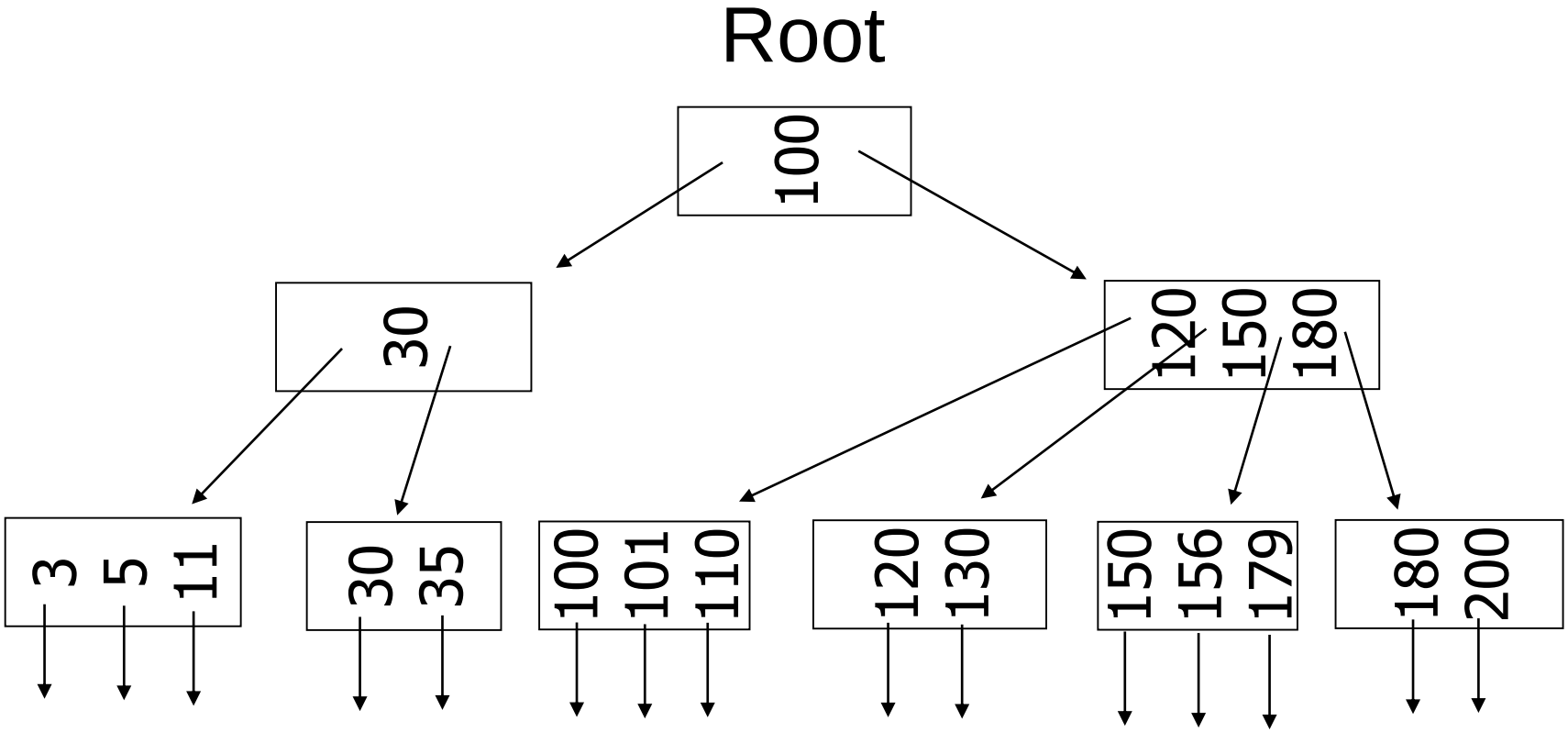
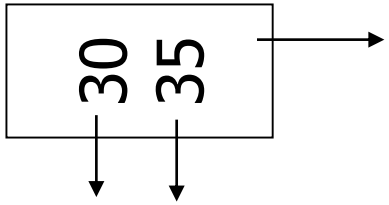
B+Tree Example n=3

