CMSC 424 – Database design
Lecture 14
B+-trees
Hashing

Mihai Pop

Administrative

- Project questions?
- HW2 answers
- HW3 postponed till after midterm2

Indexing...recap

- Index – helps find/process records fast (surrogate for sorting the file)
- Dense/sparse index
- Multi-level indexing (inner/outer index)
- Clustering/non-clustering index
- Primary/secondary index

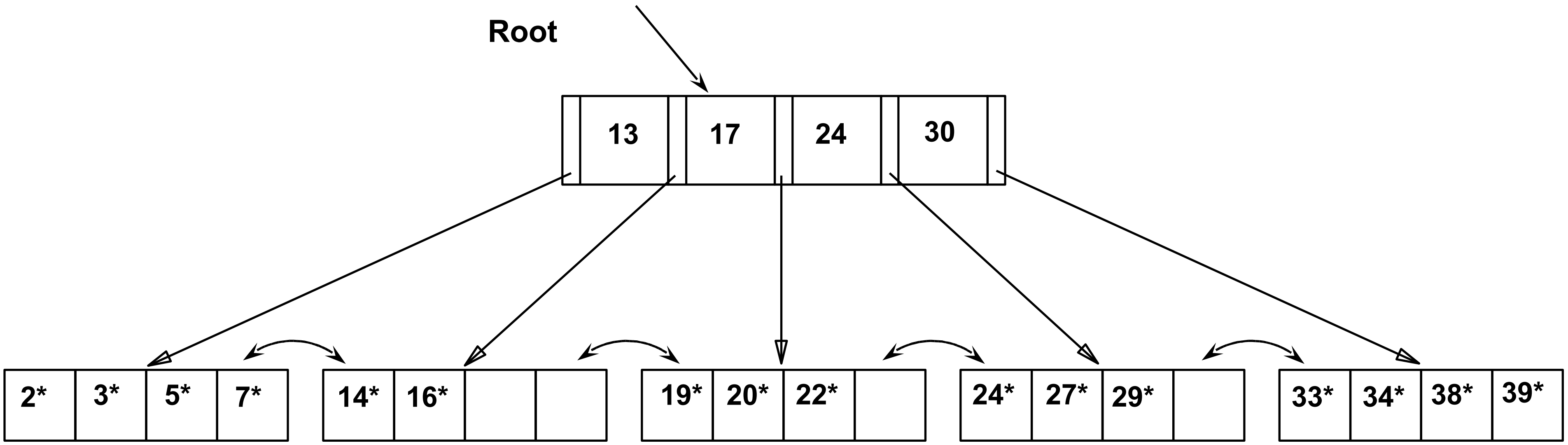
- Key elements:
  - speed of access
  - space overhead
  - speed/ease of insertion/deletion
  - access type (find exact, find range, etc.)
- Many insertion/deletions may lead to inefficient structure – indices may need to be rebuilt to improve performance

B+-trees

- A variant of multi-level indexing
- Extension of binary search tree concept
- Optimize I/O efficiency – node size = disk block size
- Balanced tree structure – all leaves are equidistant from root

Example B+ Tree

- Search begins at root, and key comparisons direct it to a leaf.
- Search for 5*, 15*, all data entries >= 24* ...



☛ *Based on the search for 15*, we <u>know</u> it is not in the tree!*

B+ Tree - Properties

- *Balanced*
- Every node *except root* must be at least ½ full.
- *Order*: the minimum number of keys/pointers in a non-leaf node
- *Fanout* of a node: the number of pointers out of the node

