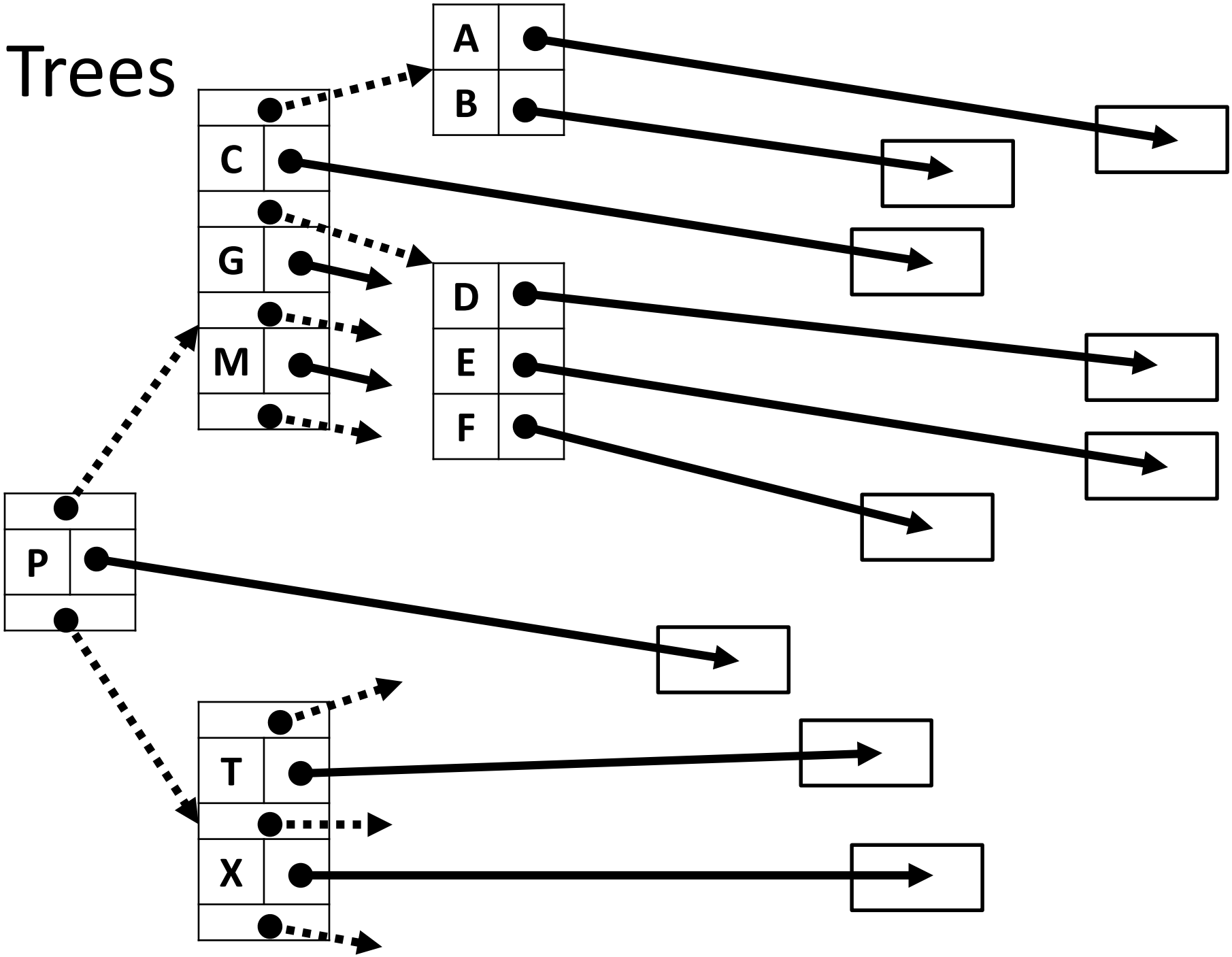
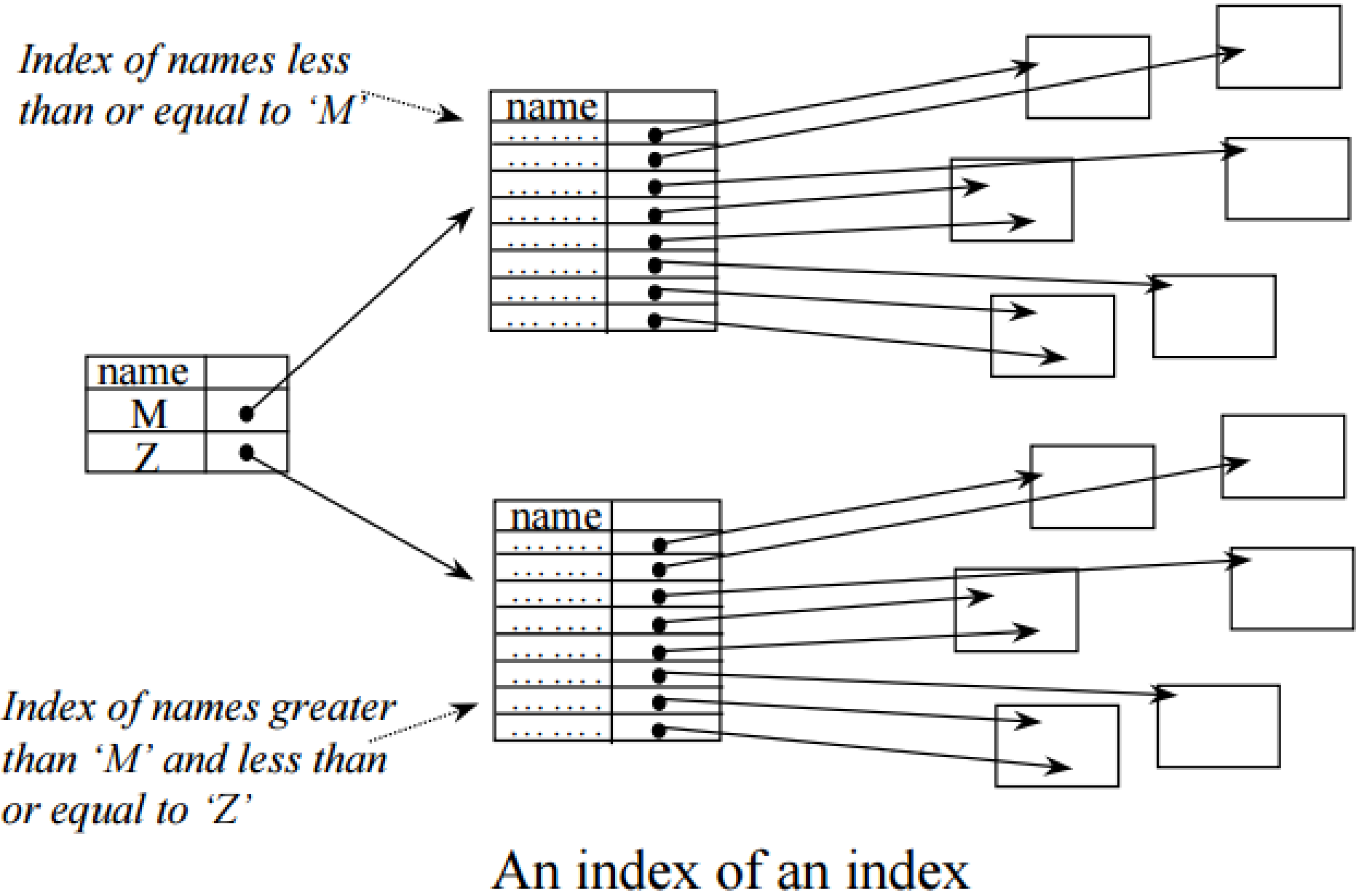
B^+ Trees

B Trees

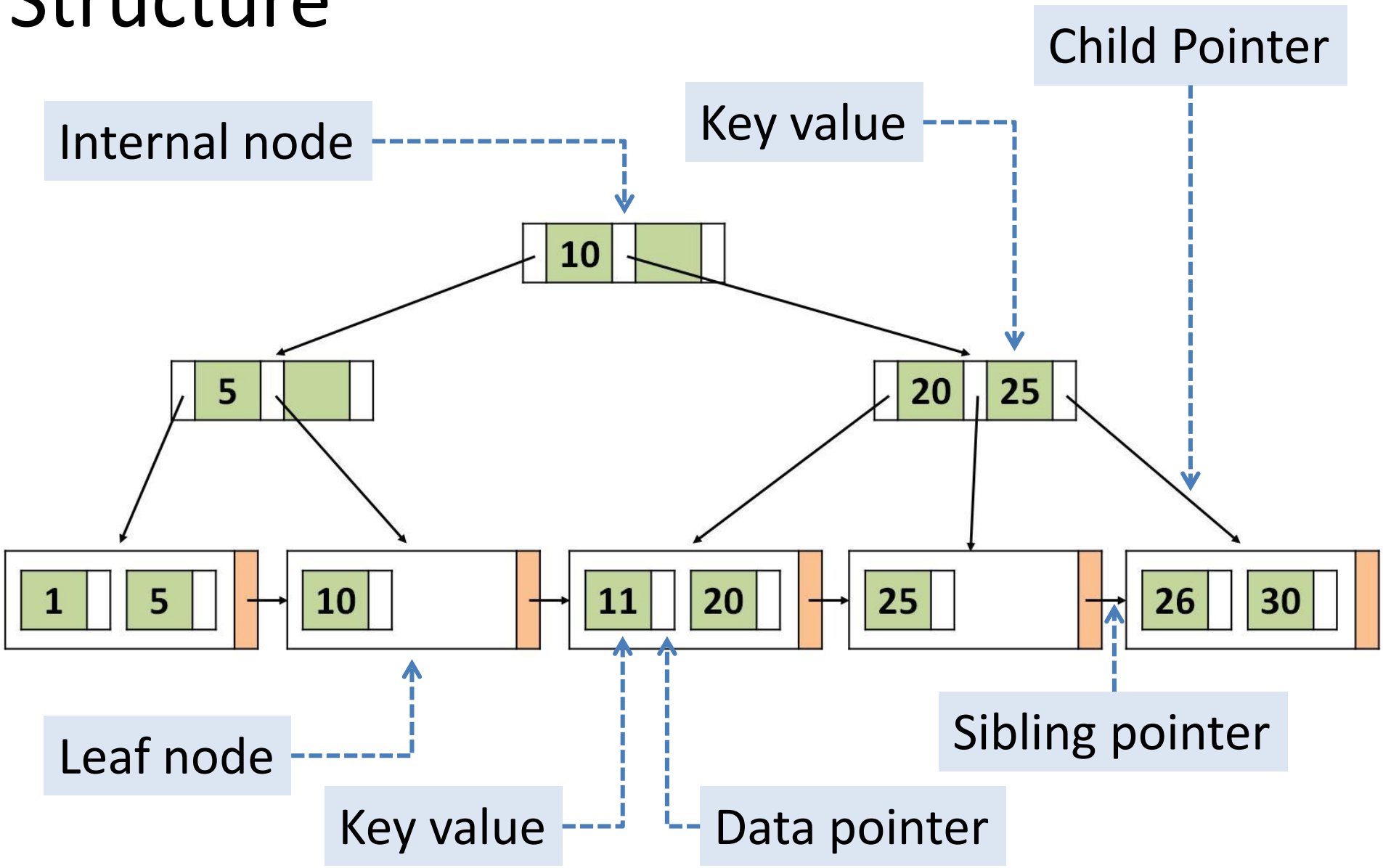


B+ Trees – Multi-level Index



An index of an index

Structure

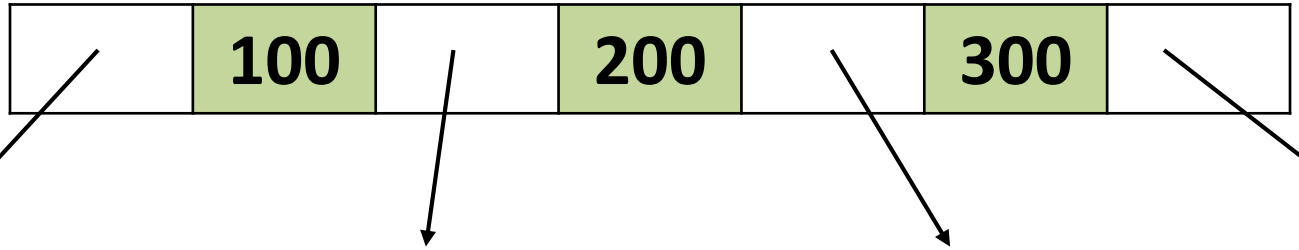

Child Pointer
Internal node
Key value
10
5
20
25
1
5
10
11
20
25
26
30
Leaf node
Sibling pointer
Key value
Data pointer

Internal nodes

Ki – Key value

Pi – Child pointer

K1 < K2 < K3



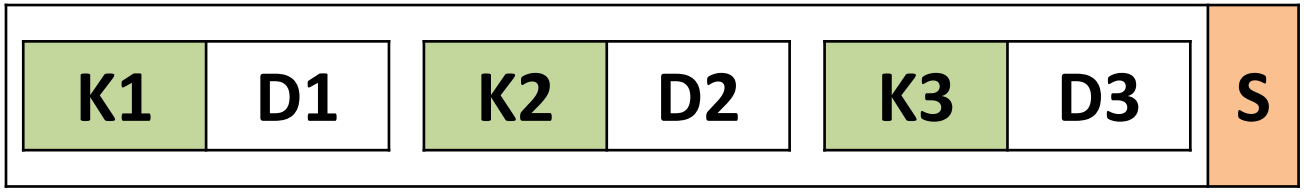
Pointer to nodes with keys less than or equal to 100

Pointer to nodes with keys less than or equal to 200 and greater than 100

Pointer to nodes with keys less than or equal to 300 and greater than 200

Pointer to nodes with keys greater than 300

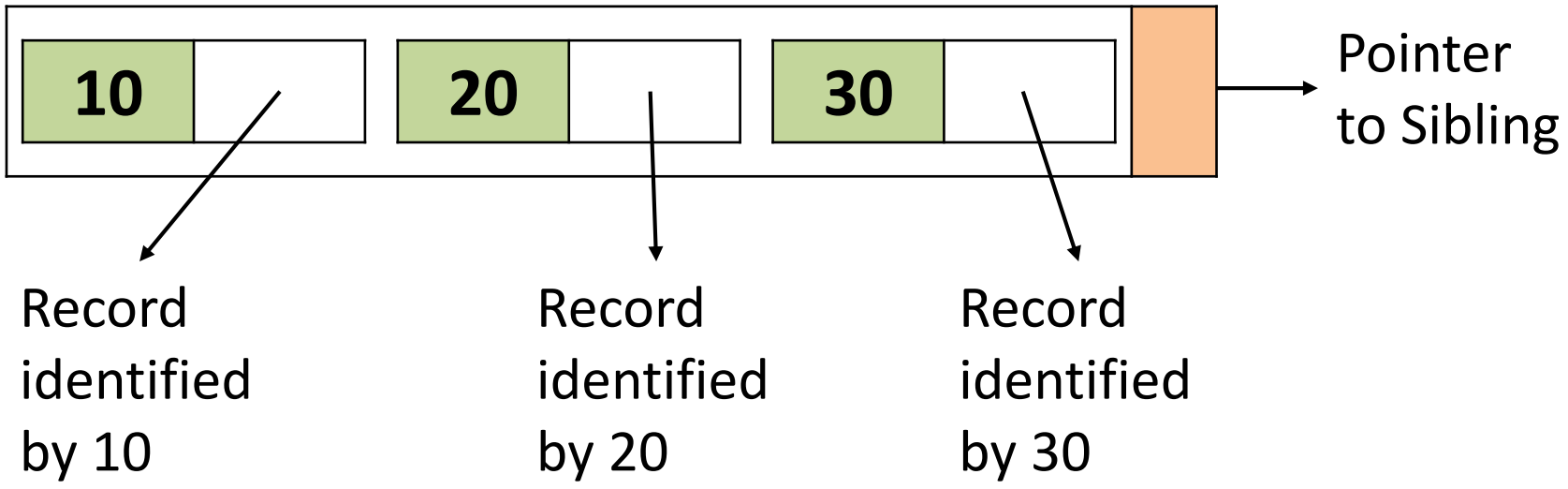
Leaf Nodes



Ki – Key value

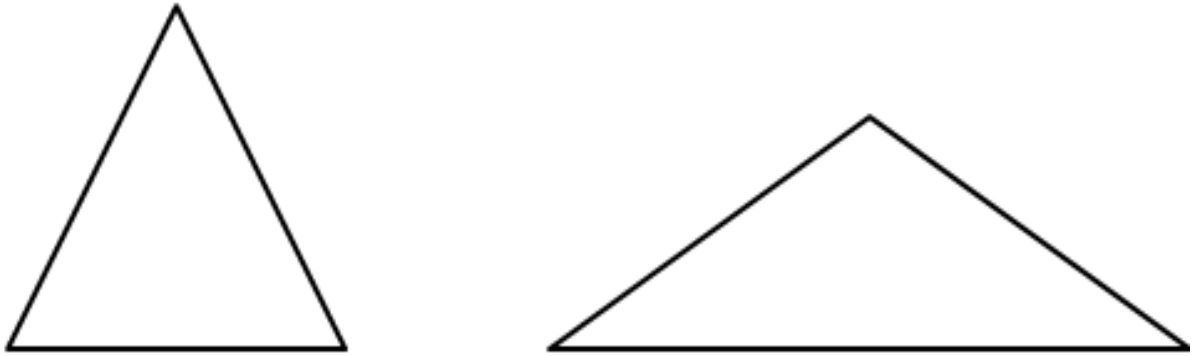
Di – Data pointer

K1 < K2 < K3



Order

- The ***order*** of a B+ Tree is the maximum number of pointers that an internal node can hold.