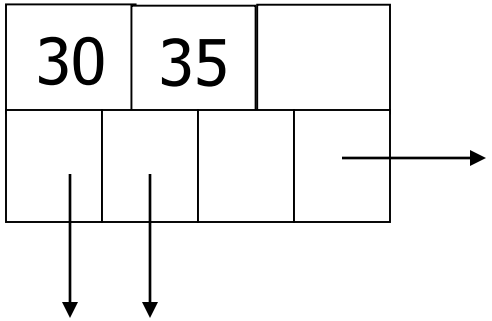


B+ tree in textbook's notation

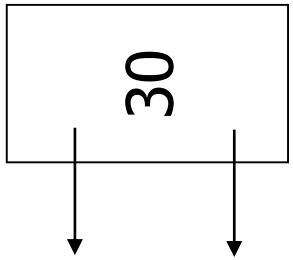
n=3

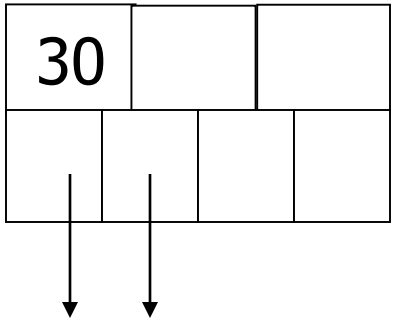
Leaf:





Non-leaf:





Size of nodes:

- n+1 pointers
- n keys

(fixed)

Don’t want nodes to be too empty

- Use at least

  Non-leaf: $\lceil (n+1)/2 \rceil$ pointers

  Leaf: $\lfloor (n+1)/2 \rfloor$ pointers to data

B+tree rules tree of order *n*

(1) All leaves at same lowest level
(balanced tree)

(2) Pointers in leaves point to records
except for “sequence pointer”
(to next leaf)

(3) Number of pointers/keys for B+tree

	Max ptrs	Max keys	Min ptrs→data	Min keys
Non-leaf (non-root)	n+1	n	$\lceil (n+1)/2 \rceil$	$\lceil (n+1)/2 \rceil - 1$
Leaf (non-root)	n+1	n	$\lfloor (n+1)/2 \rfloor$	$\lfloor (n+1)/2 \rfloor$
Root	n+1	n	1	1

Insert into B+tree

(a) simple case (insert 32)

- space available in leaf

(b) leaf overflow (insert 7)

(c) non-leaf overflow (insert 160)

(d) new root (insert 45)

Deletion from B+tree

(a) Simple case - no example

(b) Coalesce with neighbor (delete 50)

(c) Re-distribute keys (delete 50)