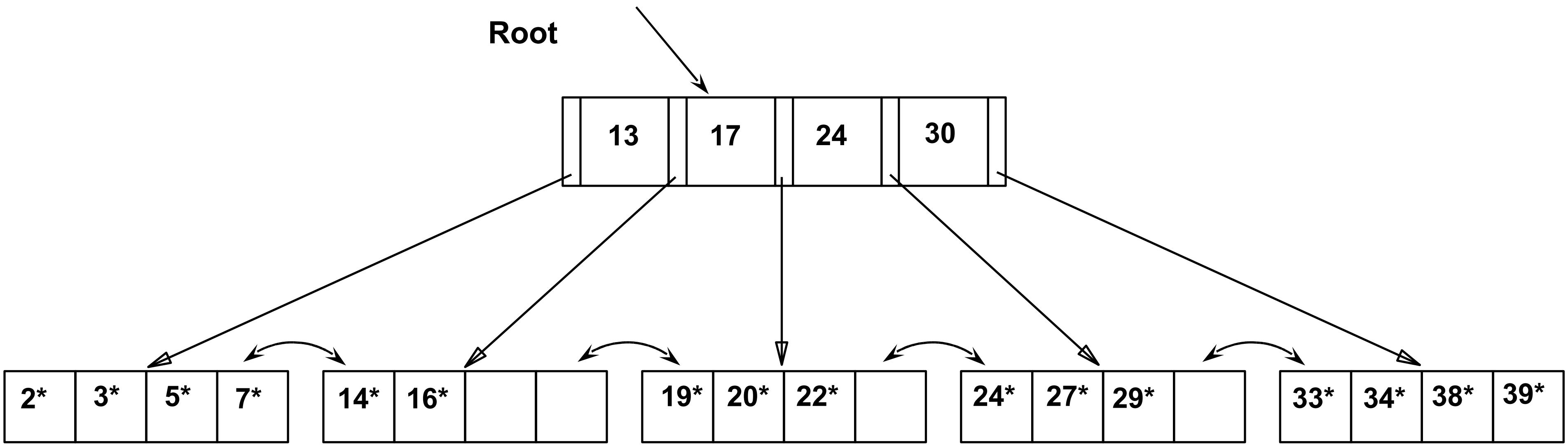
B+ Trees in Practice

- Typical order: 100. Typical fill-factor: 67%.
  - average fanout = 133
- Typical capacities:
  - Height 3: 133^3 = 2,352,637 entries
  - Height 4: 133^4 = 312,900,700 entries
- Can often hold top levels in buffer pool:
  - Level 1 = 1 page = 8 Kbytes
  - Level 2 = 133 pages = 1 Mbyte
  - Level 3 = 17,689 pages = 133 MBytes

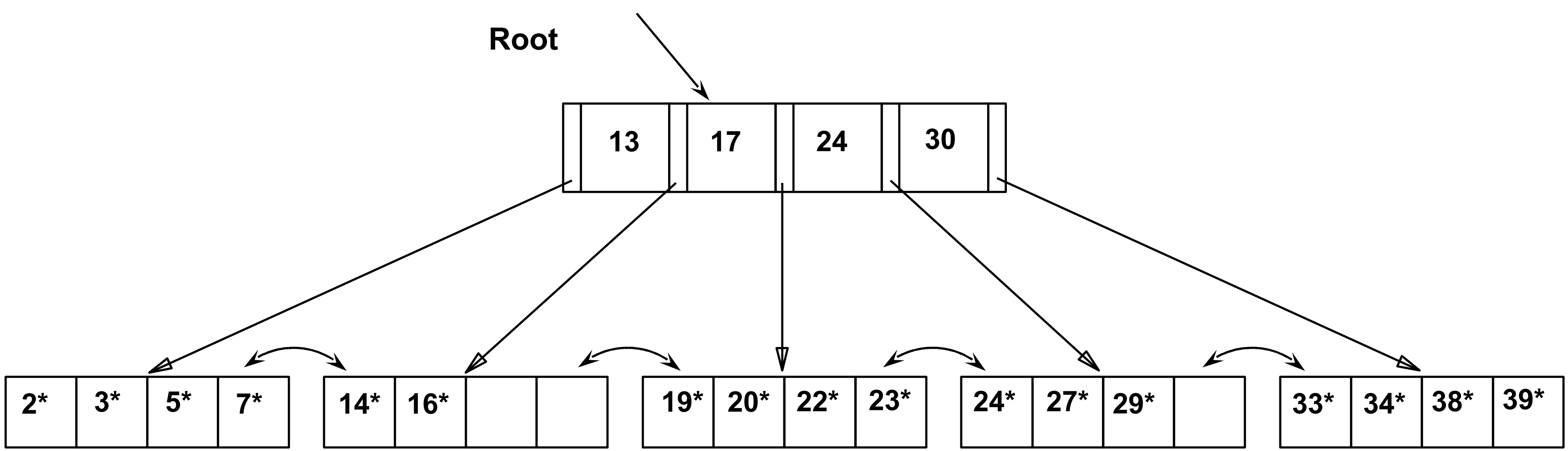
B+ Trees: Summary

- Searching:
  - $\log_d(n)$ – Where *d* is the order, and *n* is the number of entries
- Insertion:
  - Find the leaf to insert into
  - If full, split the node, and adjust index accordingly
  - Similar cost as searching
- Deletion
  - Find the leaf node
  - Delete
  - May not remain half-full; must adjust the index accordingly

