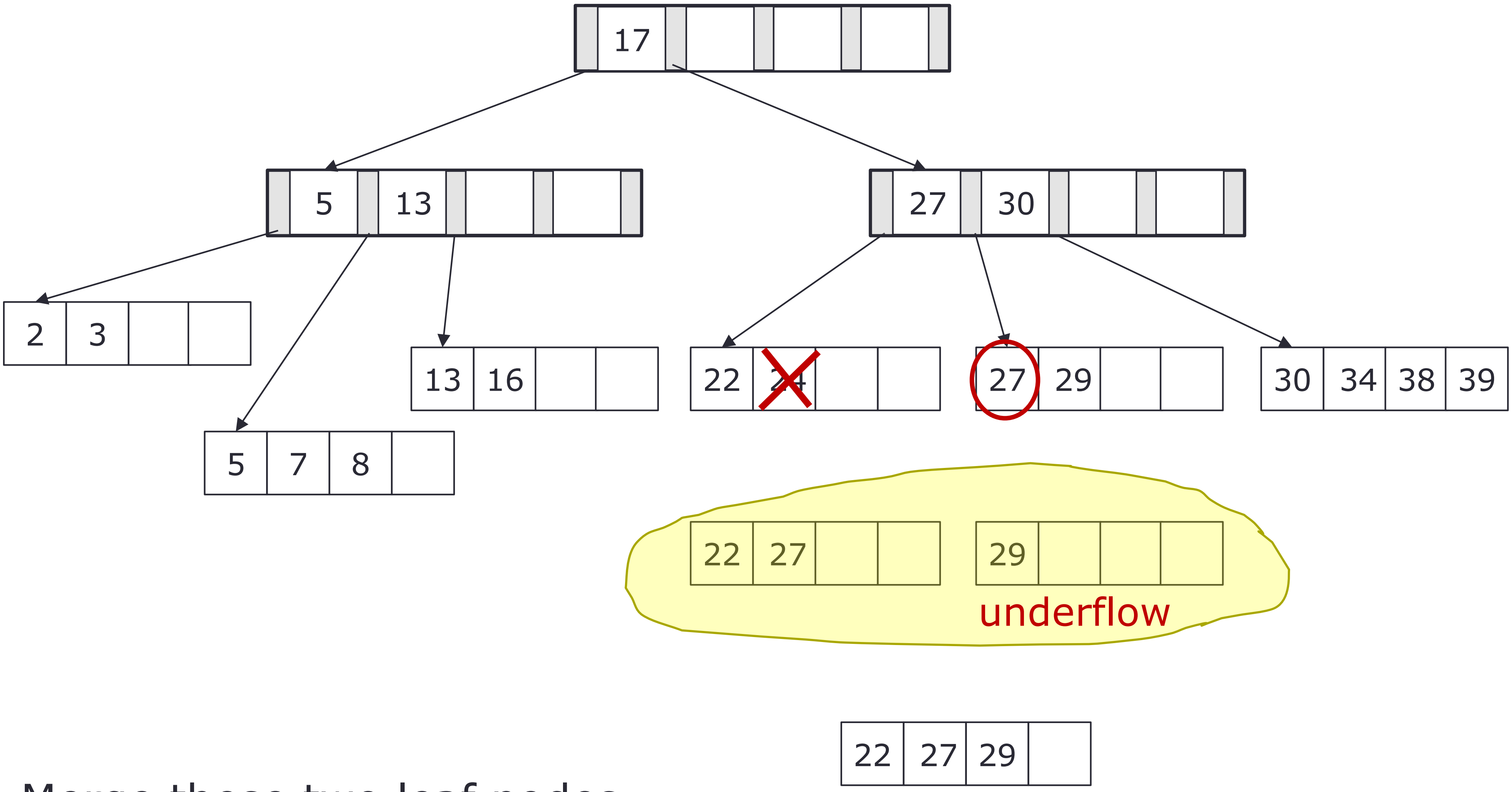
Remove key 24

Leaf node becomes underflow → borrow right sibling

Right sibling becomes underflow

Recap 5: Deletion

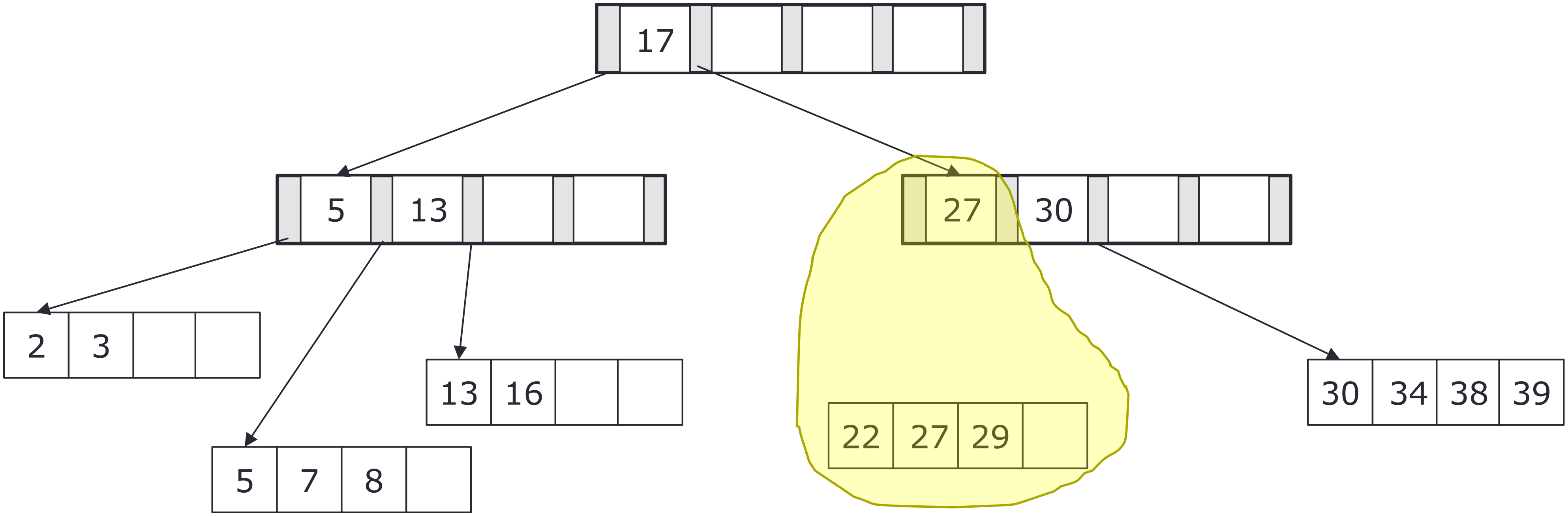
Delete key value 24 from the tree



Merge these two leaf nodes