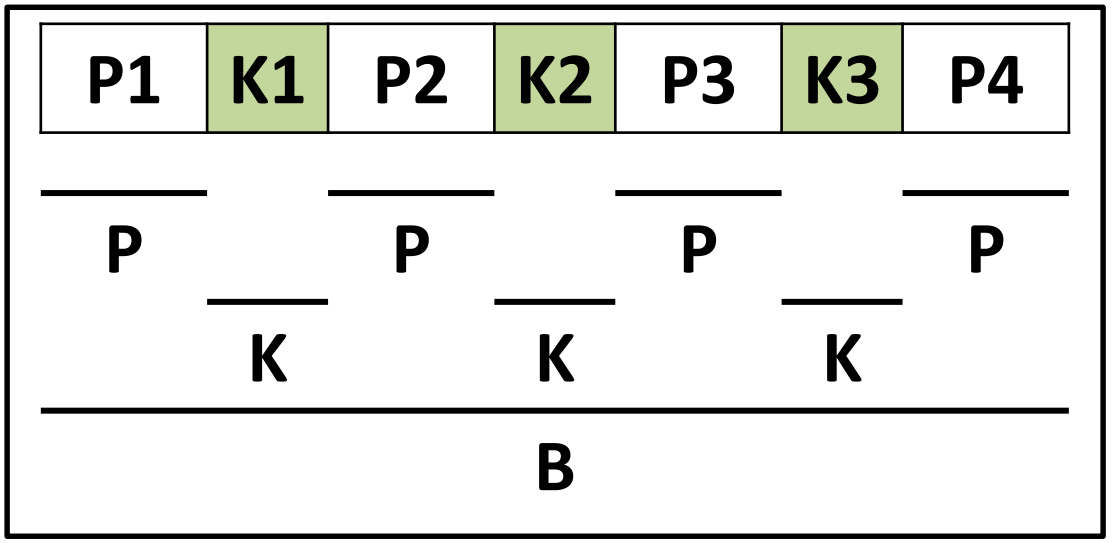


Small Order Size Large Order Size

- A B+ Tree of order $\boldsymbol{m}$ can hold
  - At most $\boldsymbol{m} - \mathbf{1}$ keys and $\boldsymbol{m}$ pointers.
  - At least $\left\lceil \frac{\boldsymbol{m}}{\mathbf{2}} \right\rceil$ pointers.

Calculating the Order Size

- B = Size of block
- K = Size of Key
- P = Size of Pointer
- O = Order of B+ tree



$$B \geq (O \times P) + ((O-1) \times K) \qquad \frac{B+K}{P+K} \geq O$$

- Example: B = 4KB, K = 8B, P = 8B

$$\frac{4 \times 1024 + 8}{8 + 8} = 256.5 \geq O$$

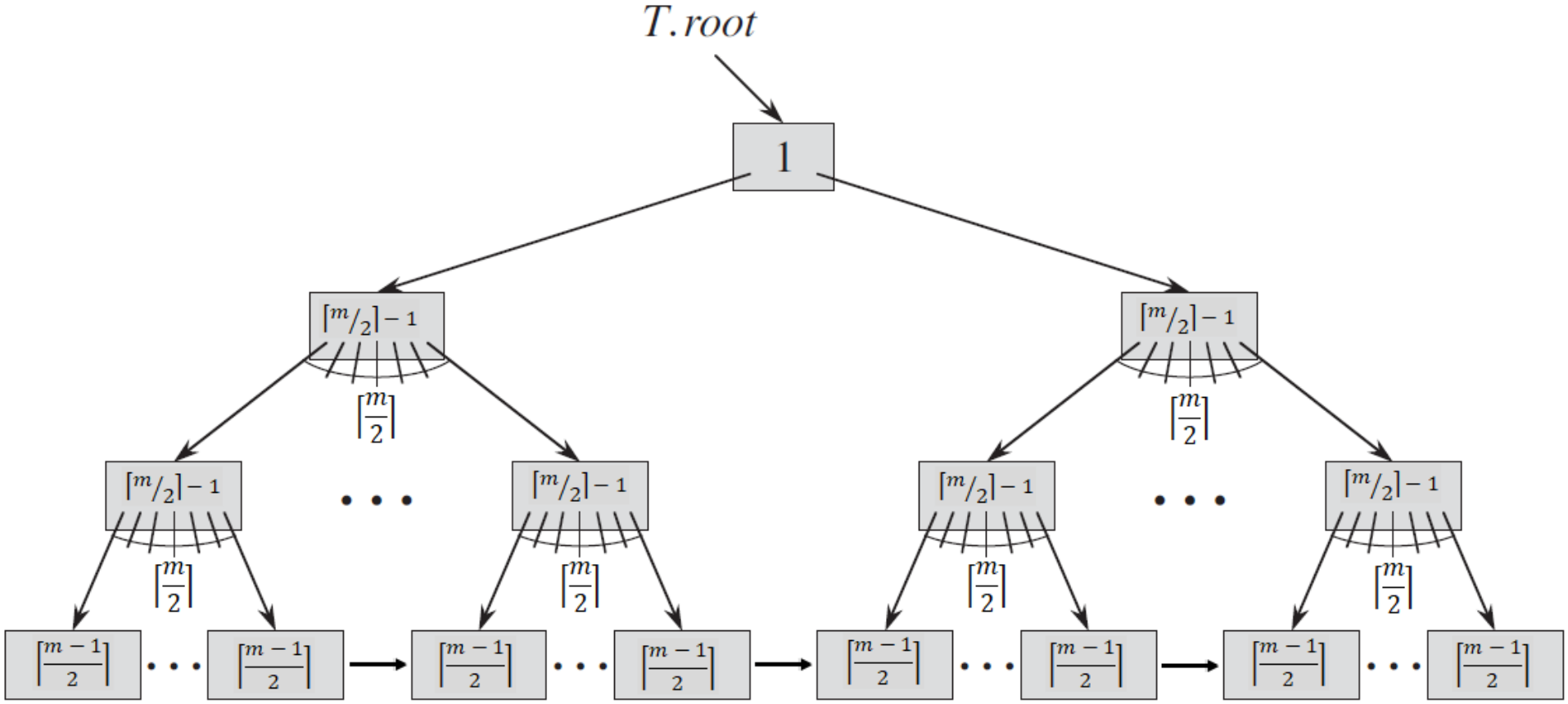
Size

- On an average 67% part of the nodes are full.
- If, order is 256
  - Average number of pointers = 0.67 x 256 = 171
  - 171 pointers and 170 keys per node.

	Nodes	**Entries**	**Pointers**
Root	1	170	171
Level 1	171	29,070	29,241
Level 2	29,241	4,970,970	5,000,211
Leaf	5,000,211	850,035,870	

Height

- Let, ***m*** be the order of B+ Tree, then minimum number of nodes at each level.



Contd…

- A B+ Tree with N keys has $N + 1$ leaf nodes,
- Therefore,

$$N + 1 \geq 2 \left\lceil \frac{m}{2} \right\rceil^{l-1}$$

$$1 + log_{\left\lceil \frac{m}{2} \right\rceil} \left(\frac{N + 1}{2} \right) \geq l$$

Example

- $N = 1{,}000{,}000$
- $m = 256$
- Determine height.

Therefore, to store 1,000,000 keys a B+ tree of at most 3 levels is required.

$$1 + log_{\lceil \frac{m}{2} \rceil}\left(\frac{N+1}{2}\right) \geq l$$

$$= 1 + log_{\lceil \frac{256}{2} \rceil}\left(\frac{1000000+1}{2}\right)$$

$$= 1 + log_{\lceil 128 \rceil}\left(\frac{1000001}{2}\right)$$

$$= 1 + log_{\lceil 128 \rceil}(500000.5)$$

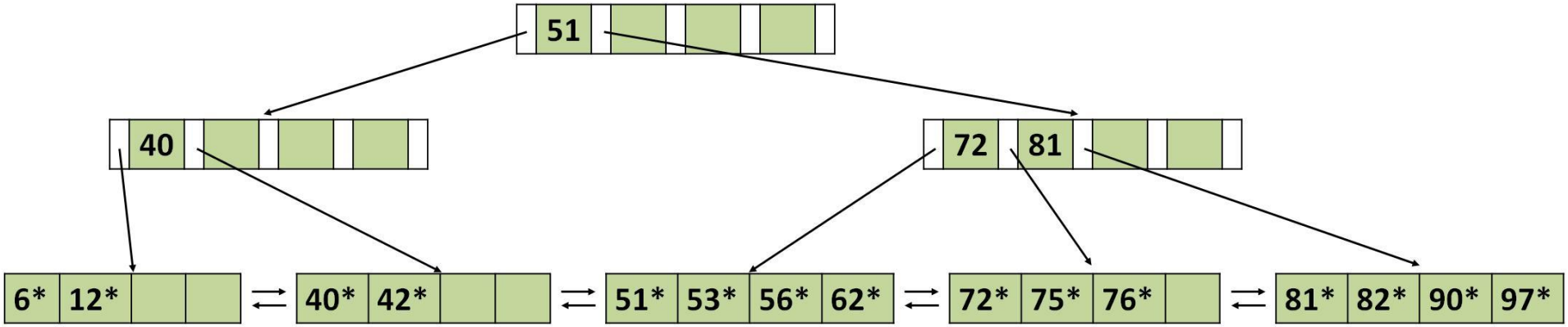
$$= 1 + 2.7 = 3.7 \geq l$$

Properties

- All paths from root to leaf are of the same length.
- The root node points to at least two nodes.
- All non-root nodes are at least half full.
- A B+ Tree grows upwards.
- A B+ Tree is balanced.
- Sibling pointers allow sequential searching.

B+ Tree Queries

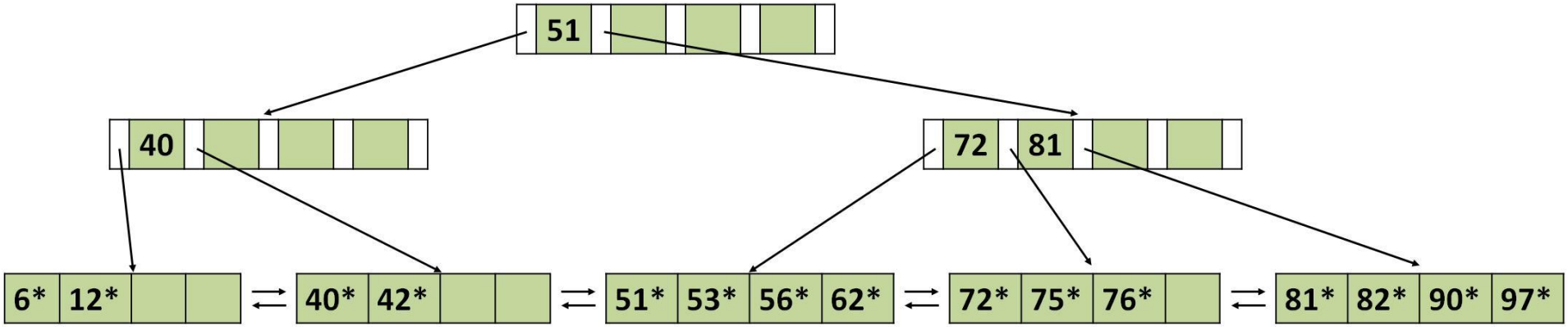
- Exact match queries.
- Search for record with key 42.



  - 42 < 51? Yes. Choose left sub-tree.
  - 42 < 40? No. Choose right sub-tree.
  - 42 == 40? No. Move to next key in the leaf.
  - 42 == 42? Found. Access the record with key 42.

Contd…

- Range queries.
- Search for records with keys >= 40 and <= 80.



  - Start a search for lower range, i.e. 40.
  - 40 < 51? Yes. Choose left sub-tree.
  - 40 < 40? No. Choose right sub-tree.
  - 40 == 40? Yes. Access the record with key 40 and keep on moving right using a sibling pointer, till key is <= 80.

Insertion

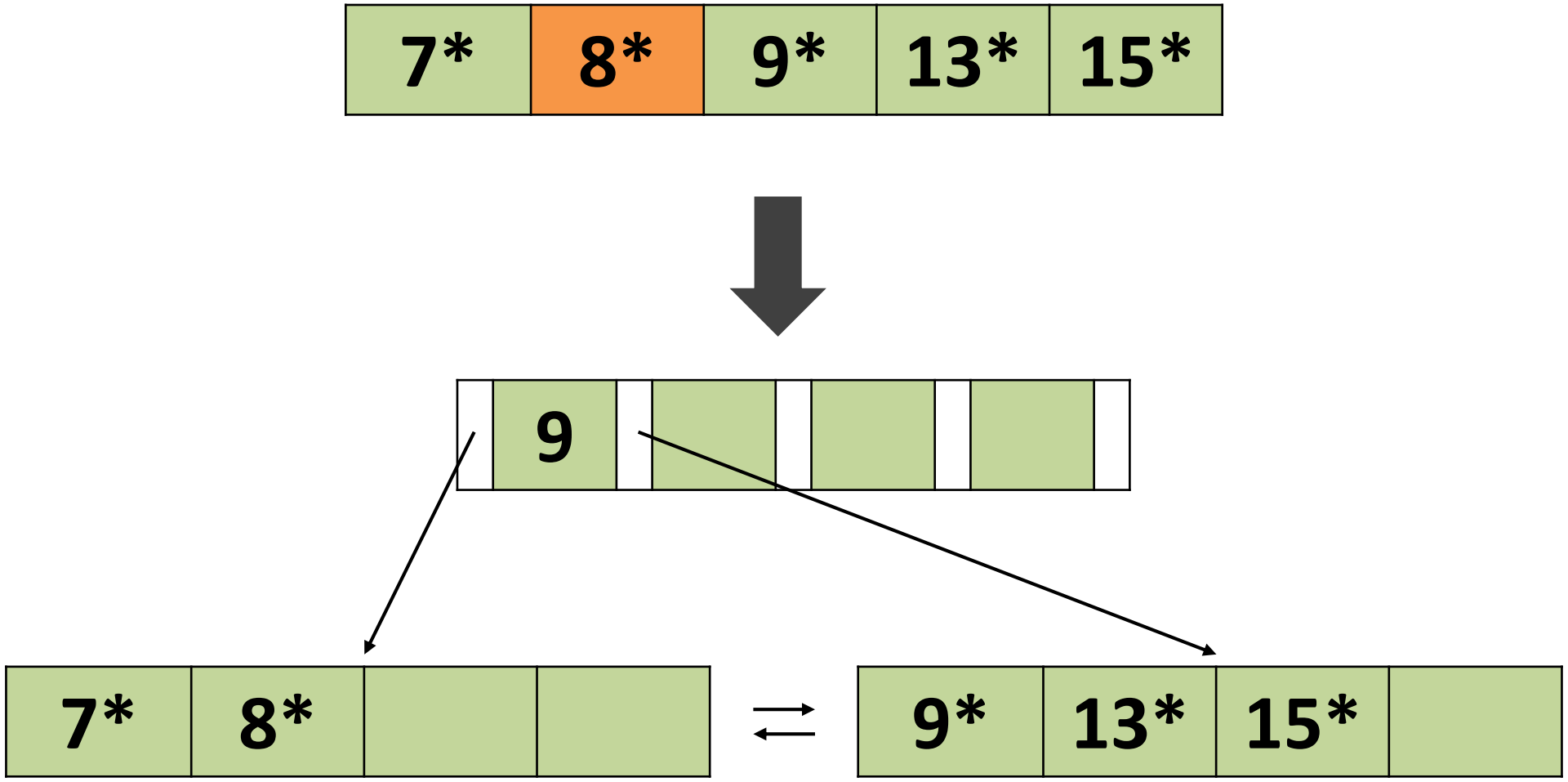
- Similar to B Trees, i.e. a new value gets inserted at the leaf only.
- Use post split.
  - First insert the key,
  - Then check for overflow.
- How to handle an overflow?
  - Depends on the type of a node being overflowed.
    - Leaf node, or
    - Internal node.

Overflow – Leaf Node

- Split into two nodes.
- First node contains $ceil((m-1)/2)$ values.
- Second node contains the remaining values.
- Copy the smallest search-key value of the second node to the parent node.
- Example: Insert 8 in the leaf node of a B+ tree with $m = 5$.