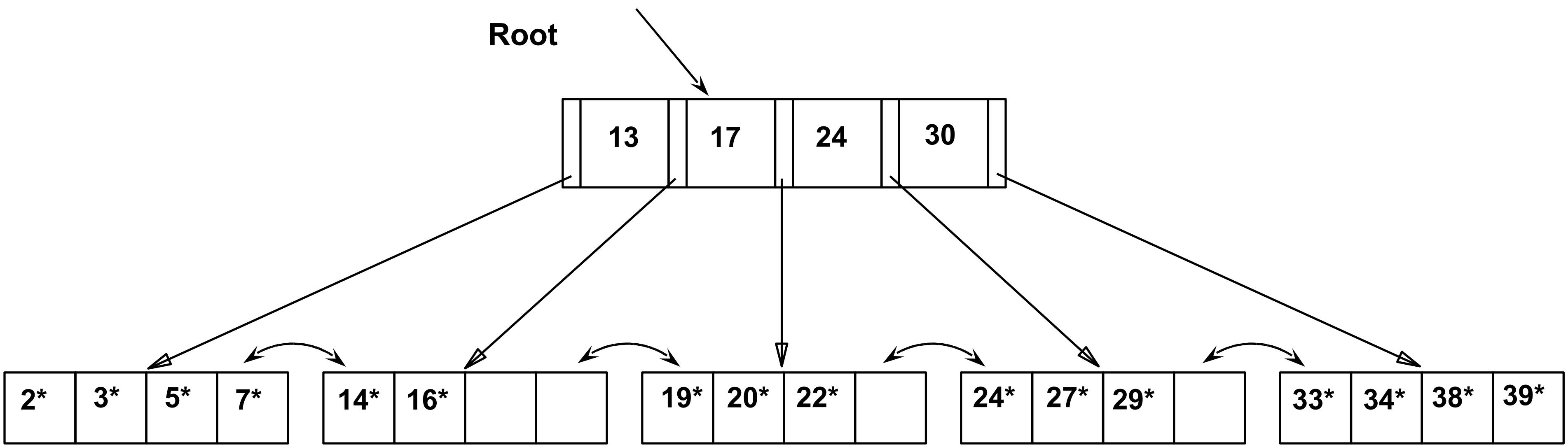
Insert 23*



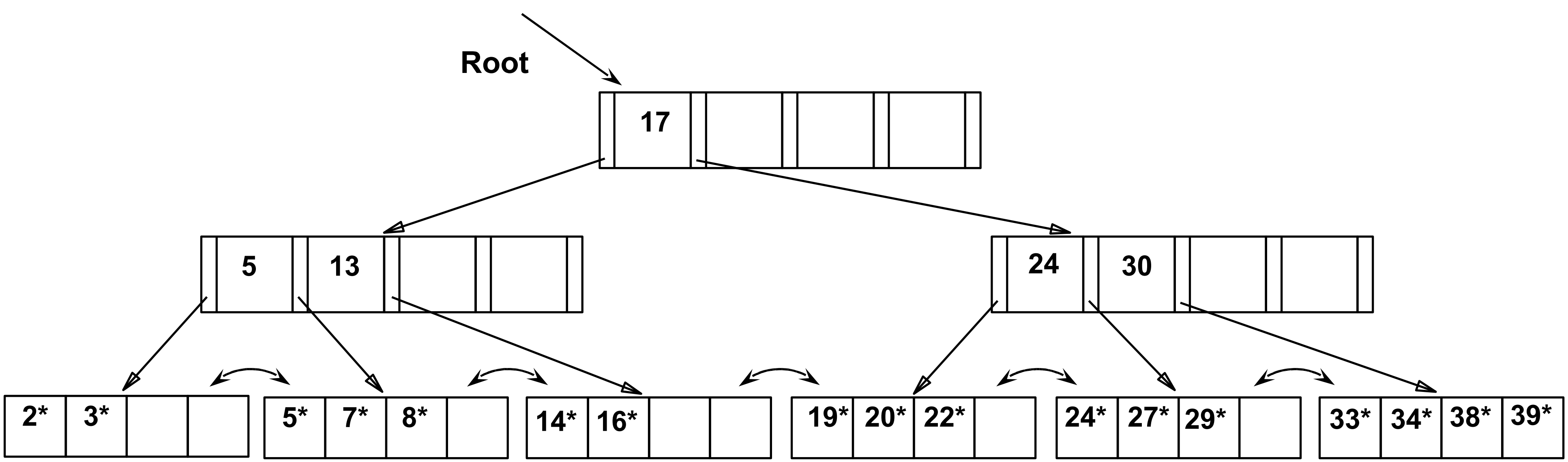
No splitting required.



Insert 8*

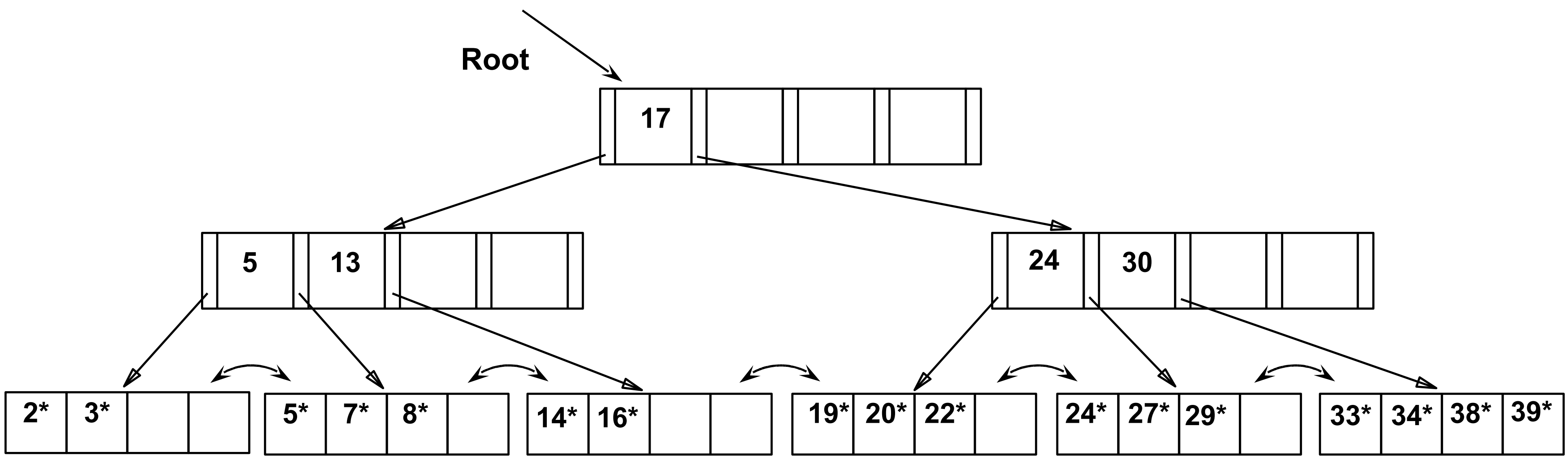


Root
13
17
24
30
2*
3*
5*
7*
14*
16*
19*
20*
22*
24*
27*
29*
33*
34*
38*
39*



Root
17
5
13
24
30
2*
3*
5*
7*
8*
14*
16*
19*
20*
22*
24*
27*
29*
33*
34*
38*
39*