Recap 5: Deletion

Delete key value 24 from the tree



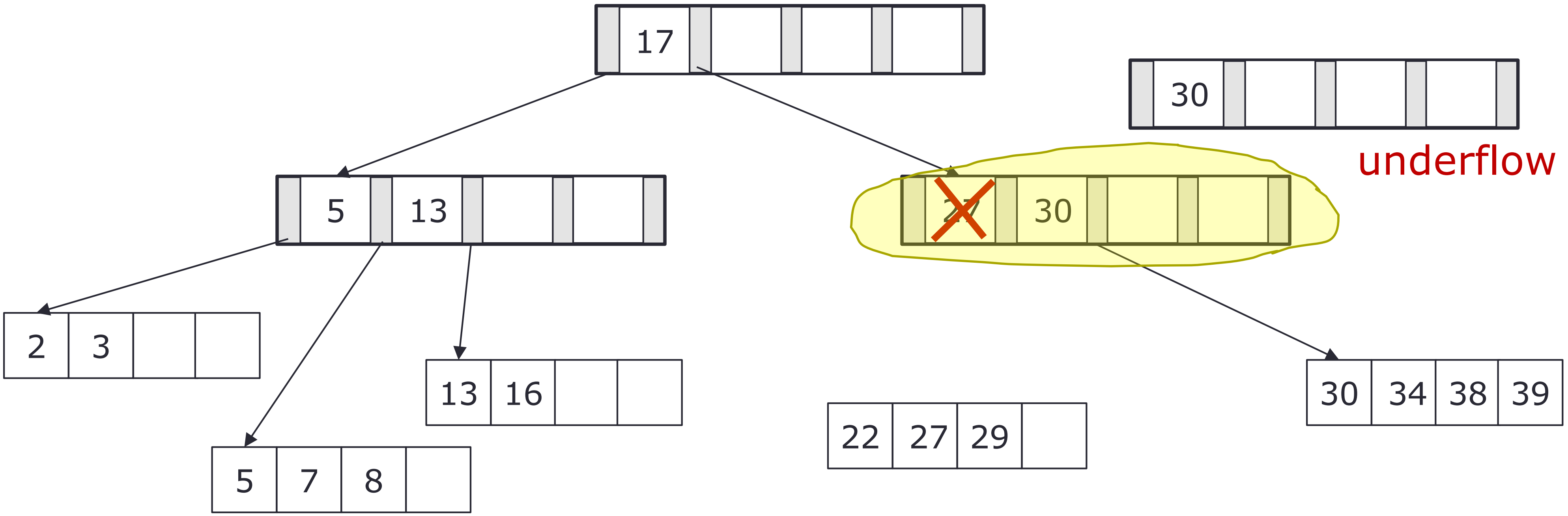
The tree is now unordered.

We now have one leaf, need to get rid of one pointer.

Re-distribute the tree to ensure the balance and preserve the order.