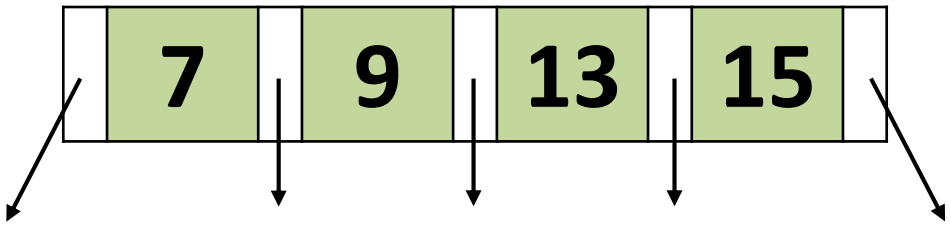
7*	9*	13*	15*

Contd…

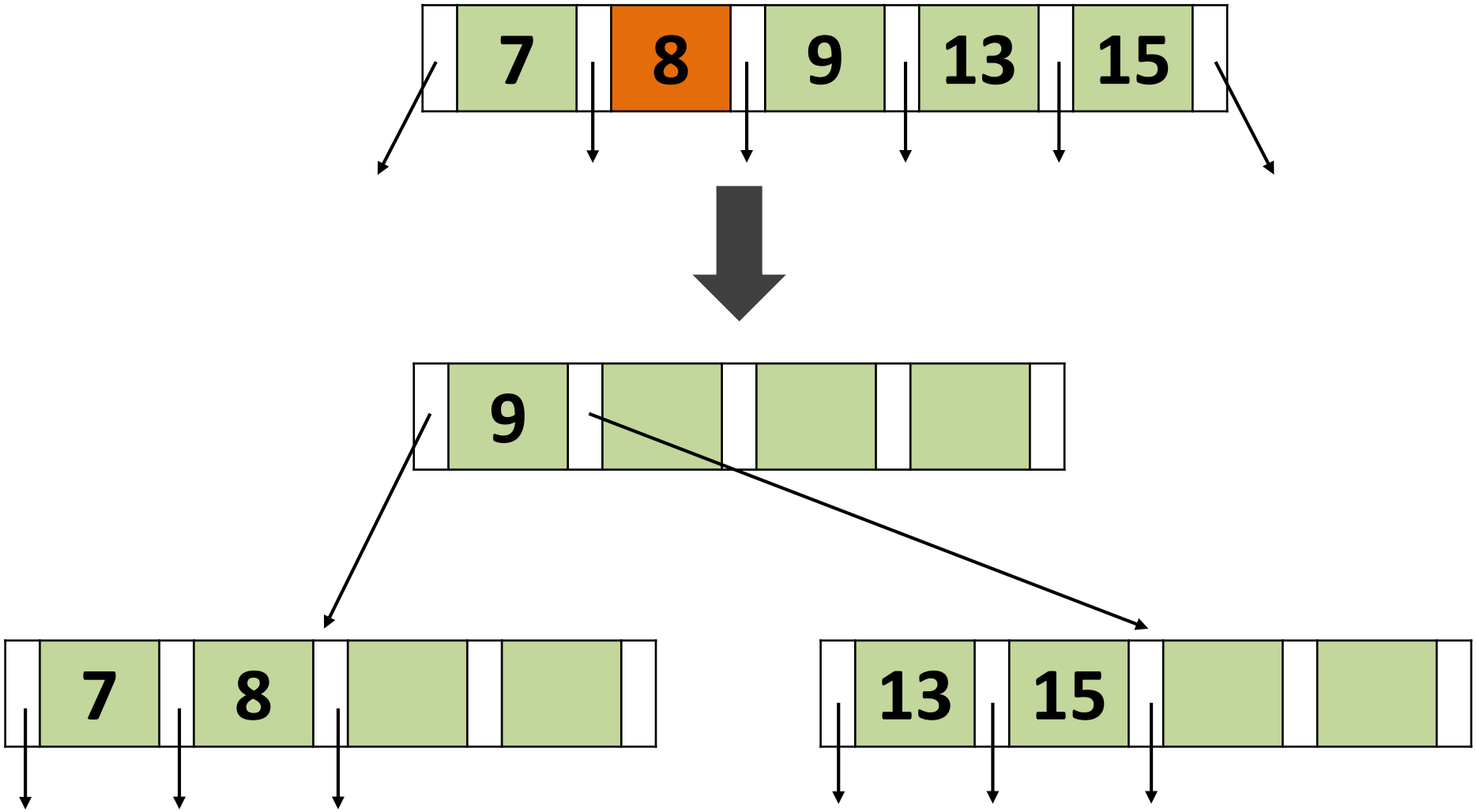


Overflow – Internal Node

- Split into two nodes:
- First node contains $\boldsymbol{ceil(m/2) - 1}$ values.
- Move the smallest of the remaining values, together with pointer, to the parent.
- Second node contains the remaining values.
- Example: Insert 8 in the internal node of a B+ tree with $\boldsymbol{m = 5}$.



Contd…



Insertion

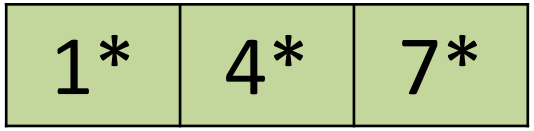
- Construct a B+ tree for (1, 4, 7, 10, 17, 21, 31, 25, 19, 20, 28, 42) with order 4.

Order m = 4	Minimum	Maximum
#Keys	Root: 1 Internal: ceil(4/2) – 1 = 1 Leaf: ceil((4–1)/2) = 2	3
#Childs	ceil(4/2) = 2	4

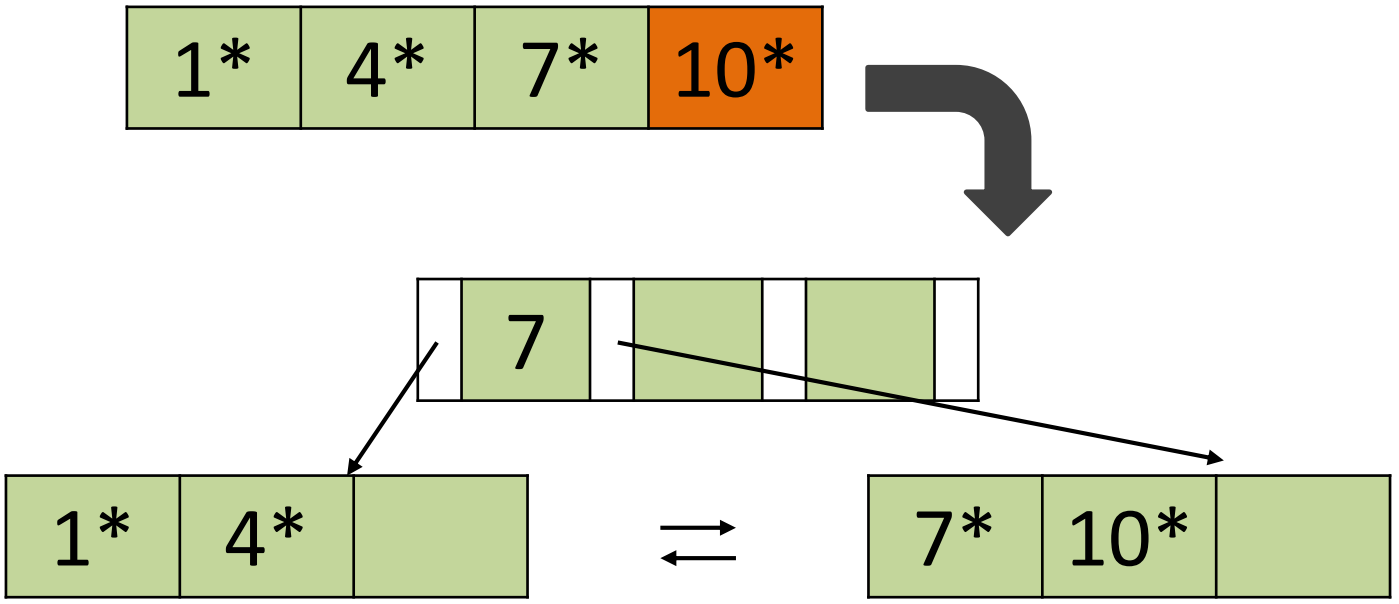
Insert 1, 4, 7, 10, 17, 21, 31, 25, 19, 20, 28, 42

$m = 4$

- Insert 1, 4, 7.



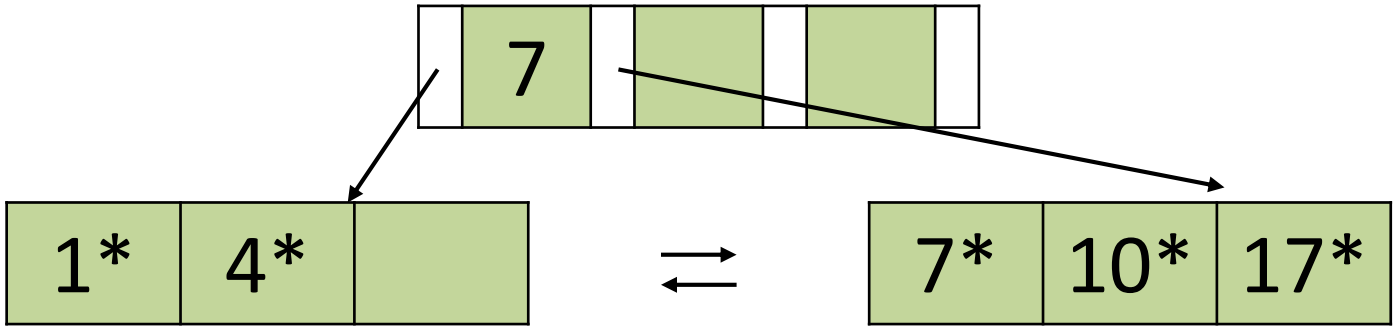
- Insert 10.



Insert 1, 4, 7, 10, 17, 21, 31, 25, 19, 20, 28, 42

$m = 4$

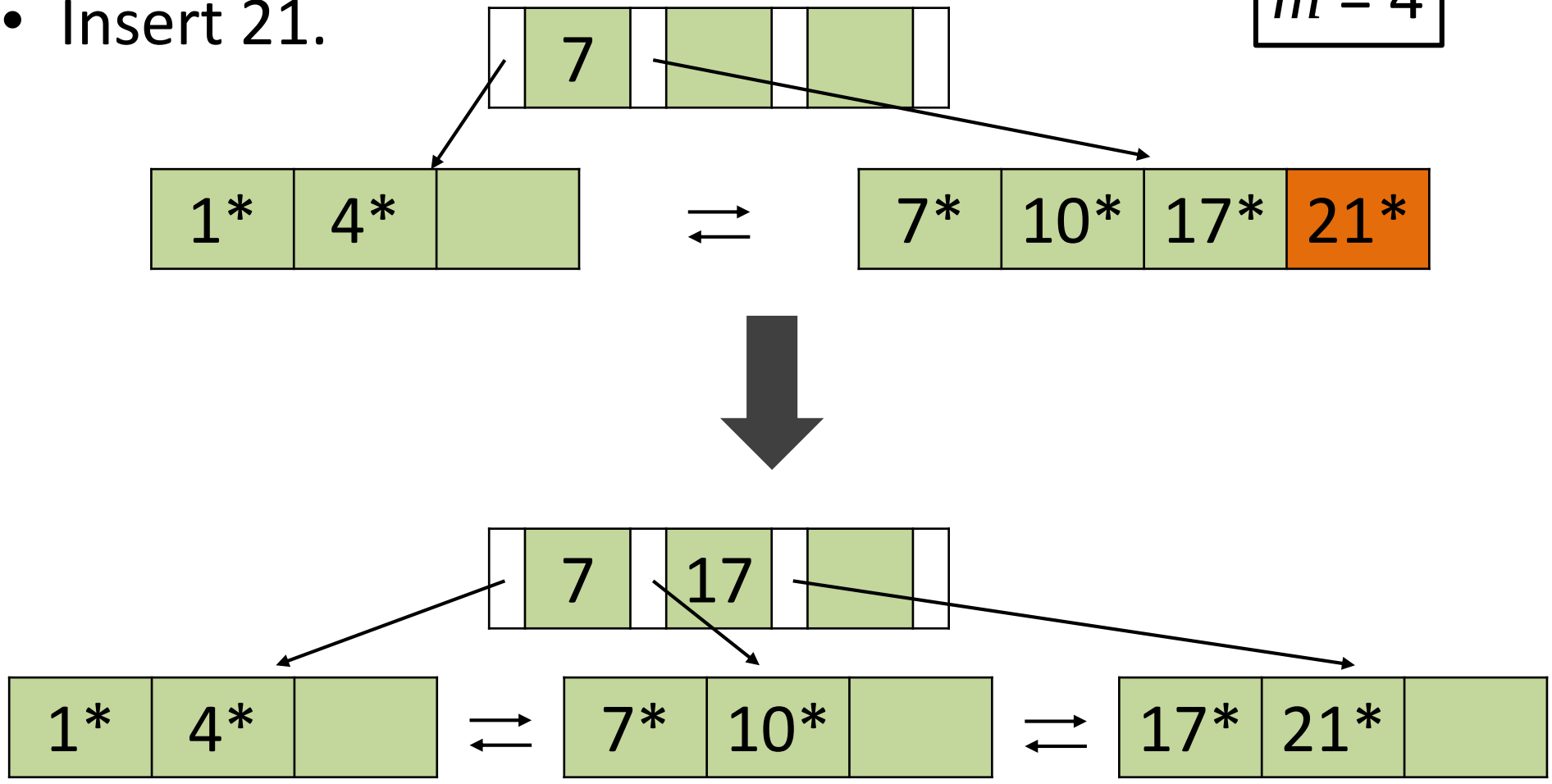
- Insert 17.



Insert 1, 4, 7, 10, 17, 21, 31, 25, 19, 20, 28, 42

$m = 4$

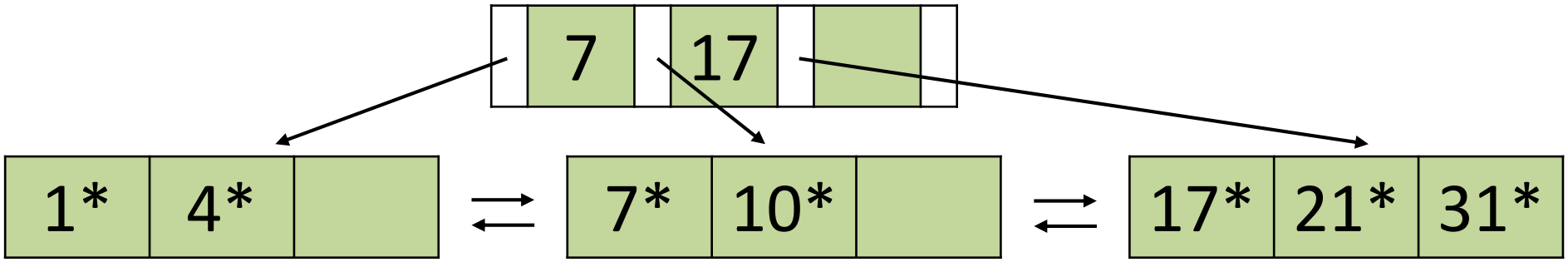
- Insert 21.



Insert 1, 4, 7, 10, 17, 21, 31, 25, 19, 20, 28, 42

$m = 4$

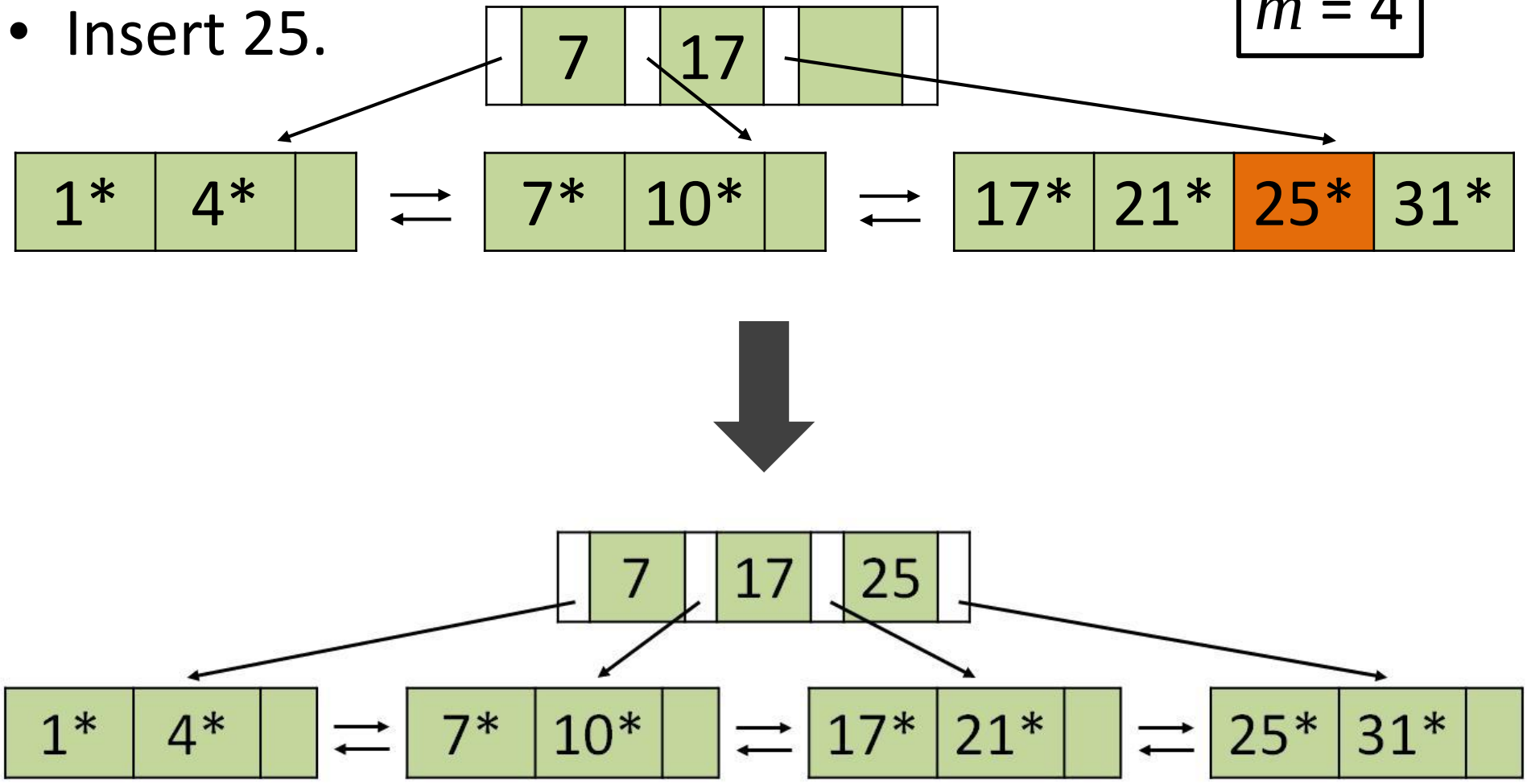
- Insert 31.



Insert 1, 4, 7, 10, 17, 21, 31, 25, 19, 20, 28, 42

$m = 4$

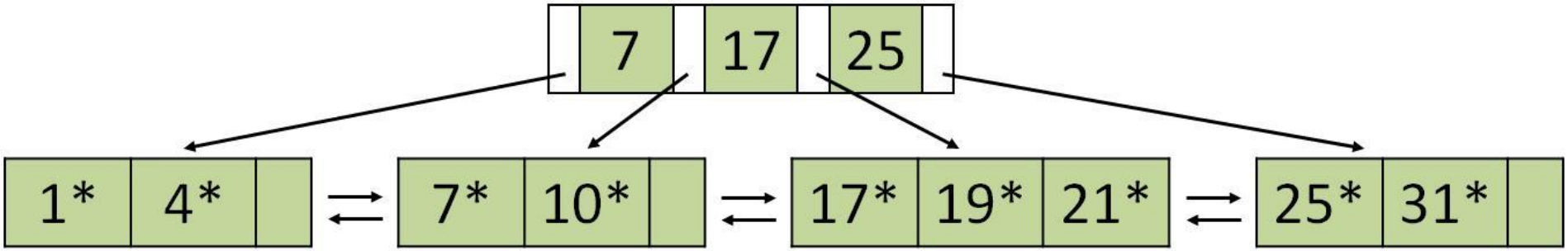
- Insert 25.



Insert 1, 4, 7, 10, 17, 21, 31, 25, 19, 20, 28, 42

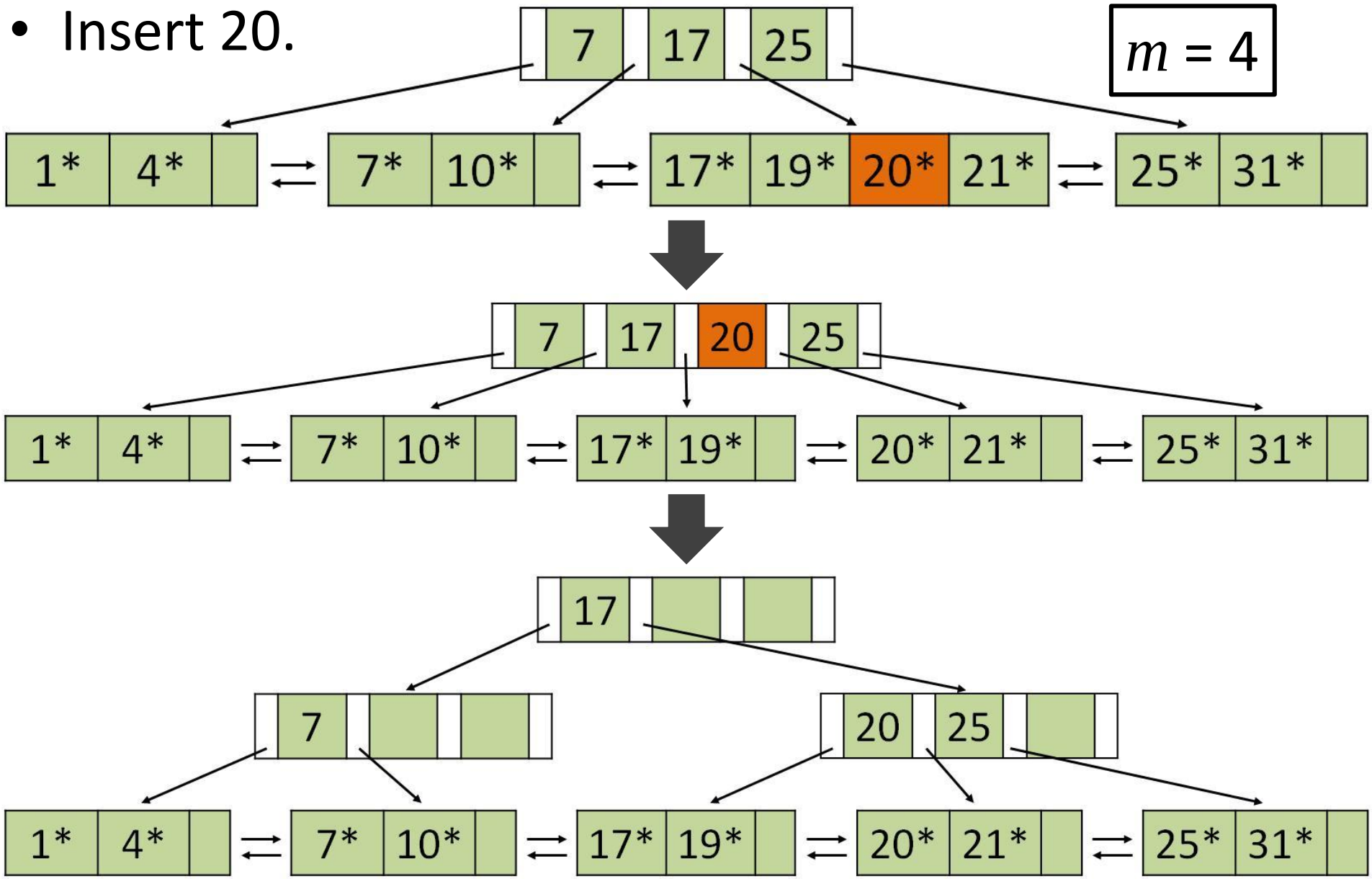
$m = 4$

- Insert 19.



Insert 1, 4, 7, 10, 17, 21, 31, 25, 19, 20, 28, 42

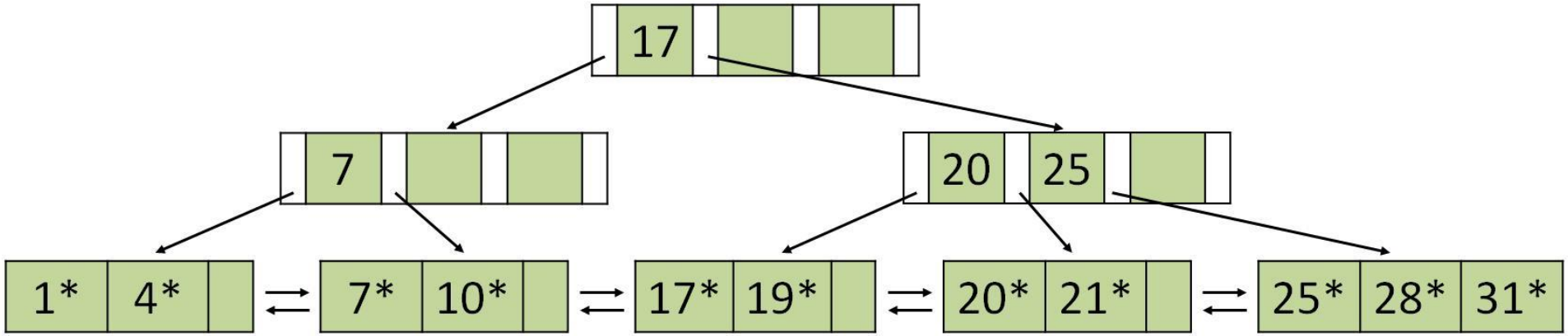
- Insert 20.



Insert 1, 4, 7, 10, 17, 21, 31, 25, 19, 20, 28, 42

$m = 4$

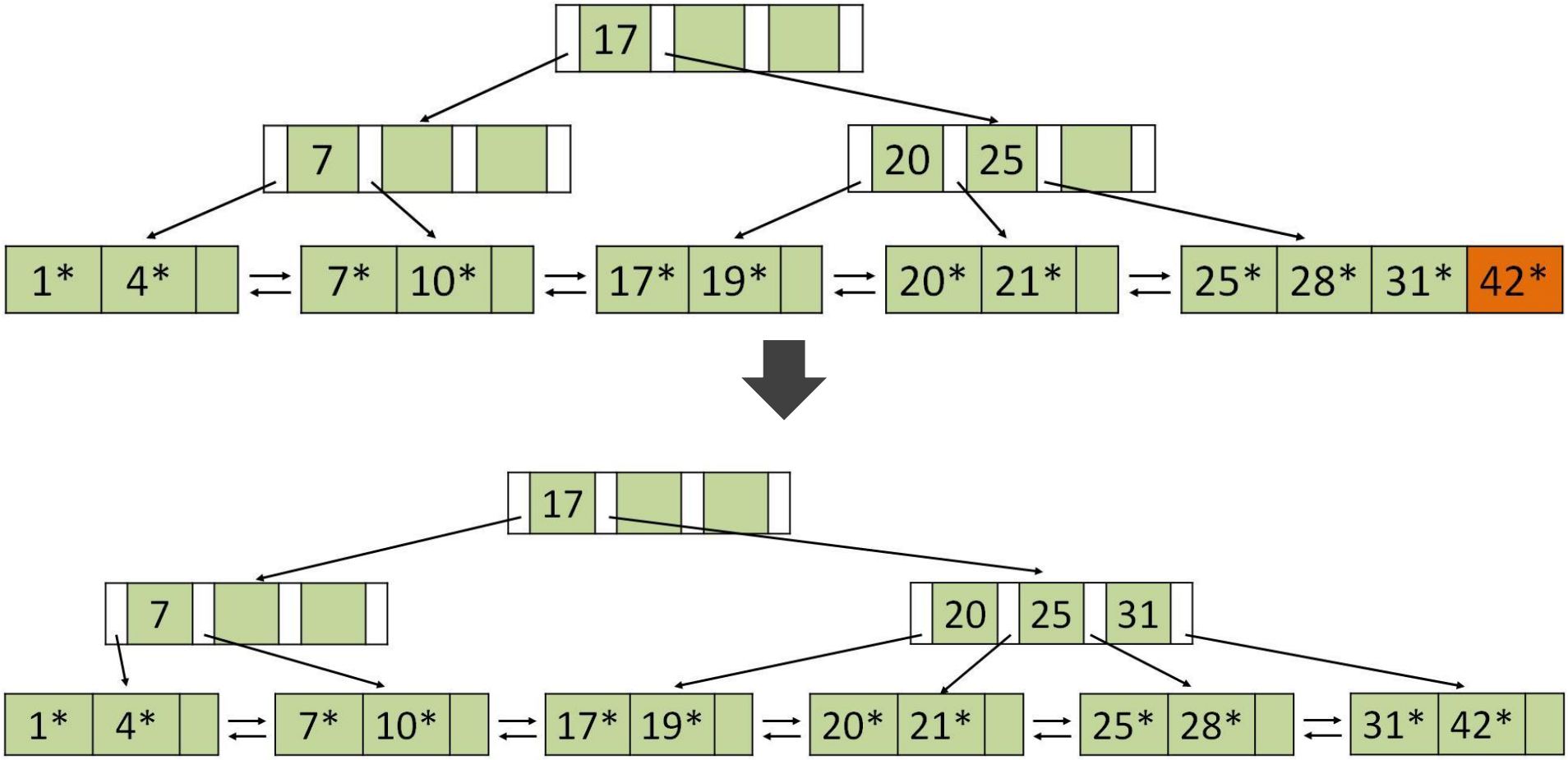
- Insert 28.



Insert 1, 4, 7, 10, 17, 21, 31, 25, 19, 20, 28, 42

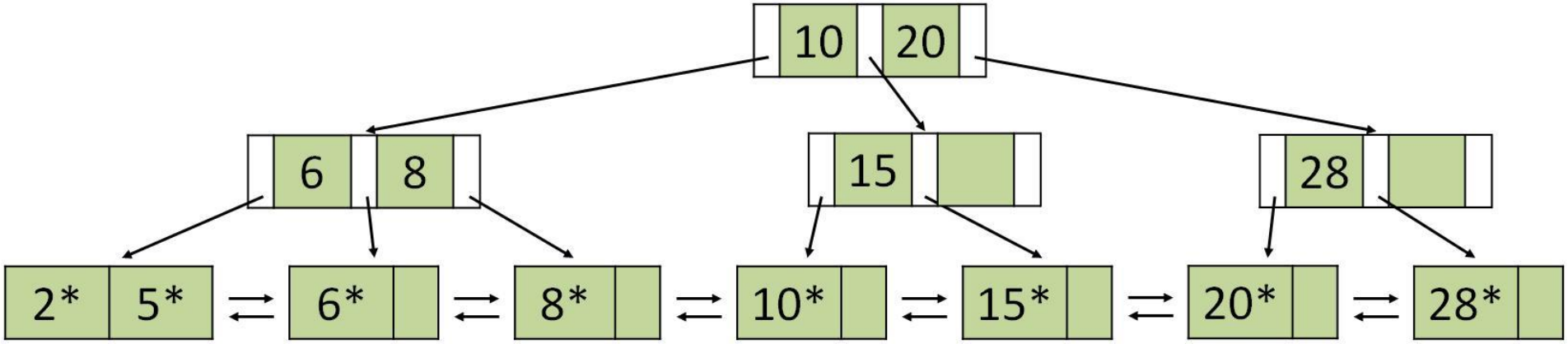
$m = 4$

- Insert 42.