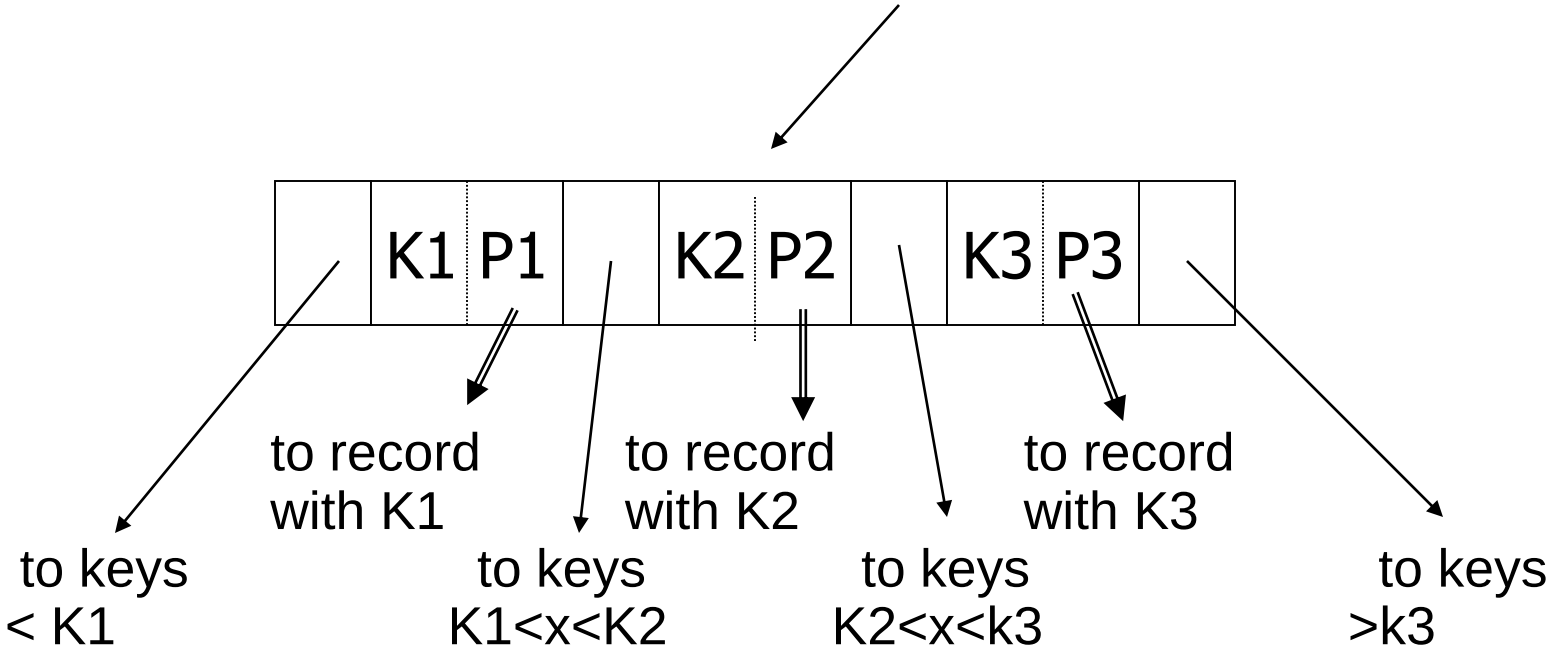
(d) Cases (b) or (c) at non-leaf (delete 37)

B+tree deletions in practice