Recap 5: Deletion

Delete key value 24 from the tree

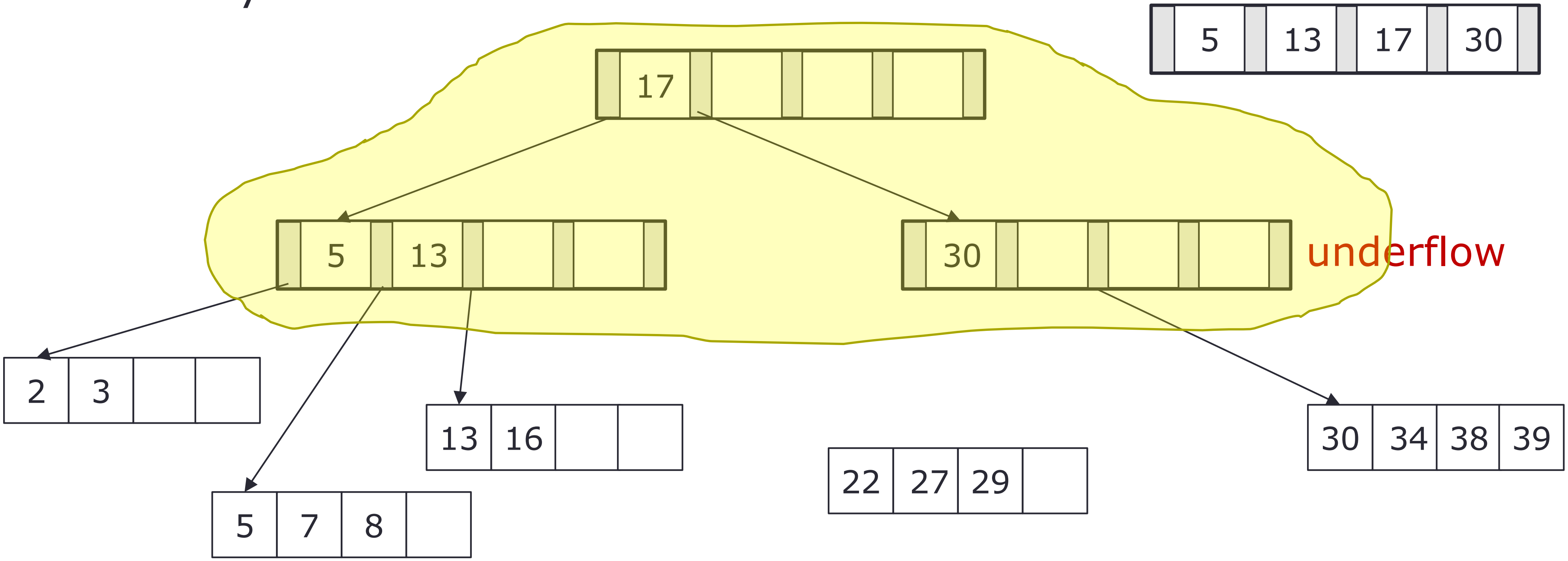


Throw away 27 (index-page, i.e., parent)

Parent becomes underflow

Recap 5: Deletion

Delete key value 24 from the tree

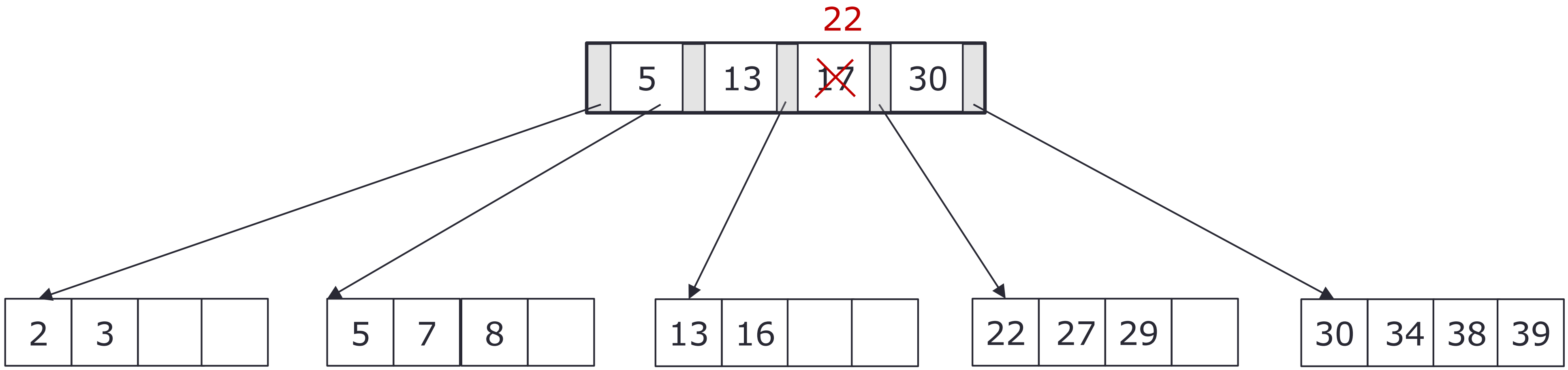


Pull 17 from root down

Merge the two internal nodes

Recap 5: Deletion

Delete key value 24 from the tree



Re-distribute the tree, update the pointers