Example B+ Tree - Inserting 8*



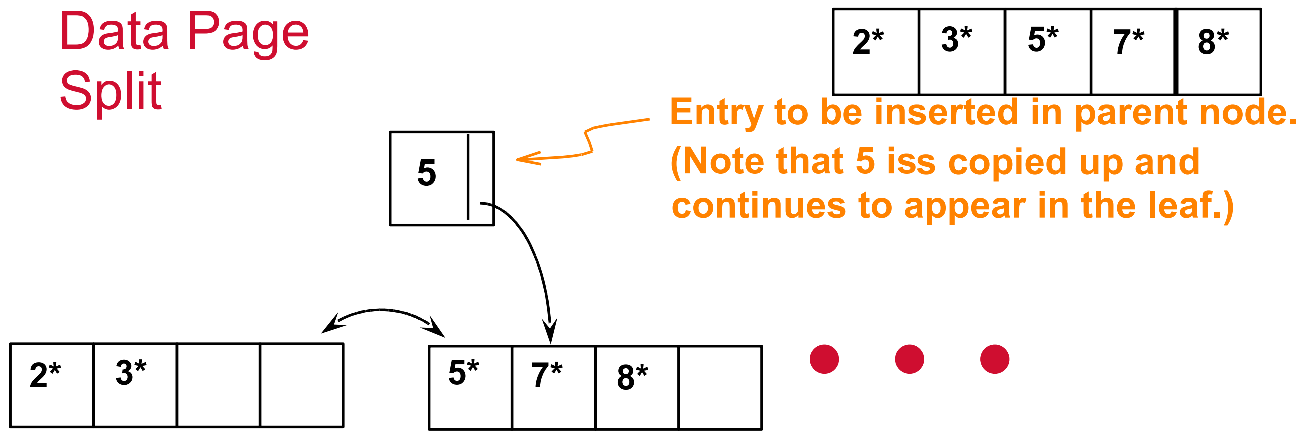
❖Notice that root was split, leading to increase in height.

❖In this example, we can avoid split by re-distributing entries; however, this is usually not done in practice.

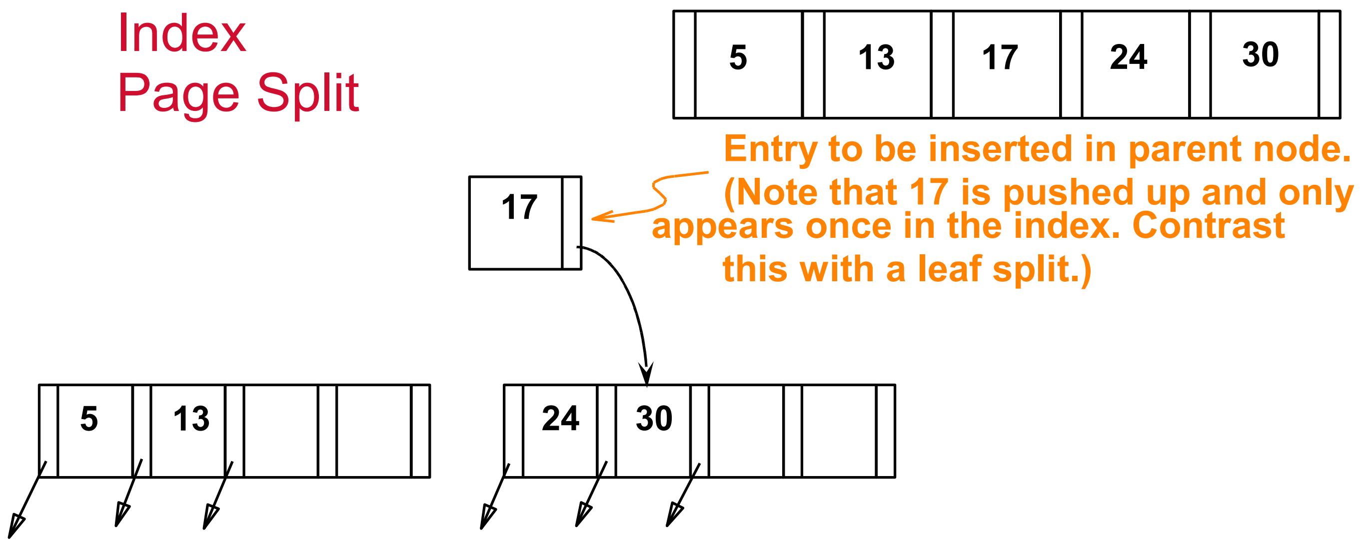
Data vs. Index Page Split
(from previous example of inserting "8")

- Observe how minimum occupancy is guaranteed in both leaf and index pg splits.
- Note difference between *copy-up* and *push-up*; be sure you understand the reasons for this.