- Often, coalescing is not implemented
  - Too hard and not worth it!

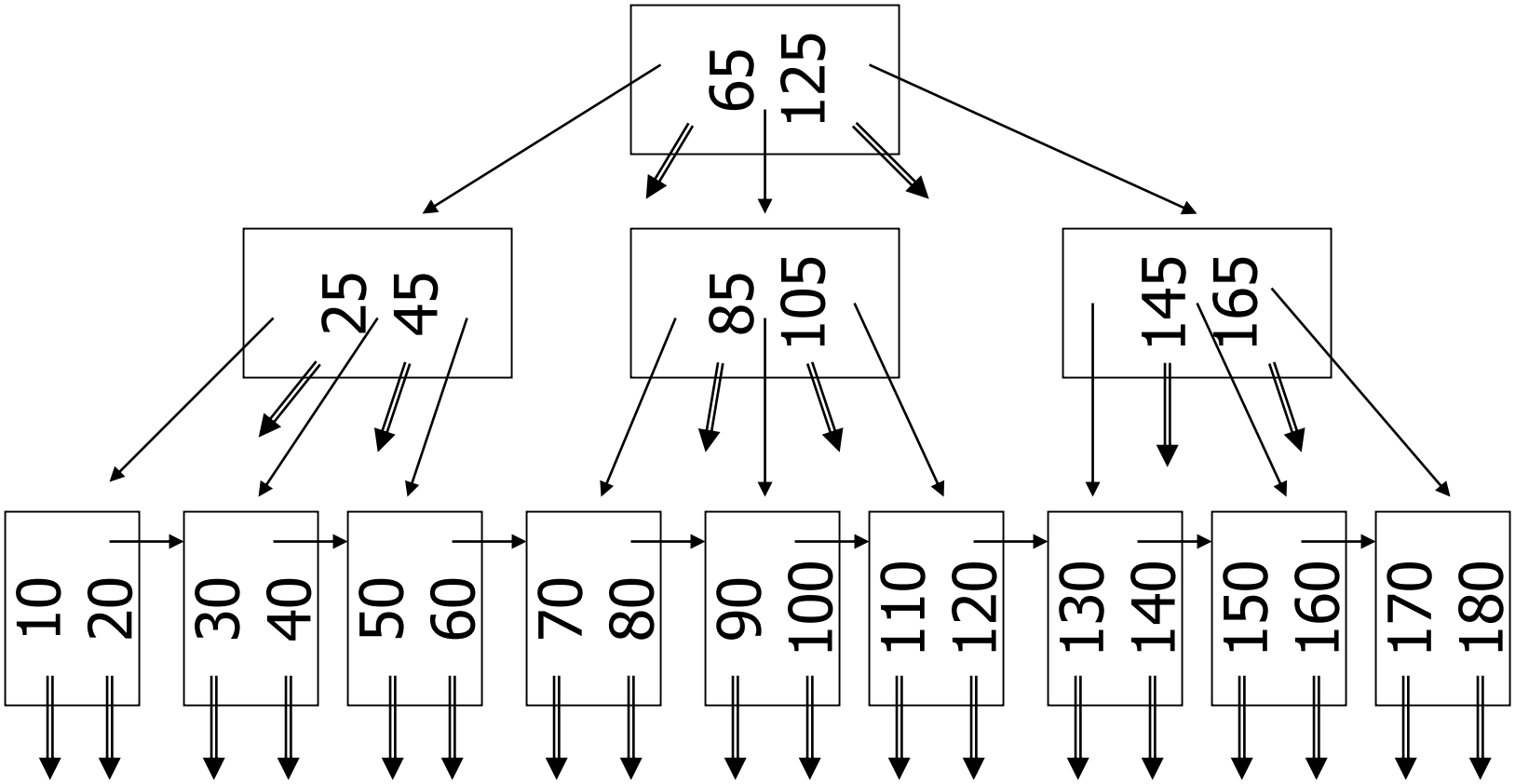
Variation on B+tree: B-tree (no +)