Example 2

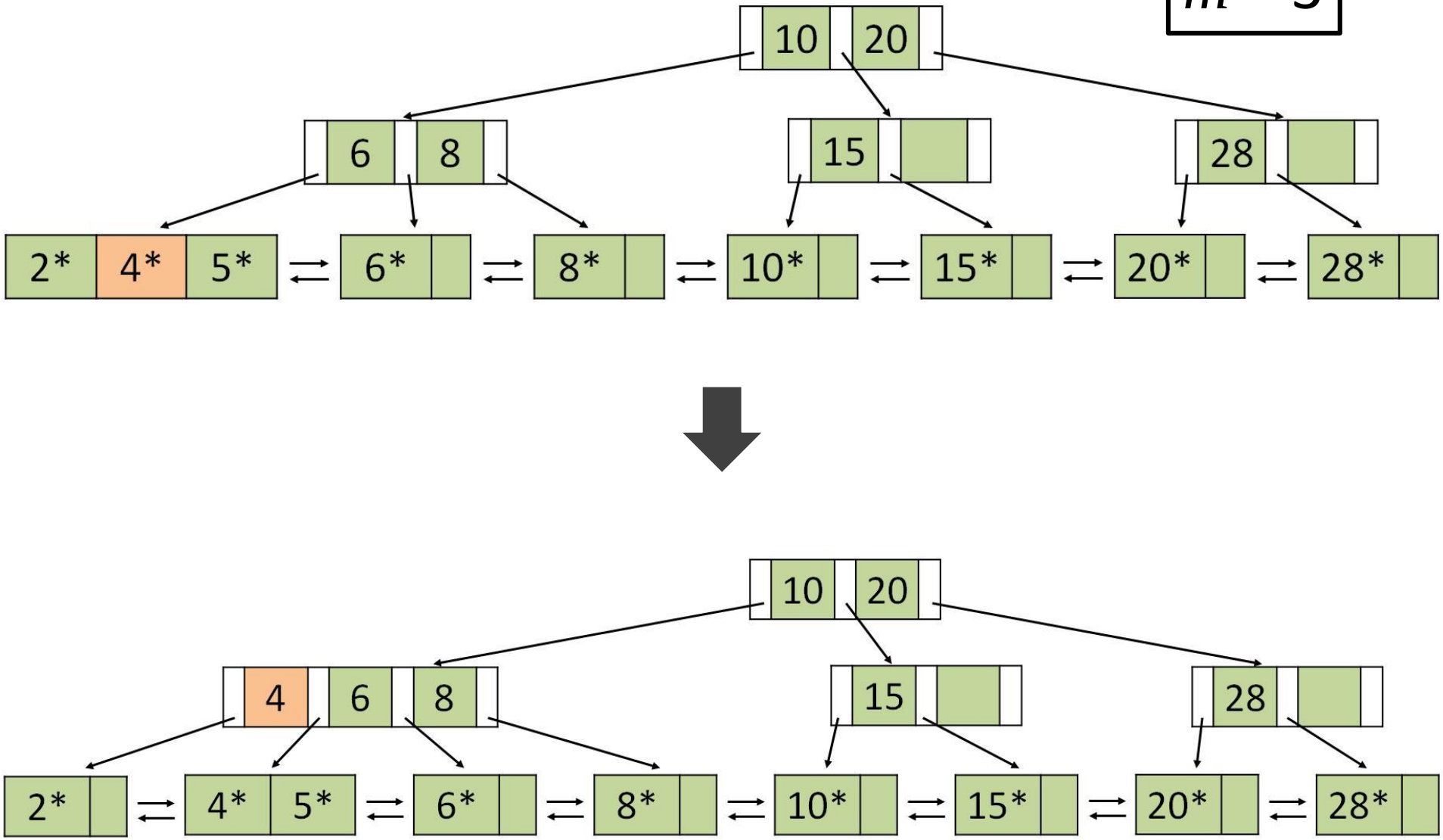
- Insert 4 in the given B+ Tree (order = 3).



Order m = 3	Minimum	Maximum
#Keys	Root: 1 Internal: ceil(3/2) – 1 = 1 Leaf: ceil((3–1)/2) = 1	2
#Childs	ceil(3/2) = 2	3

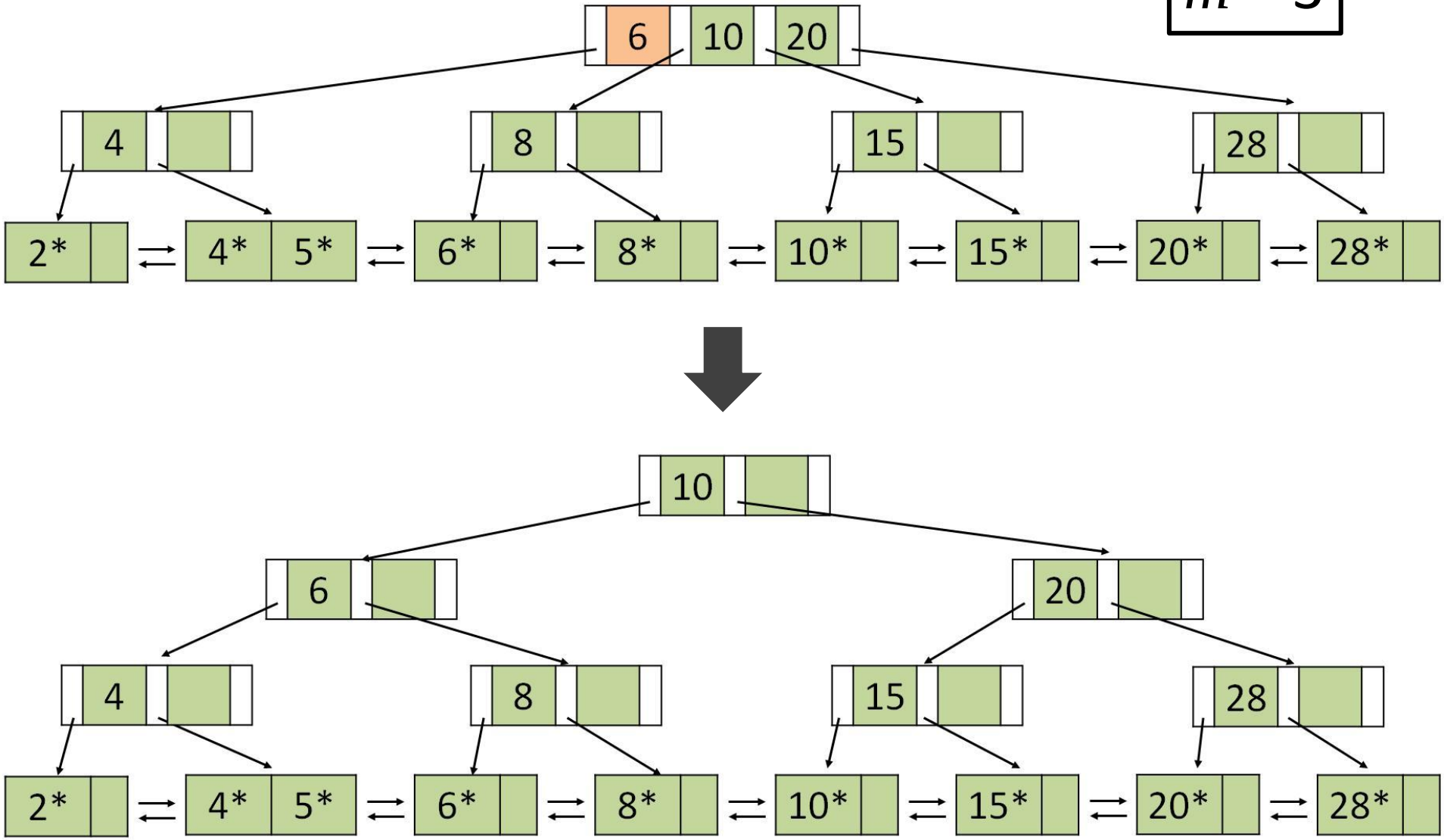
Contd…

$m = 3$



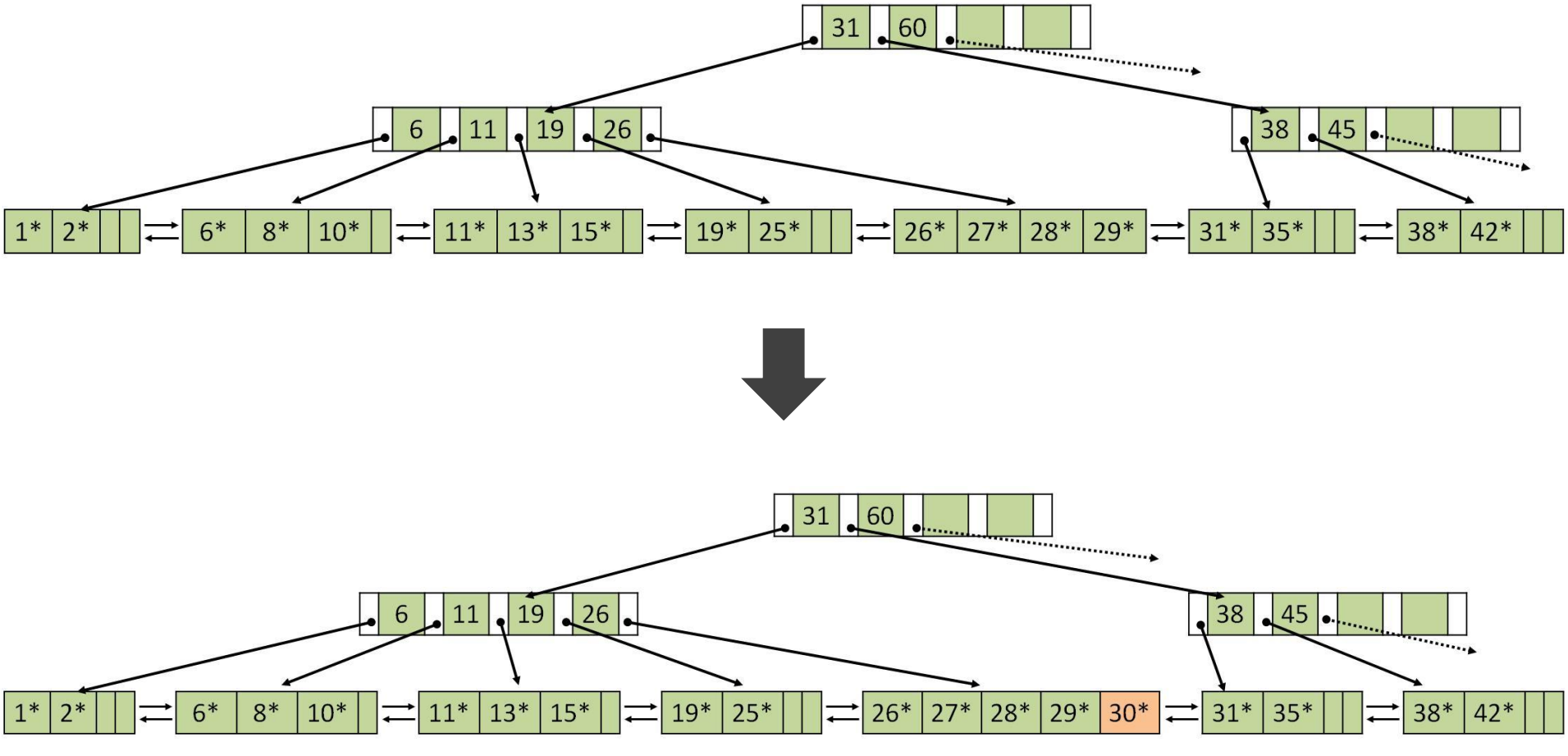
Contd…

$m = 3$



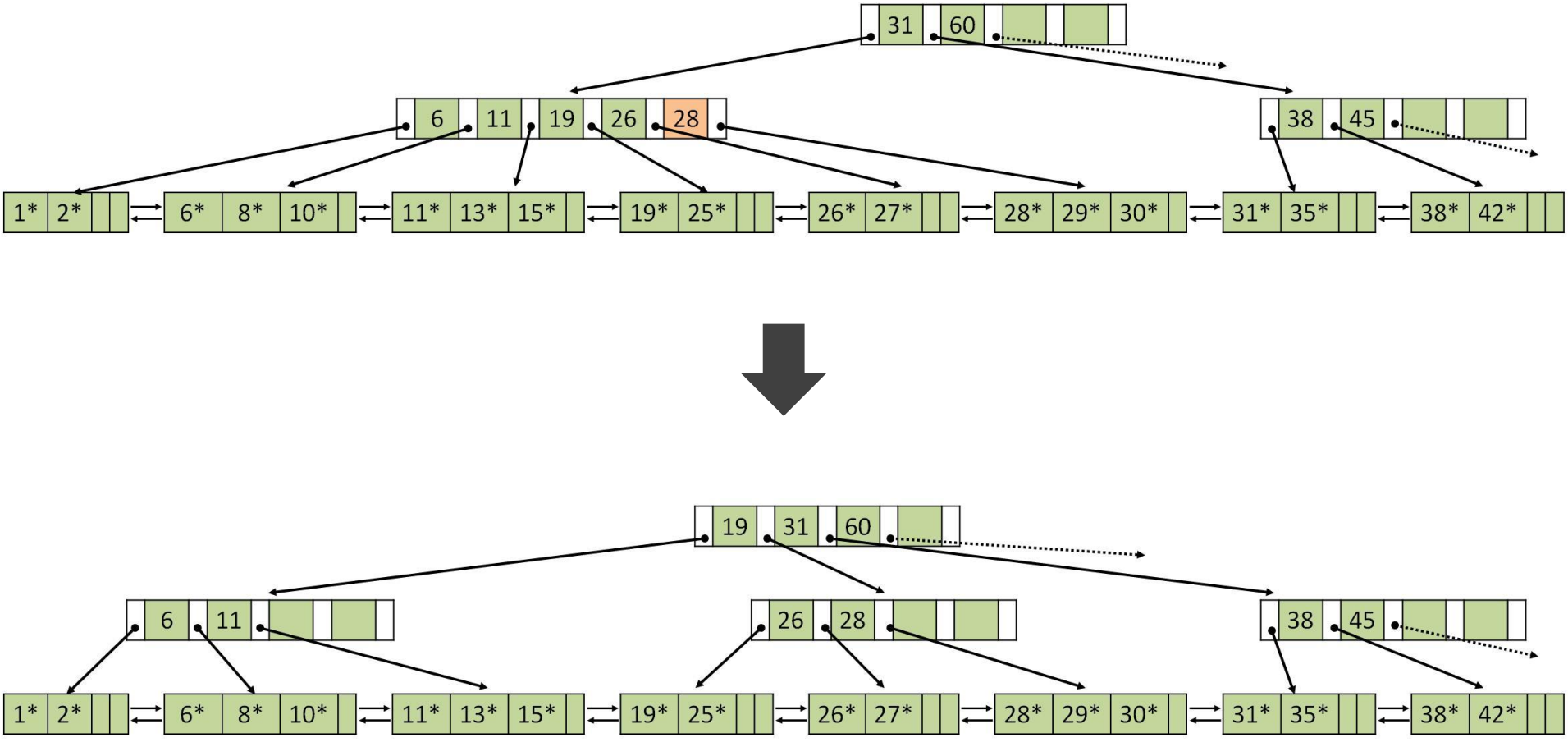
Insert 30

$m = 5$


31
60
6
11
19
26
38
45
1*
2*
6*
8*
10*
11*
13*
15*
19*
25*
26*
27*
28*
29*
31*
35*
38*
42*
31
60
6
11
19
26
38
45
1*
2*
6*
8*
10*
11*
13*
15*
19*
25*
26*
27*
28*
29*
30*
31*
35*
38*
42*

Contd…

$m = 5$

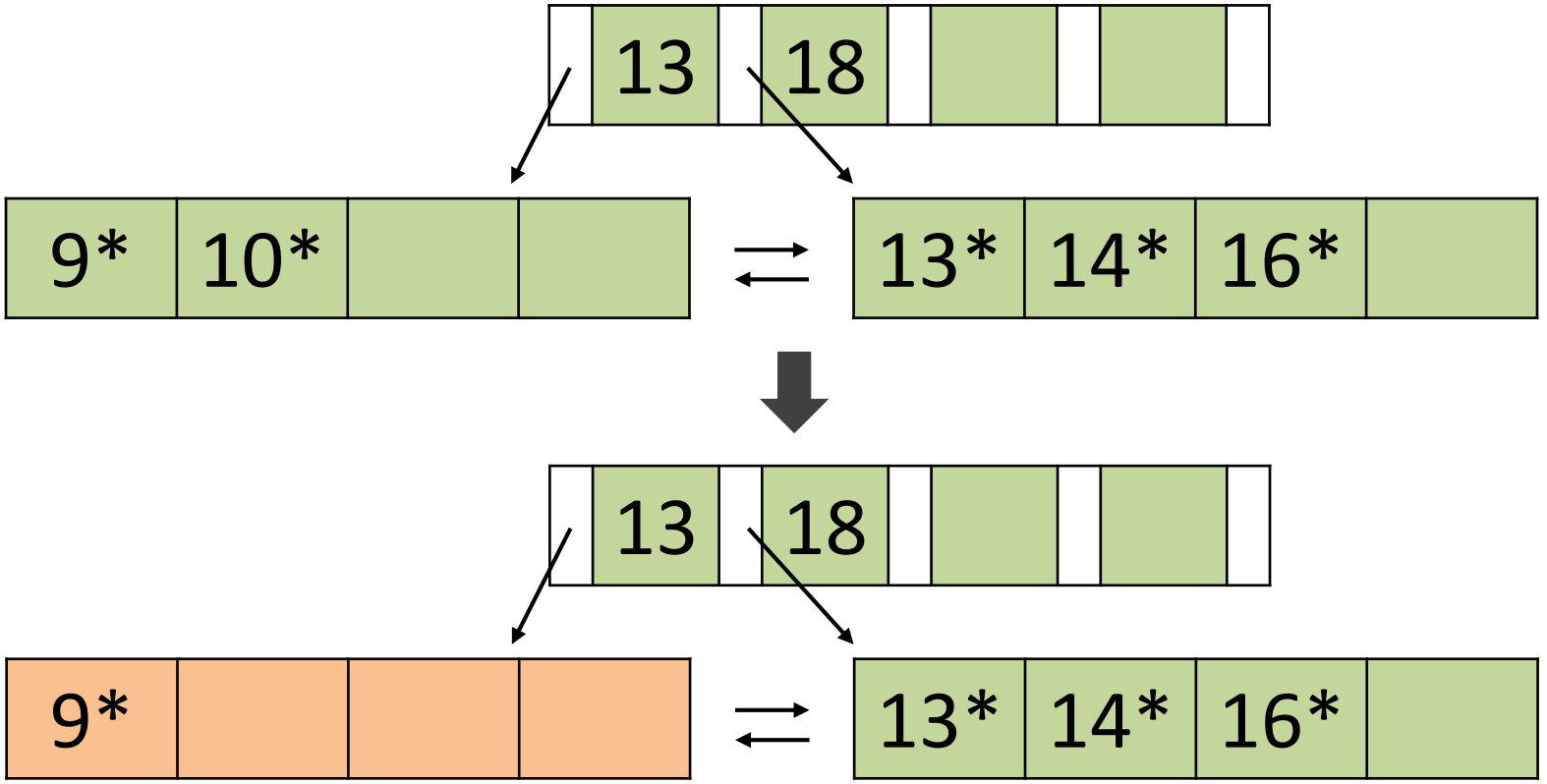

31
60
6
11
19
26
28
38
45
1*
2*
6*
8*
10*
11*
13*
15*
19*
25*
26*
27*
28*
29*
30*
31*
35*
38*
42*
19
31
60
6
11
26
28
38
45
1*
2*
6*
8*
10*
11*
13*
15*
19*
25*
26*
27*
28*
29*
30*
31*
35*
38*
42*

Deletion

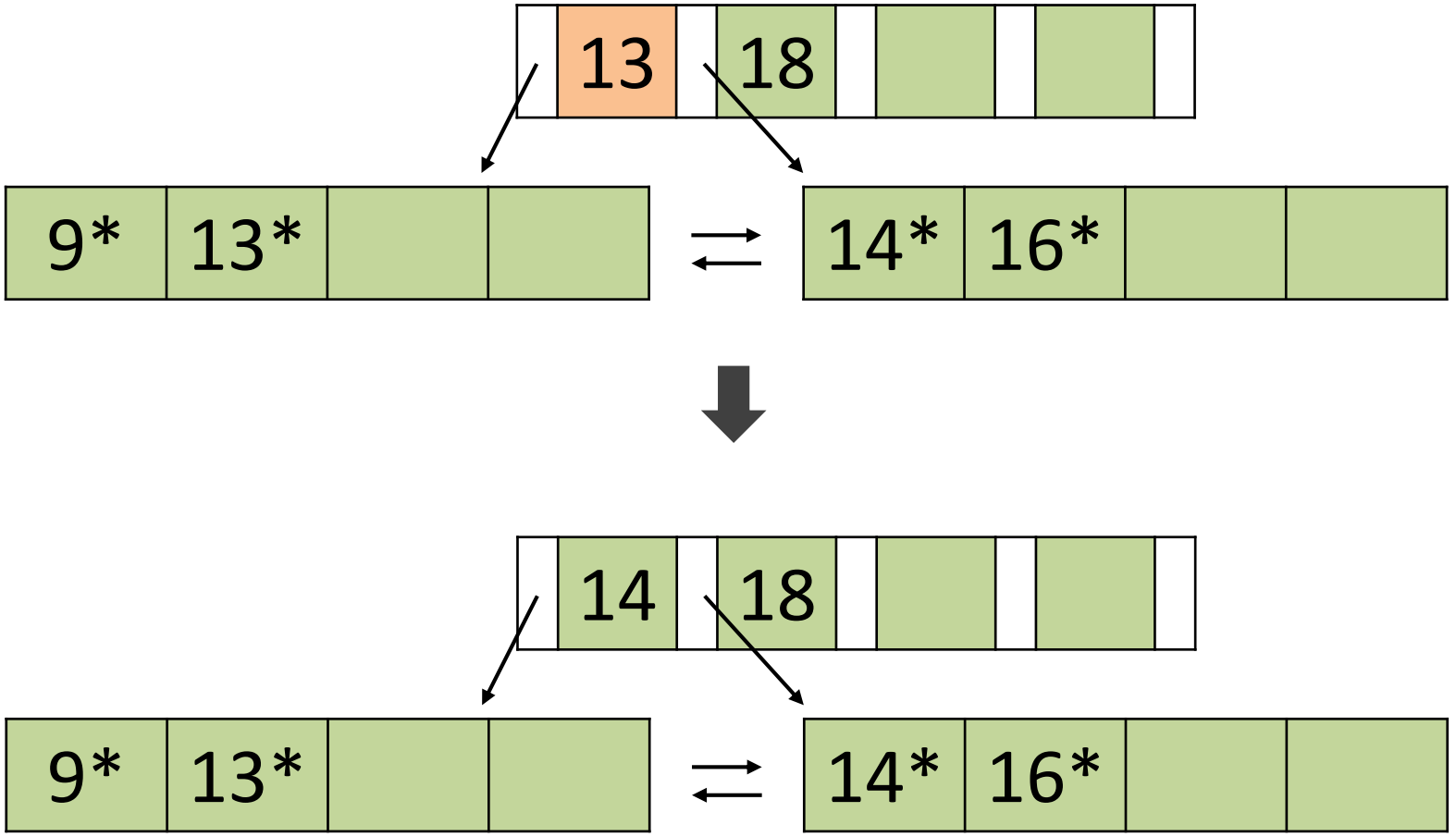
- Actual values are present only at the leaf nodes as internal nodes are index nodes only, so again deletion will take place at the leaf.
- Use post merge.
  - First delete the key,
  - Then check for underflow.
- How to handle an underflow?
  - Depends on the type of a node being underflow.
    - Leaf node, or
    - Internal node.

Underflow – Leaf Node (1)

- Sibling has keys, so borrow.
- Example: Delete 10 from a B+ tree with $\boldsymbol{m} = \mathbf{5}$.

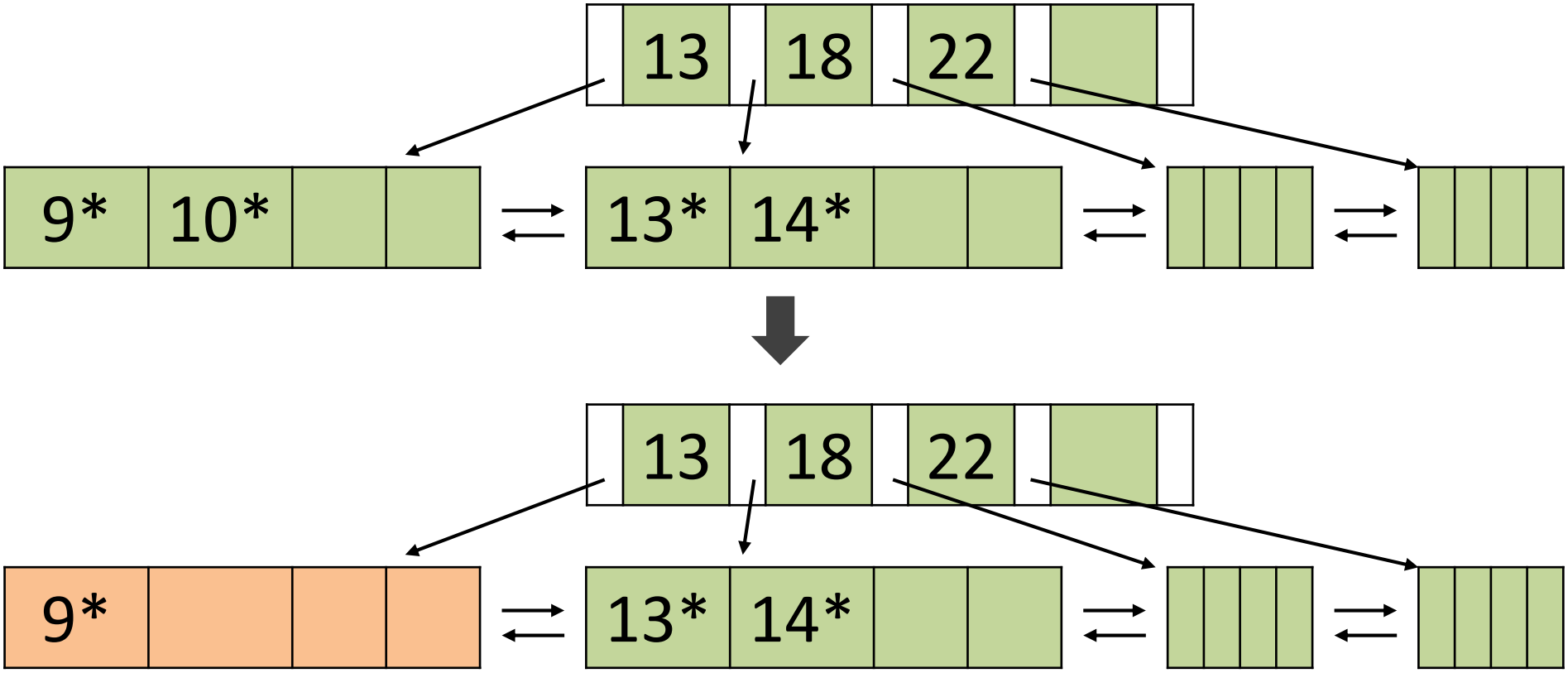


Contd…

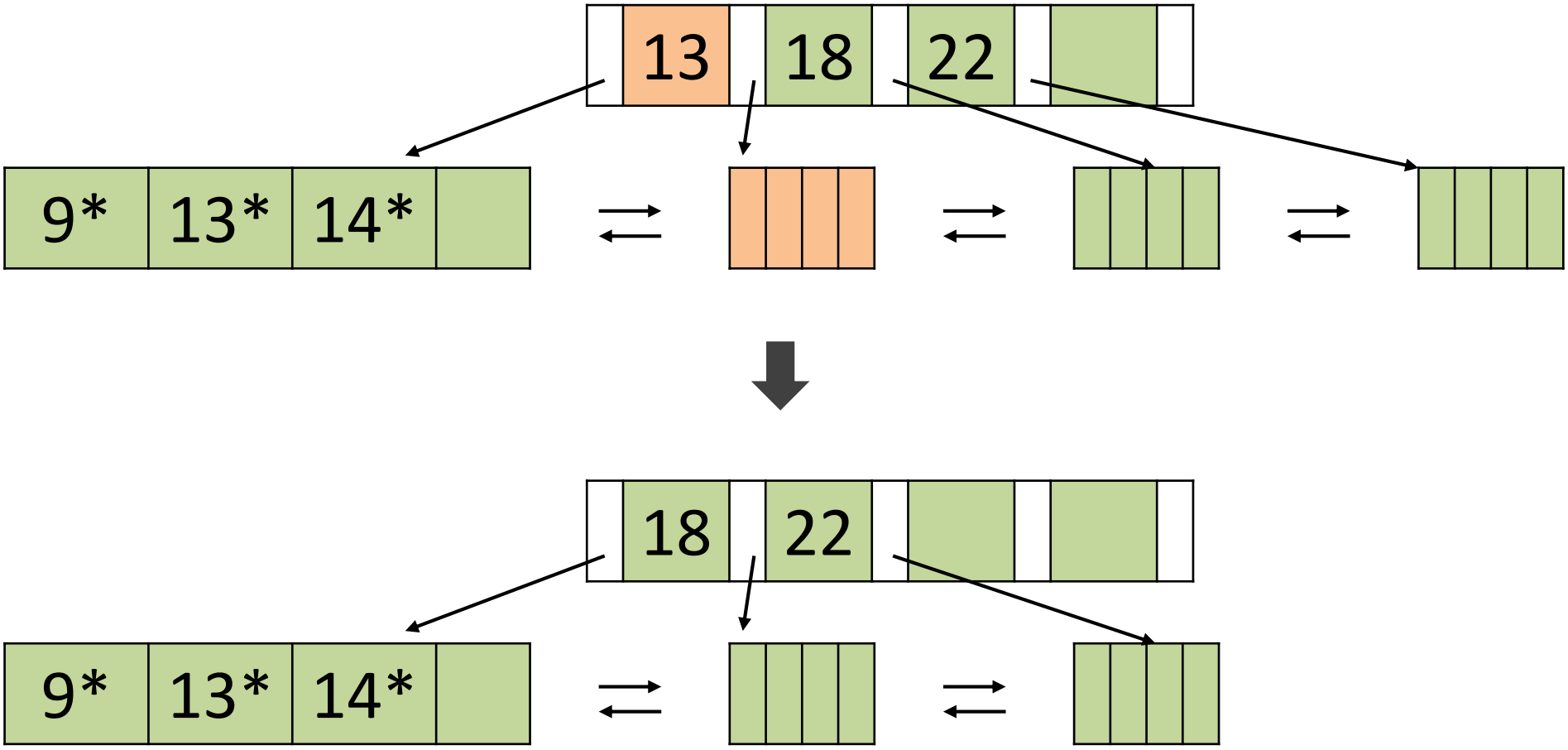


Underflow – Leaf Node (2)

- Sibling doesn't has keys, so merge and discard parent.
- Example: Delete 10 from a B+ tree with $\boldsymbol{m} = 5$.

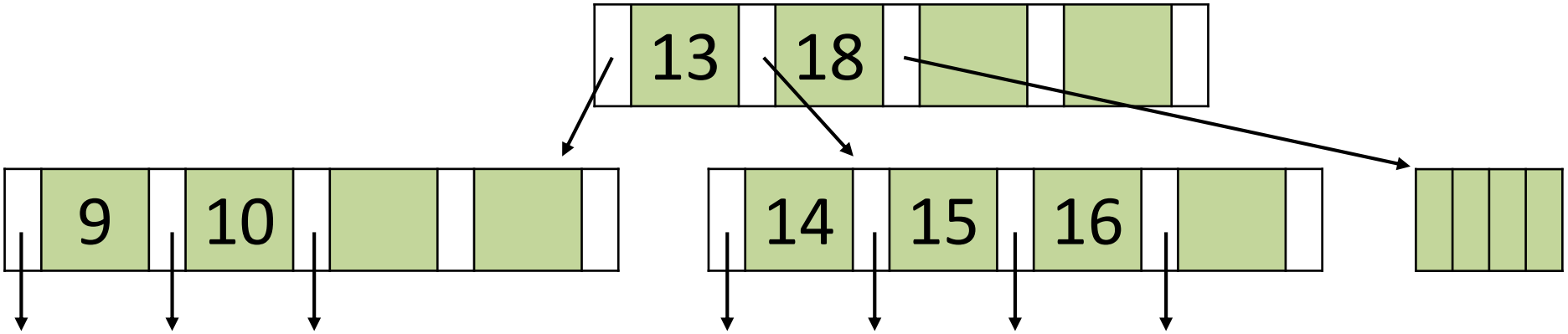


Contd…

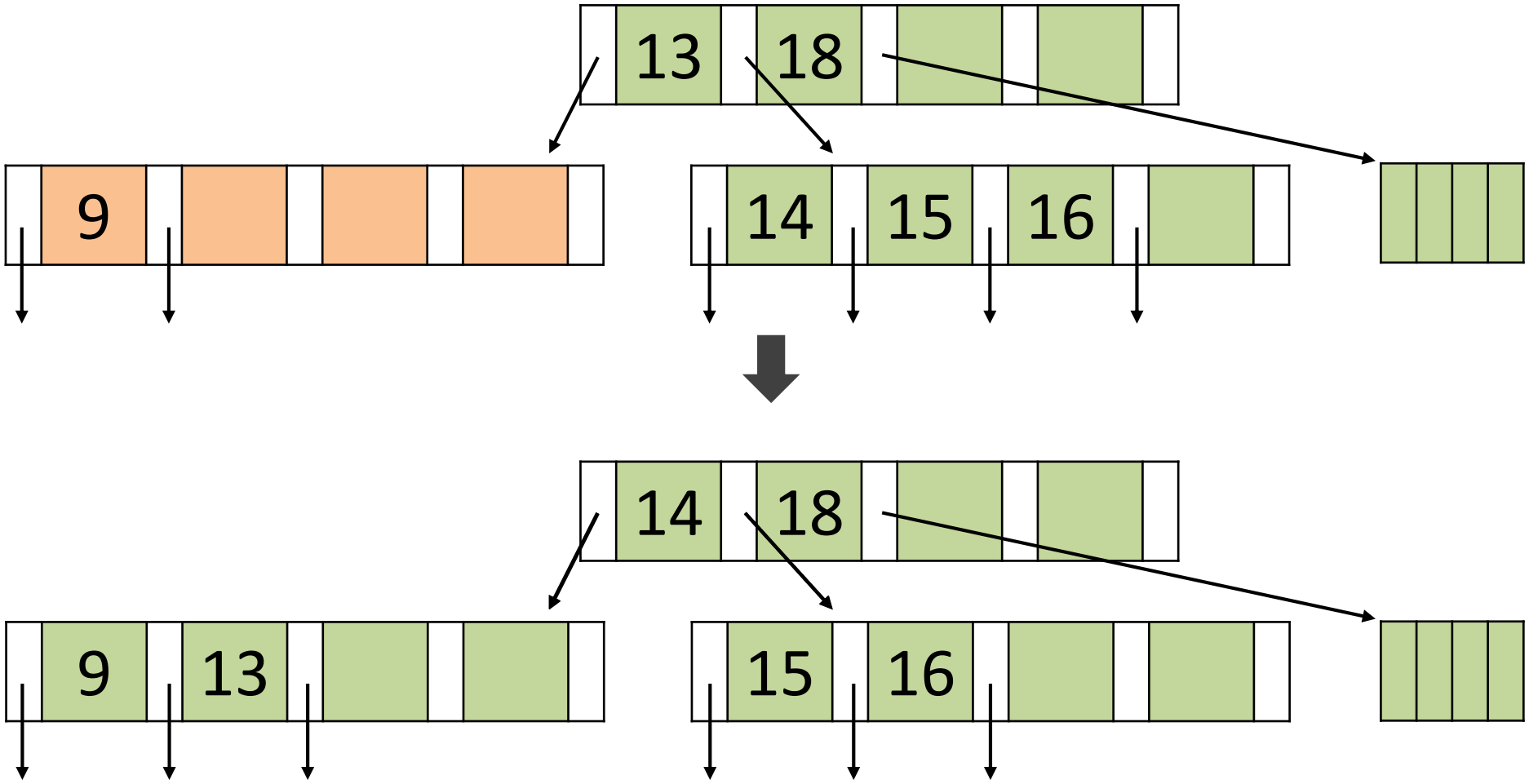


Underflow – Internal Node (1)

- Sibling has keys, so borrow.
- Example: Delete 10 from a B+ tree with $\boldsymbol{m} = \mathbf{5}$.

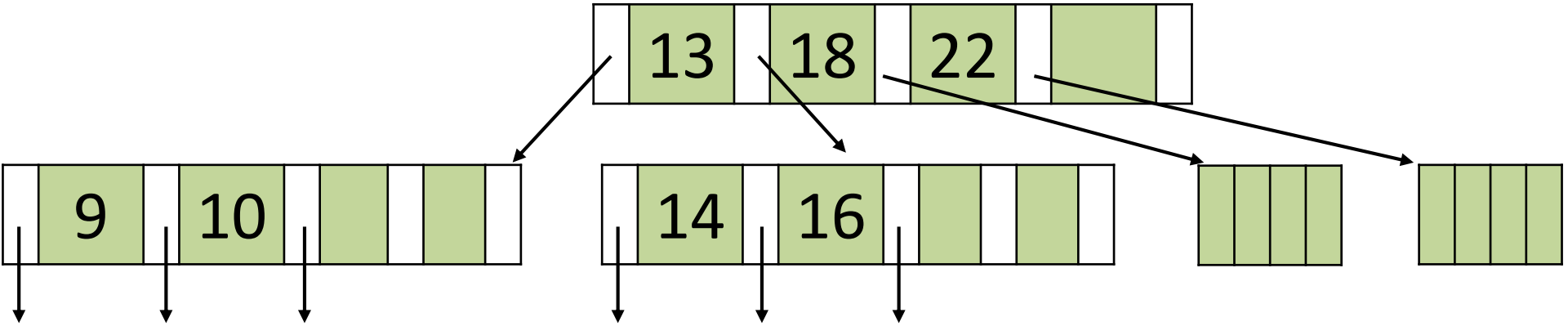


Contd…

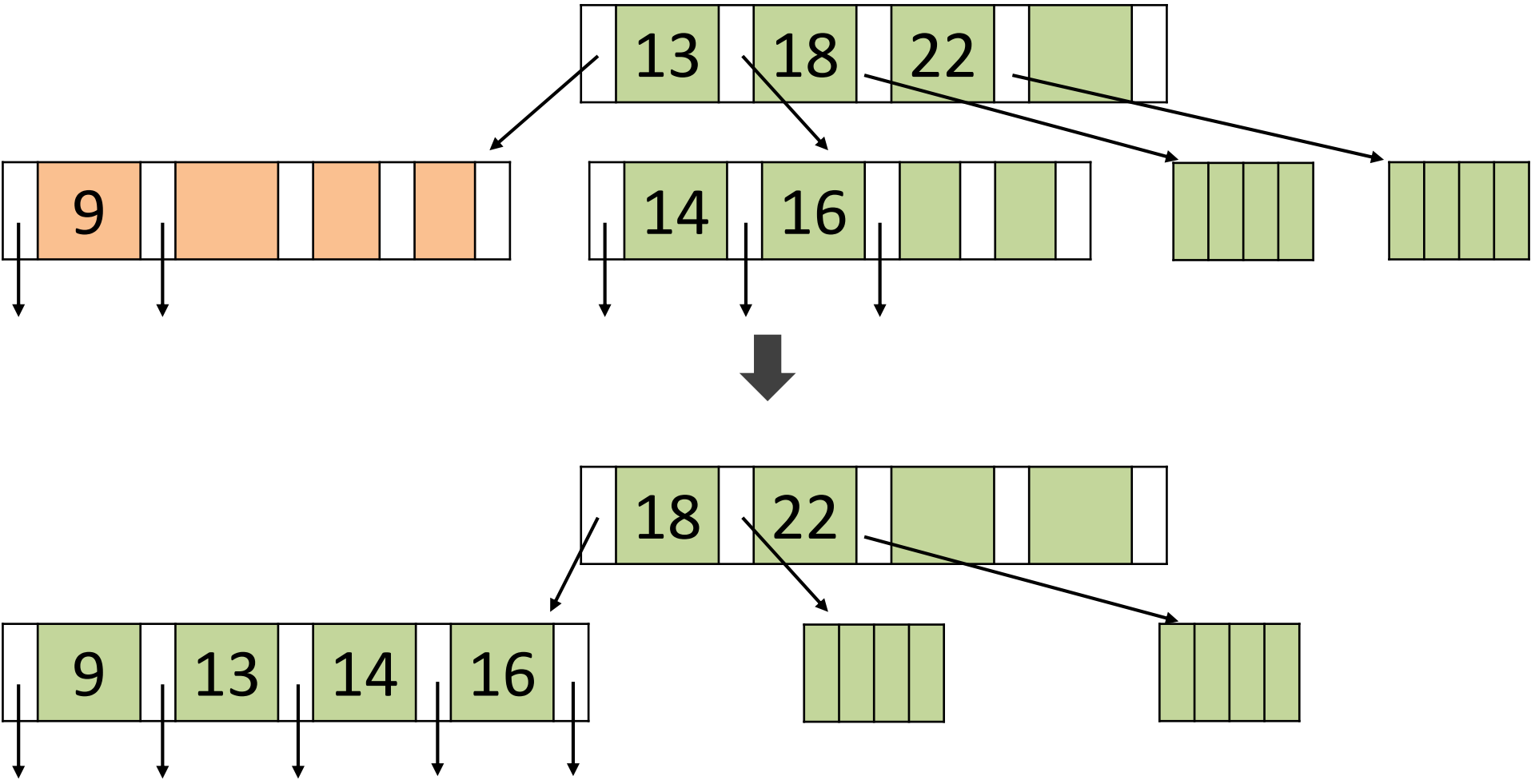


Underflow – Internal Node (2)

- Sibling doesn't has keys, so merge and keep the parent.
- Example: Delete 10 from a B+ tree with $\boldsymbol{m} = \mathbf{5}$.

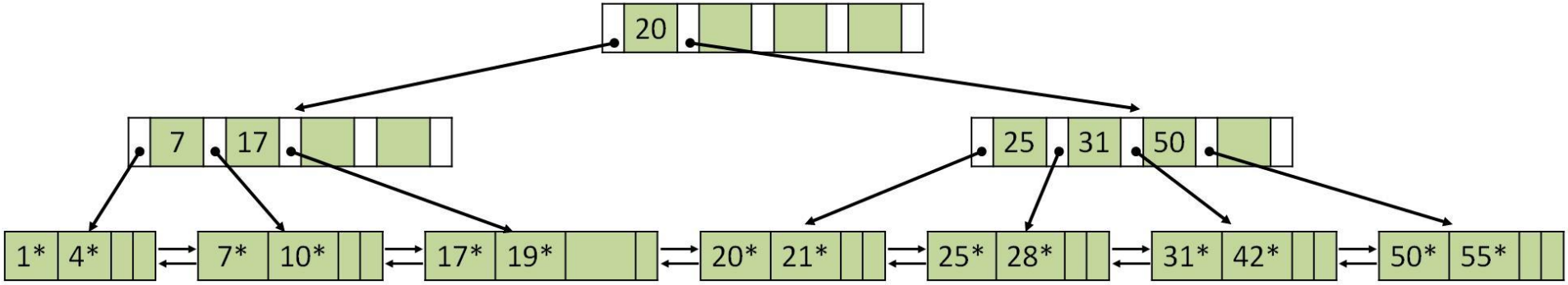


Contd…



Example – Delete 28, 31, 21, 25, 19.

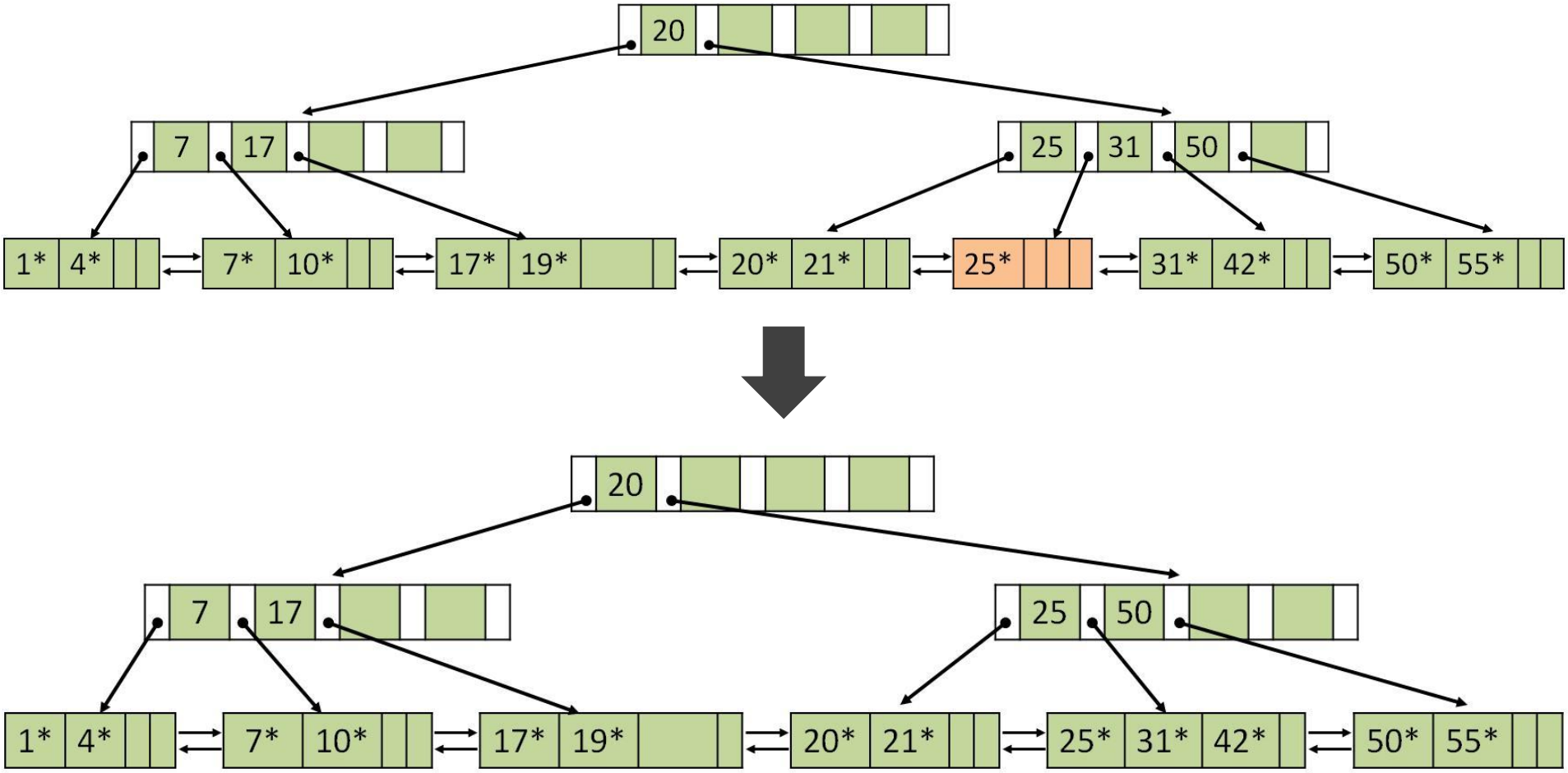
$m = 5$



Contd...

- Delete 28

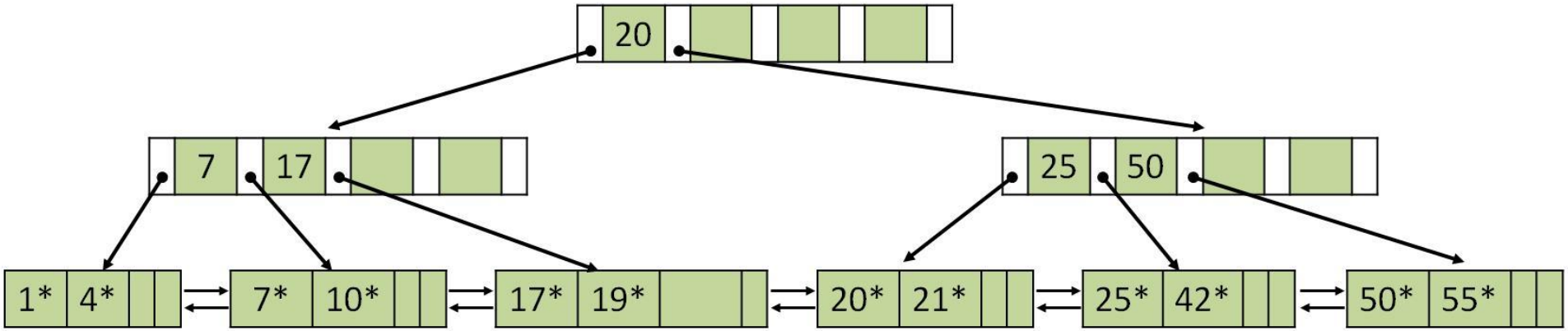
$m = 5$



Contd...

- Delete 31

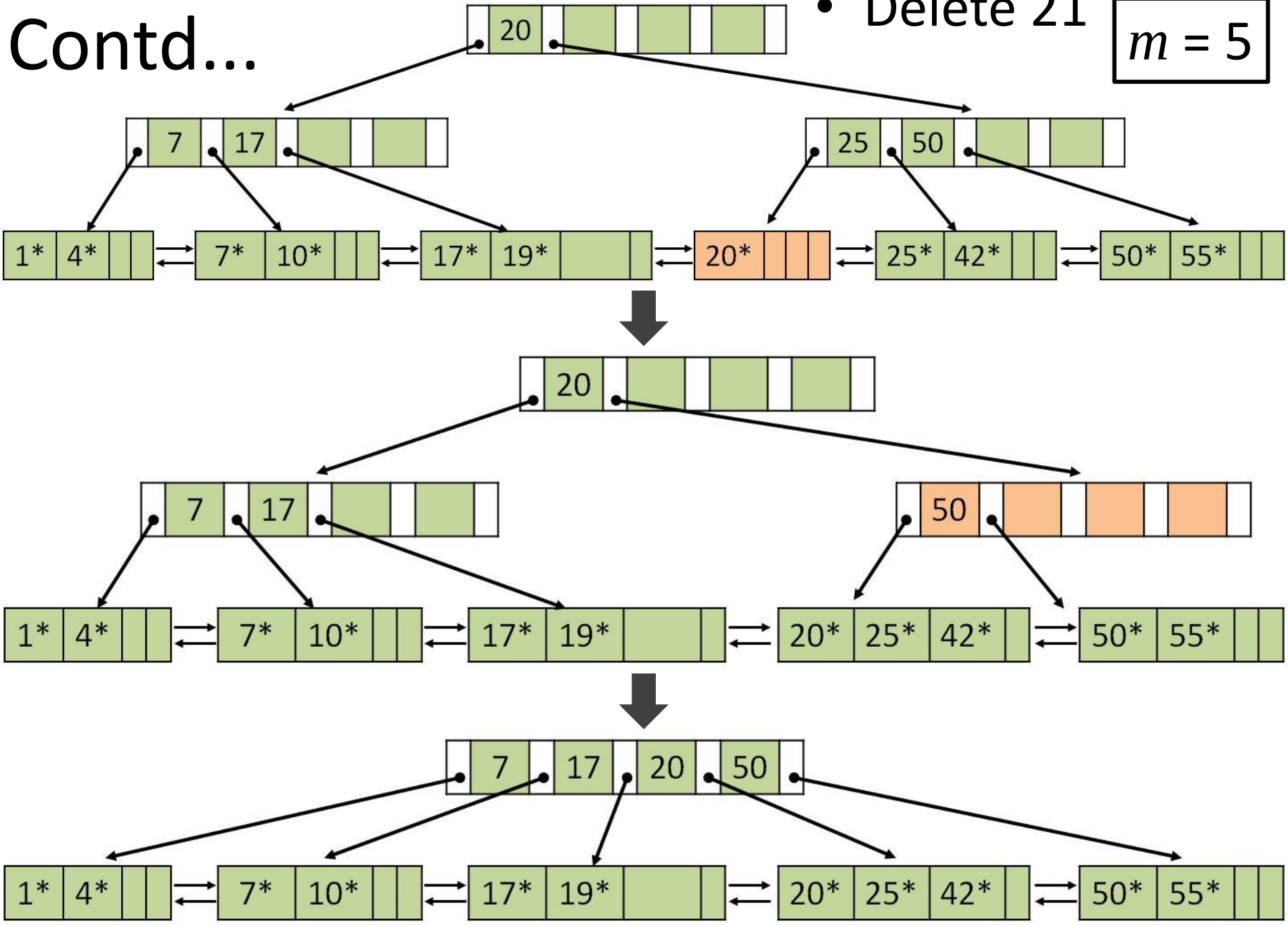
$m = 5$



Contd...

- Delete 21

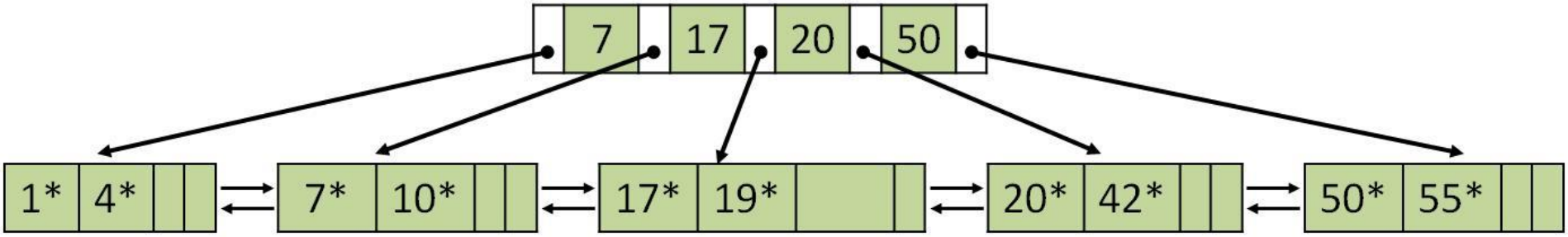
$m = 5$



Contd...

- Delete 25

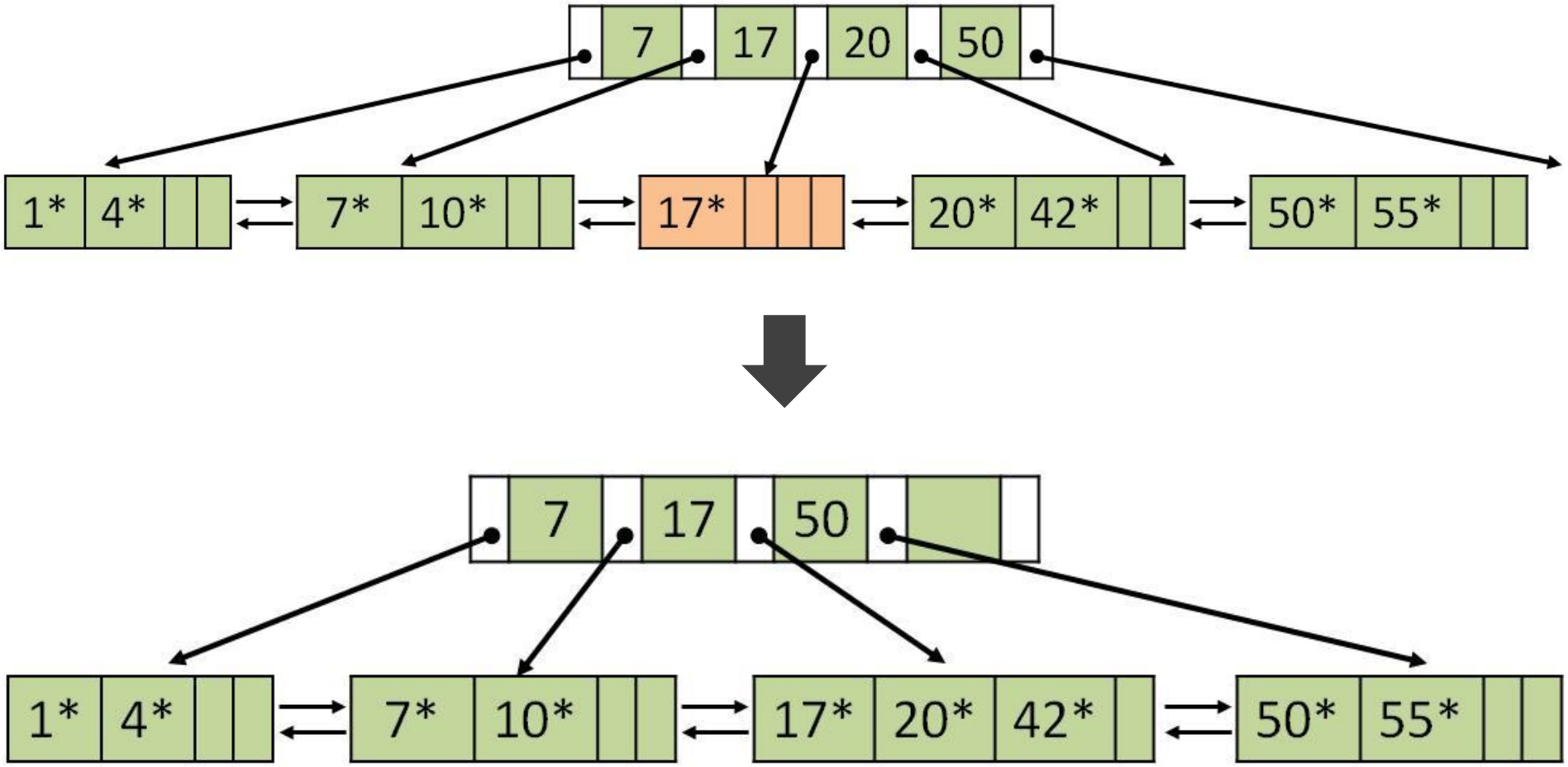
$m = 5$



Contd...

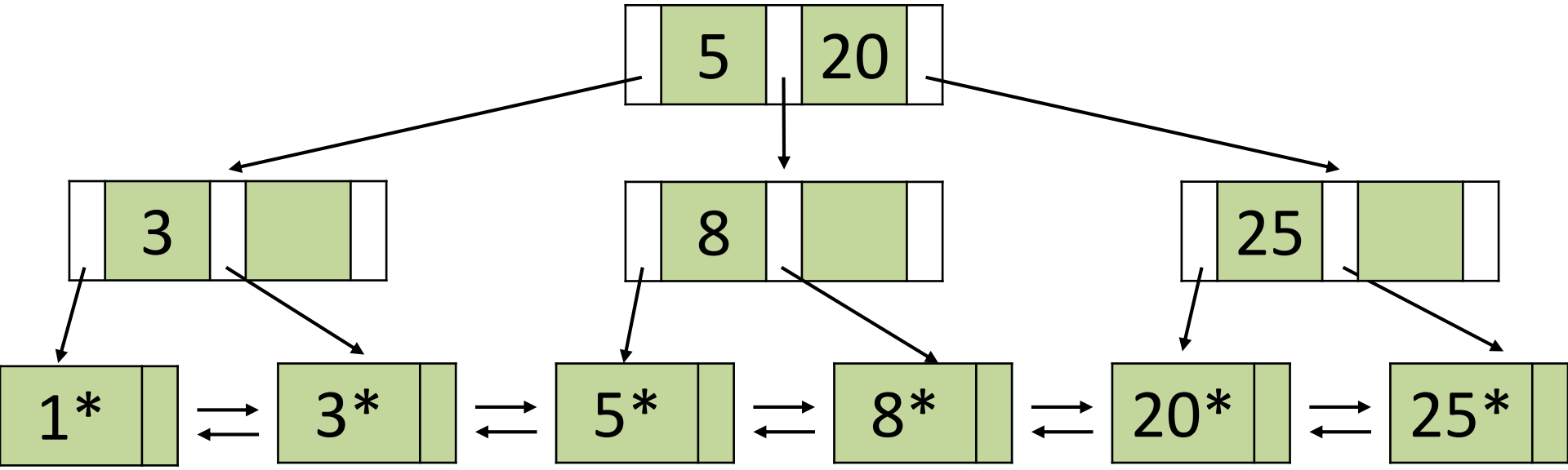
- Delete 19

$m = 5$

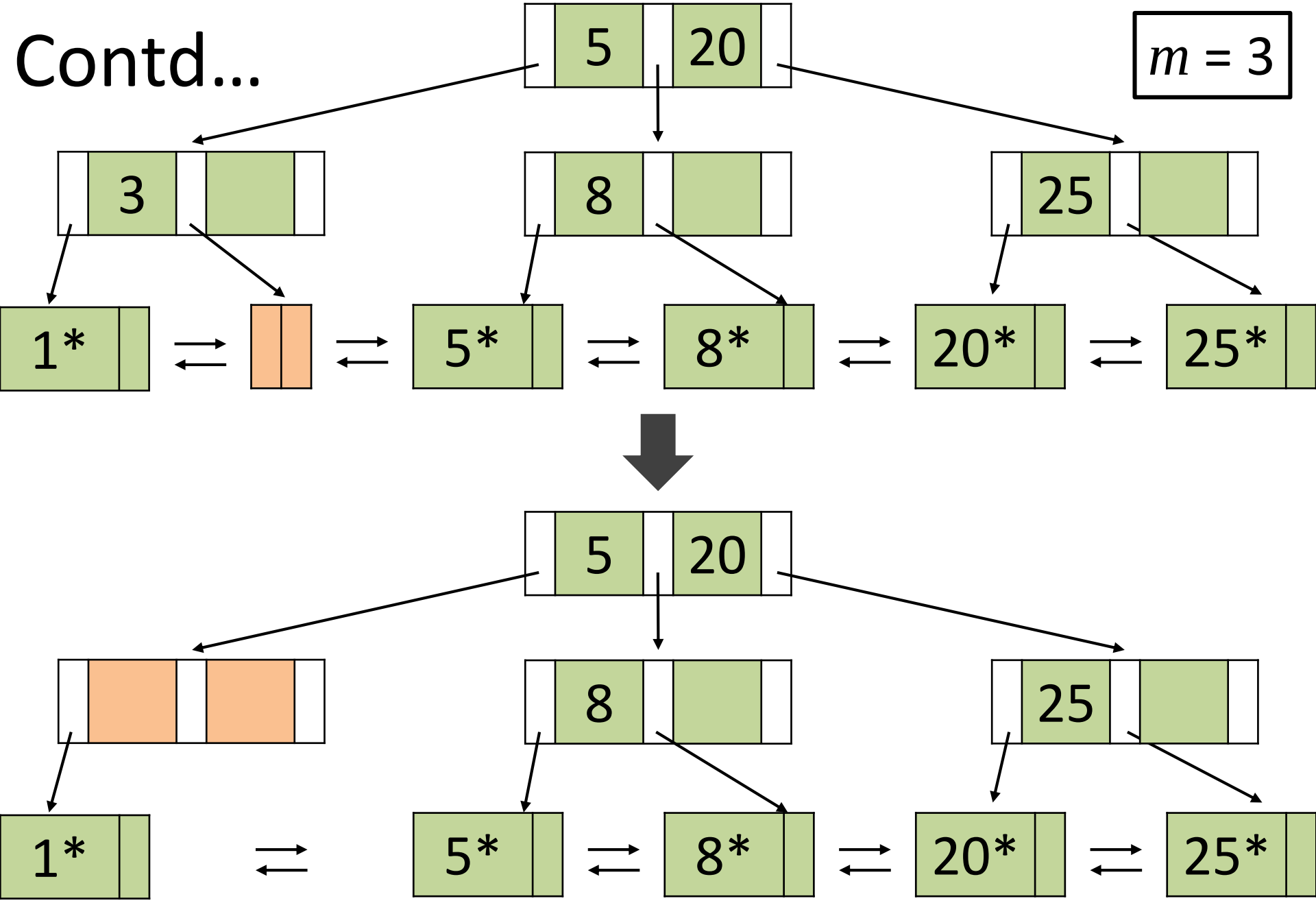


Example – Delete 3.

$m = 3$

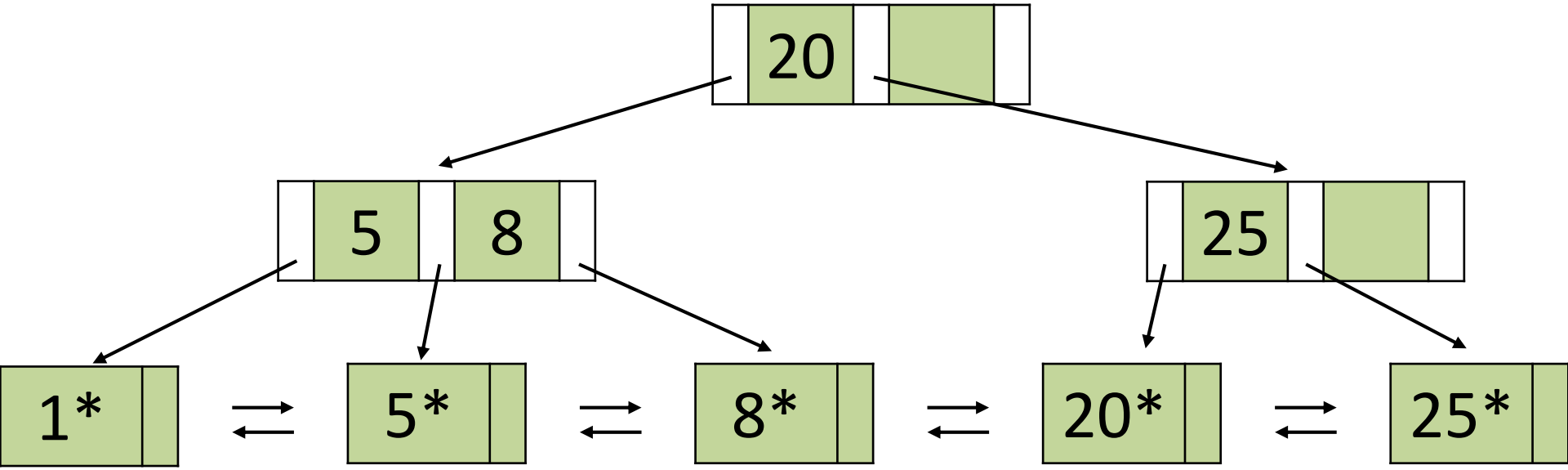


Contd…



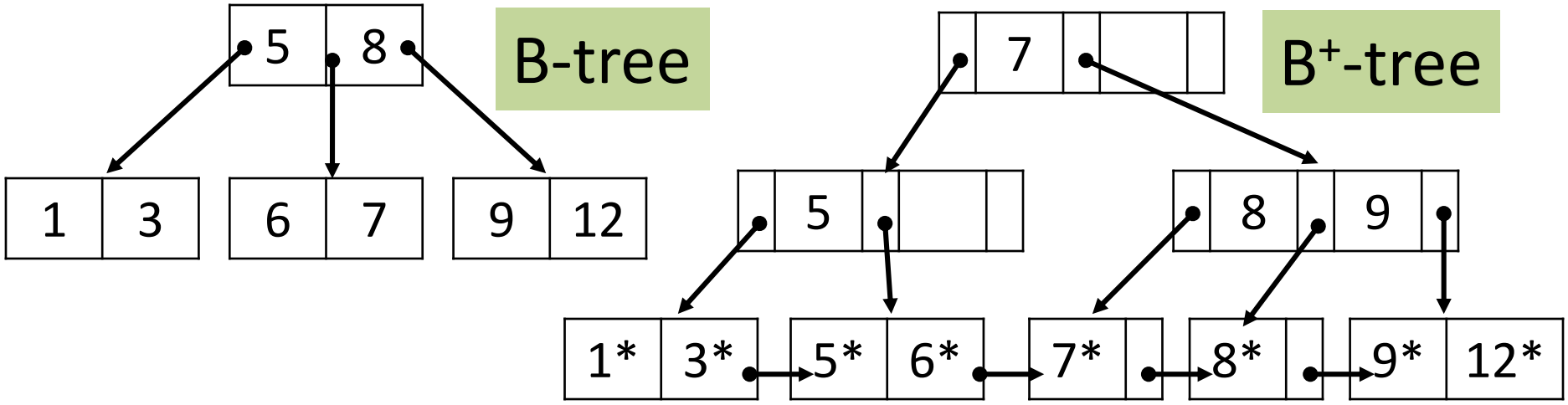
Contd…

$m = 3$



Construct B-Tree and B$^+$-Tree of order 3

- 8, 5, 1, 7, 3, 12, 9, 6.
- At most 2 keys and 3 pointers.
- At least **(ceil(3/2) – 1 = 1)** key and **(ceil(3/2) = 2)** pointers.
- Odd order: use post split for B-tree as well.



Thank You