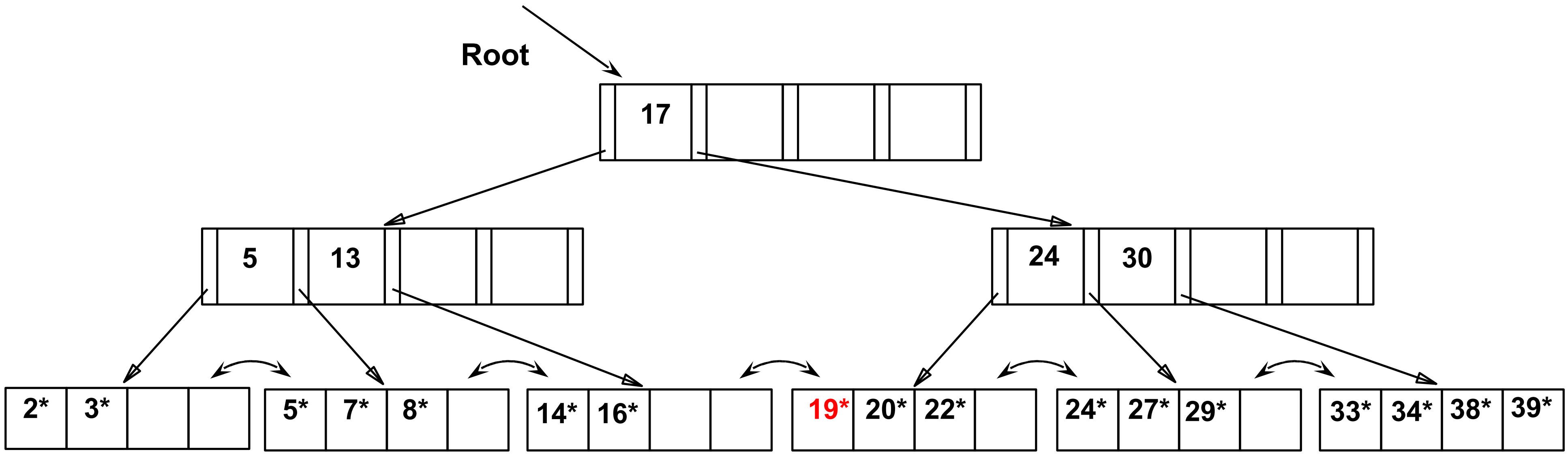
Data Page Split

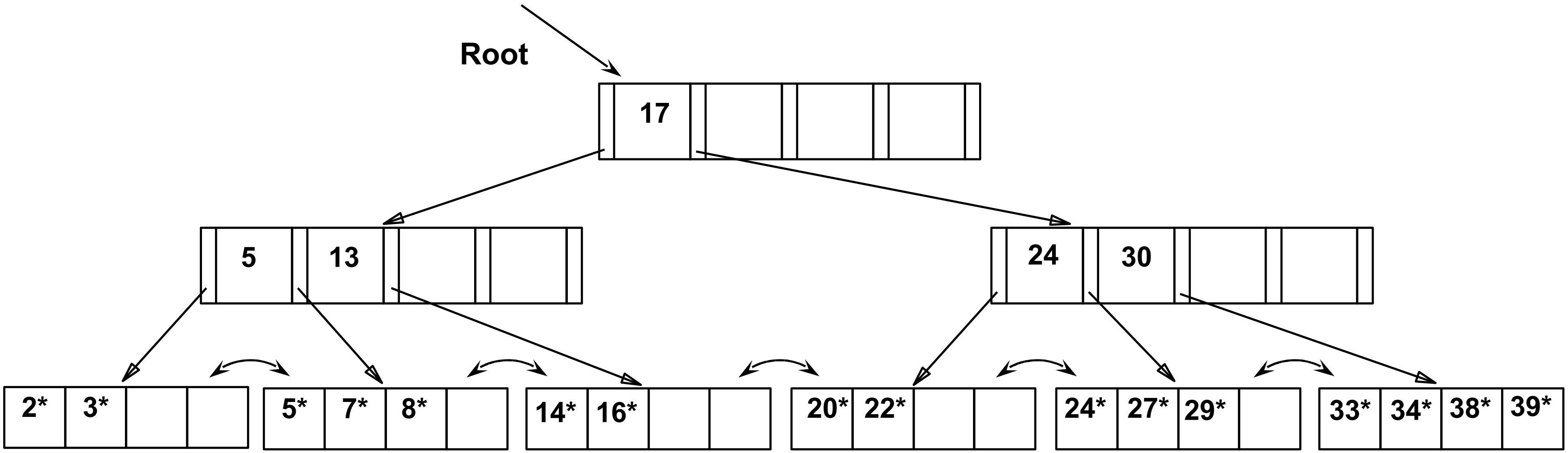


Index Page Split

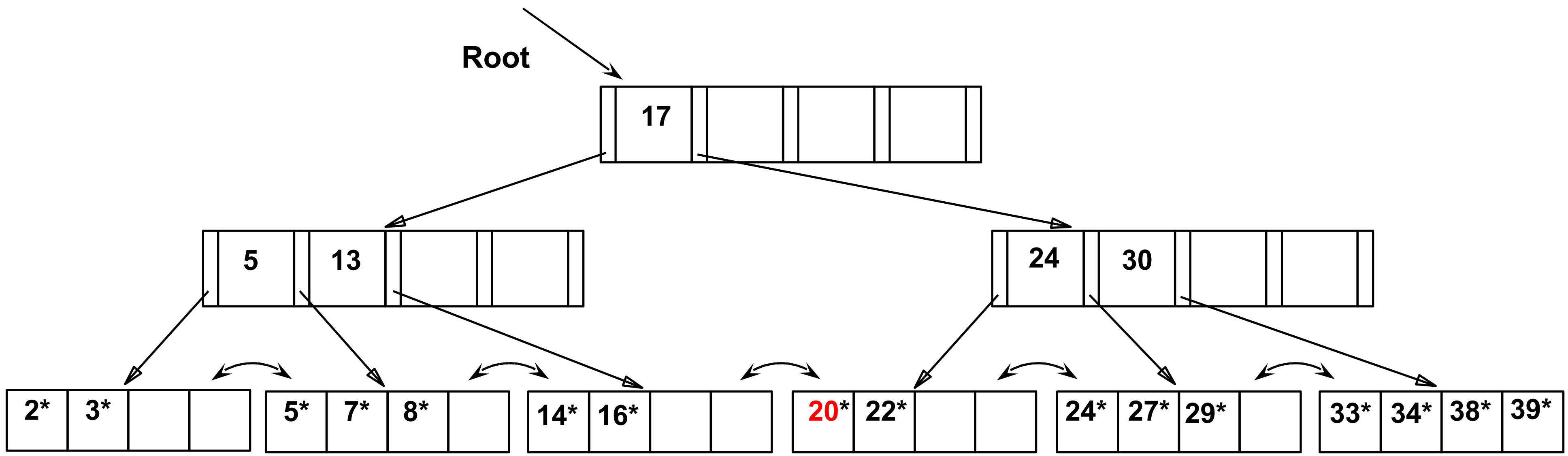


Delete 19*

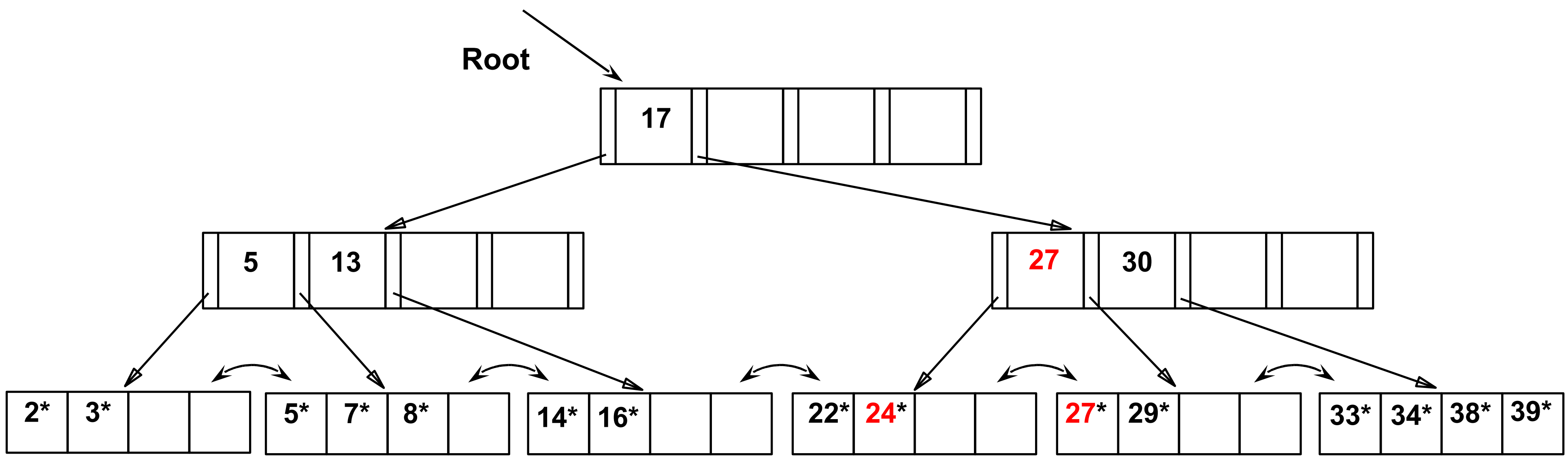

Root
17
5
13
24
30
2*
3*
5*
7*
8*
14*
16*
19*
20*
22*
24*
27*
29*
33*
34*
38*
39*


Root
17
5
13
24
30
2*
3*
5*
7*
8*
14*
16*
20*
22*
24*
27*
29*
33*
34*
38*
39*

Delete 20* ...



Root
17
5
13
24
30
2*
3*
5*
7*
8*
14*
16*
20*
22*
24*
27*
29*
33*
34*
38*
39*

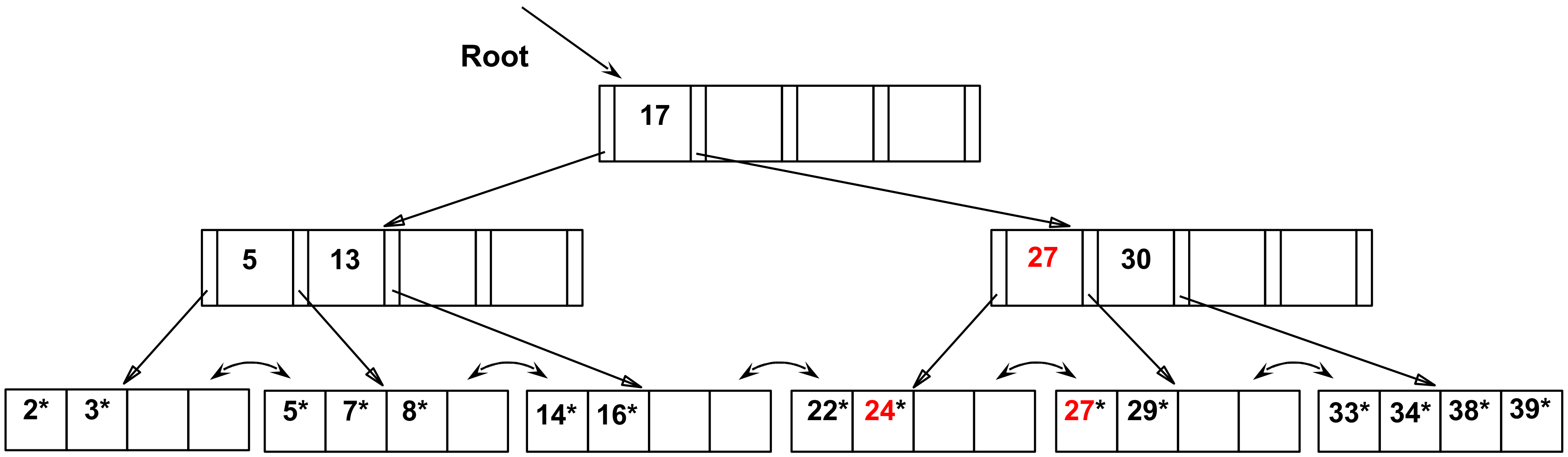


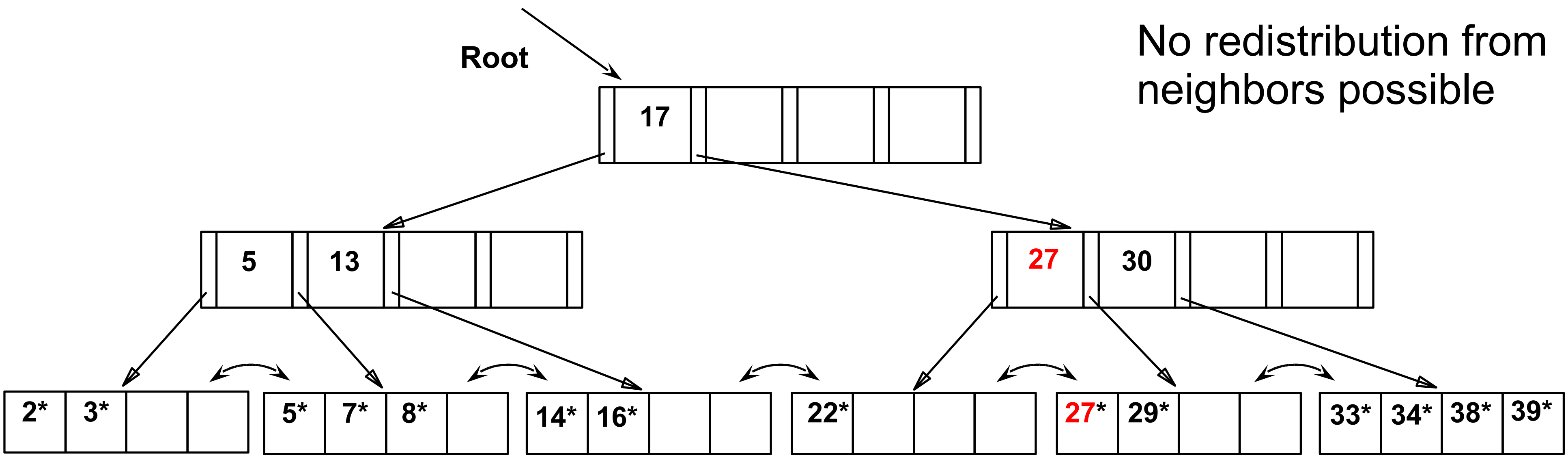
Root
17
5
13
27
30
2*
3*
5*
7*
8*
14*
16*
22*
24*
27*
29*
33*
34*
38*
39*

Delete 19* and 20* ...

- Deleting 19* is easy.
- Deleting 20* is done with re-distribution. Notice how middle key is *copied up*.
- Further deleting 24* results in more drastic changes

Delete 24* ...

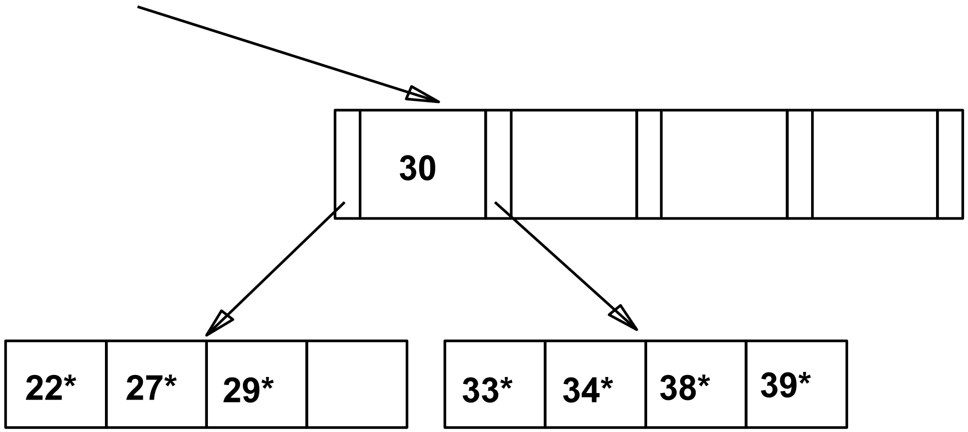


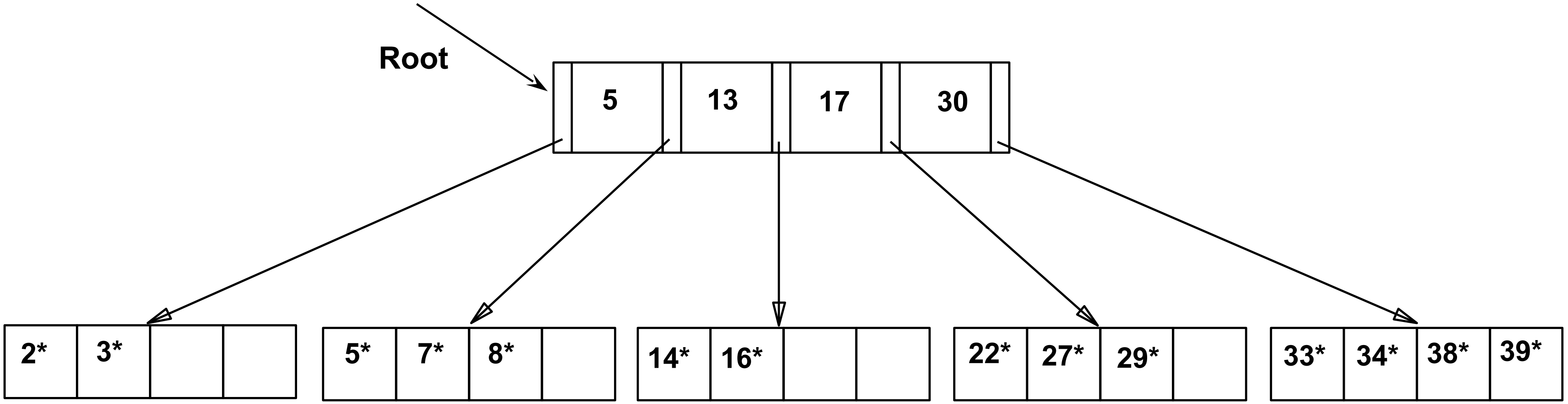


No redistribution from neighbors possible

Deleting 24*

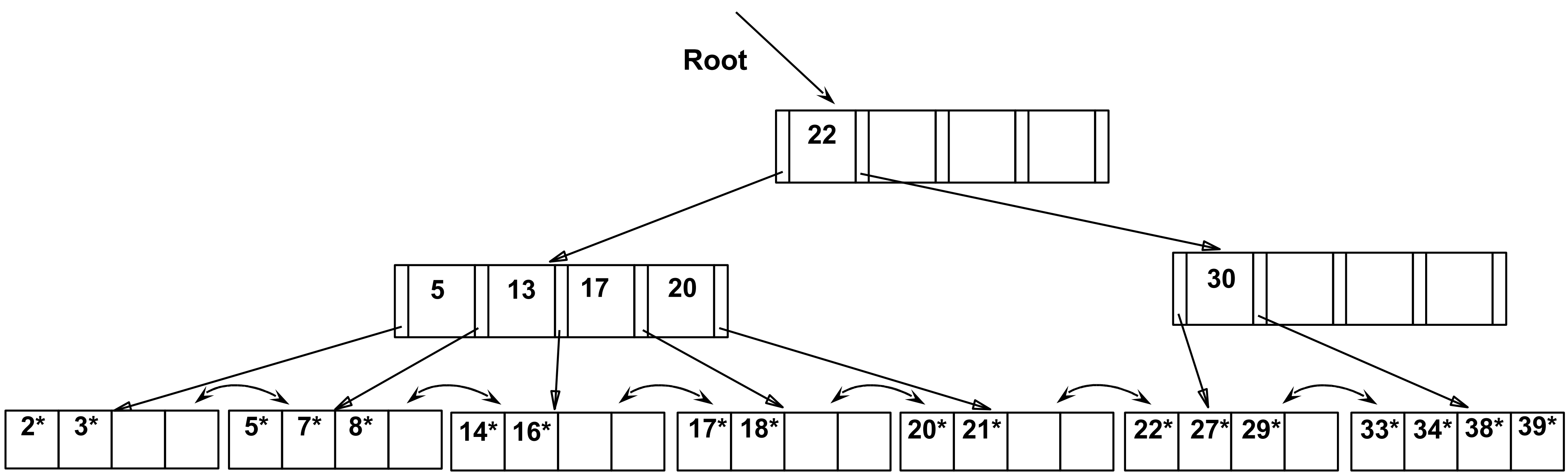
- Must merge.
- Observe `*toss*' of index entry (on right), and `*pull down*' of index entry (below).





Example of Non-leaf Re-distribution

- Tree is shown below *during deletion* of 24*. (What could be a possible initial tree?)
- In contrast to previous example, can re-distribute entry from left child of root to right child.



After Re-distribution

- Intuitively, entries are re-distributed by `*pushing through*' the splitting entry in the parent node.
- It suffices to re-distribute index entry with key 20; we've re-distributed 17 as well for illustration.