- Idea:
  - Avoid duplicate keys (leaf and non-leaf)
  - Have record pointers in non-leaf nodes


K1
P1
K2
P2
K3
P3
to record with K1
to record with K2
to record with K3
to keys < K1
to keys K1<x<K2
to keys K2<x<k3
to keys >k3

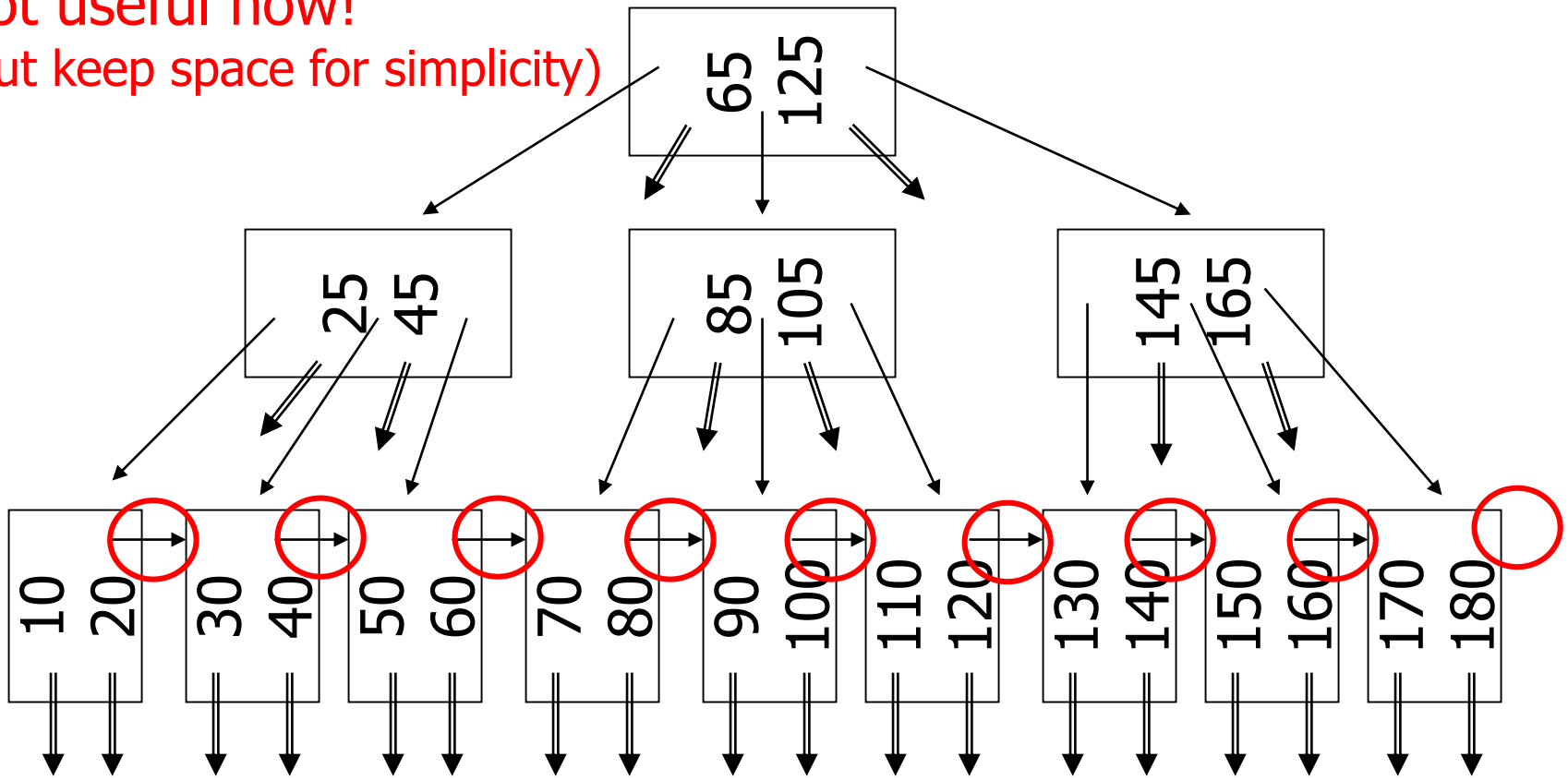
B-tree example

n=2



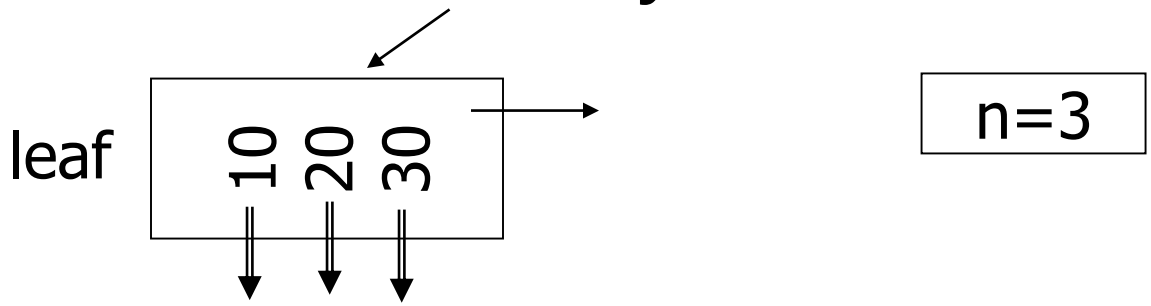
B-tree example n=2

- sequence pointers not useful now!
  (but keep space for simplicity)



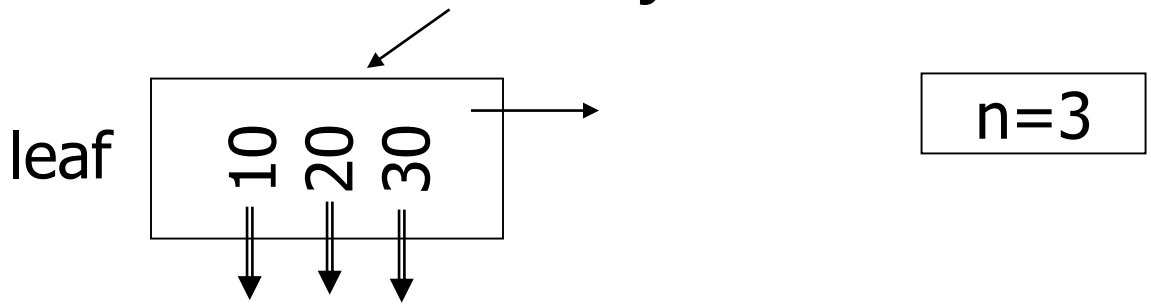
Note on inserts

- Say we insert record with key = 25



Note on inserts

- Say we insert record with key = 25



- Afterwards: