Introduction to Database Systems

Exercises: indexing (Part 1)

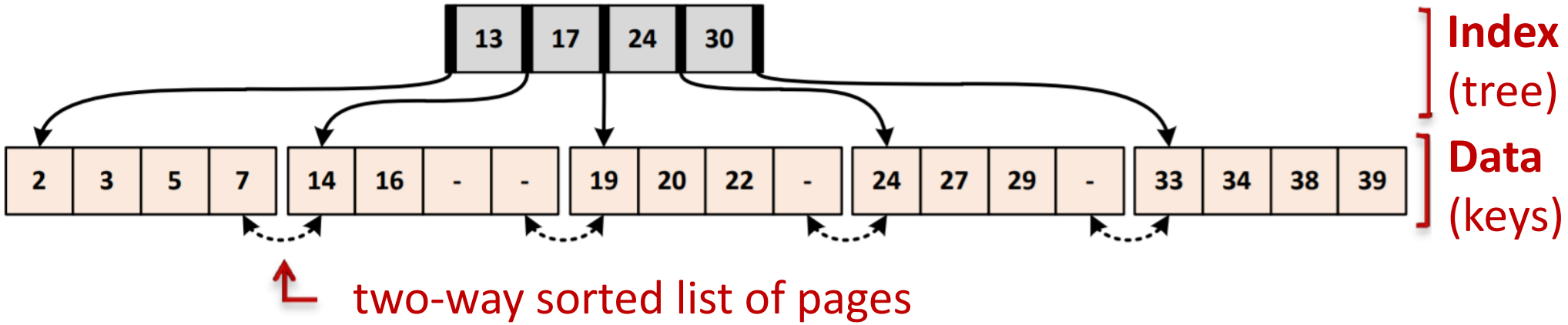
Source: D. Lavbič, Osnove podatkovnih baz - vaje, UL FRI

Introduction

- The purpose of indexing is to **increase the efficiency of the system**.
  - We usually want to improve the efficiency (speed) of the queries.
- Indexes are **pointers** (shortcuts) **to records** so that they can be found faster and we do not need to search the complete database.
- We will cover the following indexing methods, which are commonly found in databases:
  - B+ index (most commonly used in databases),
  - ISAM index and
  - Hash index

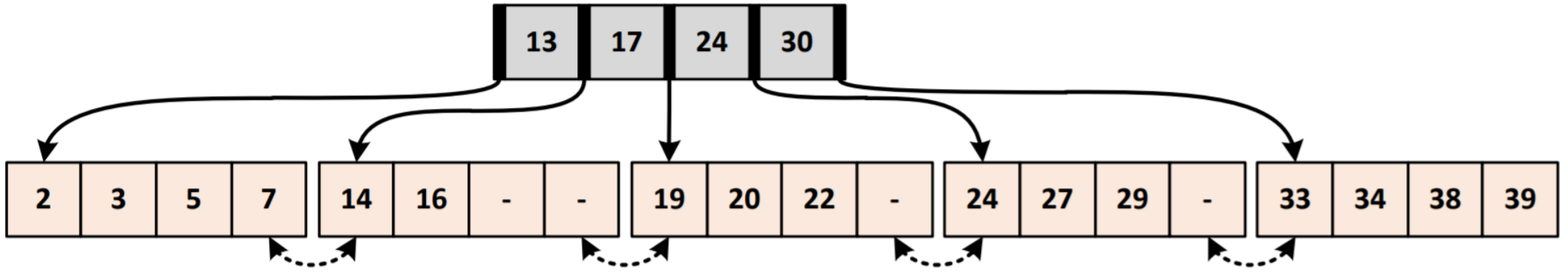
B+ index

- Dynamic = The structure is adjusted to the changes in the file.
  - Balanced tree where nodes direct search, and data (keys) is stored in leafs.
  - The insert and delete operations keep the tree balanced.



B+ index

- Condition of the node: **d ≤ m ≤ 2d** (at least 50% full)
  - **m** … number of records (occupancy) of the node
  - **d** … order of the tree (**2d** = capacity of the node)
  - For the root: **1 ≤ m ≤ 2d**

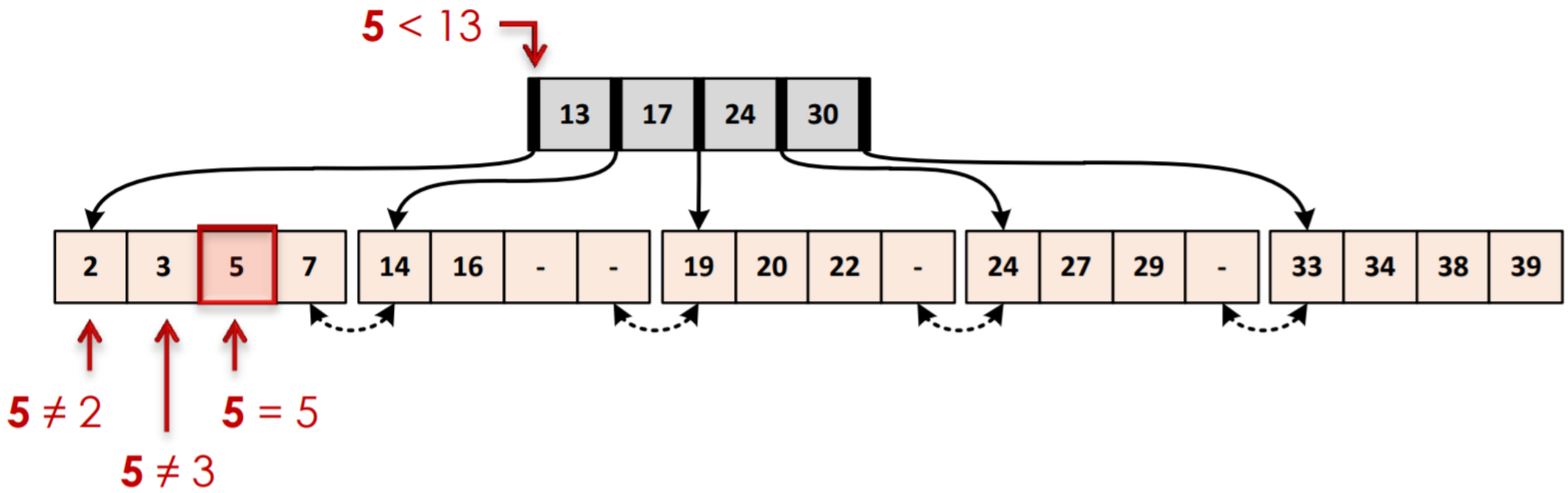


Order of the tree **d=2**.

Node **occupancy** (index) is **at least 2** and **up to 4** records.

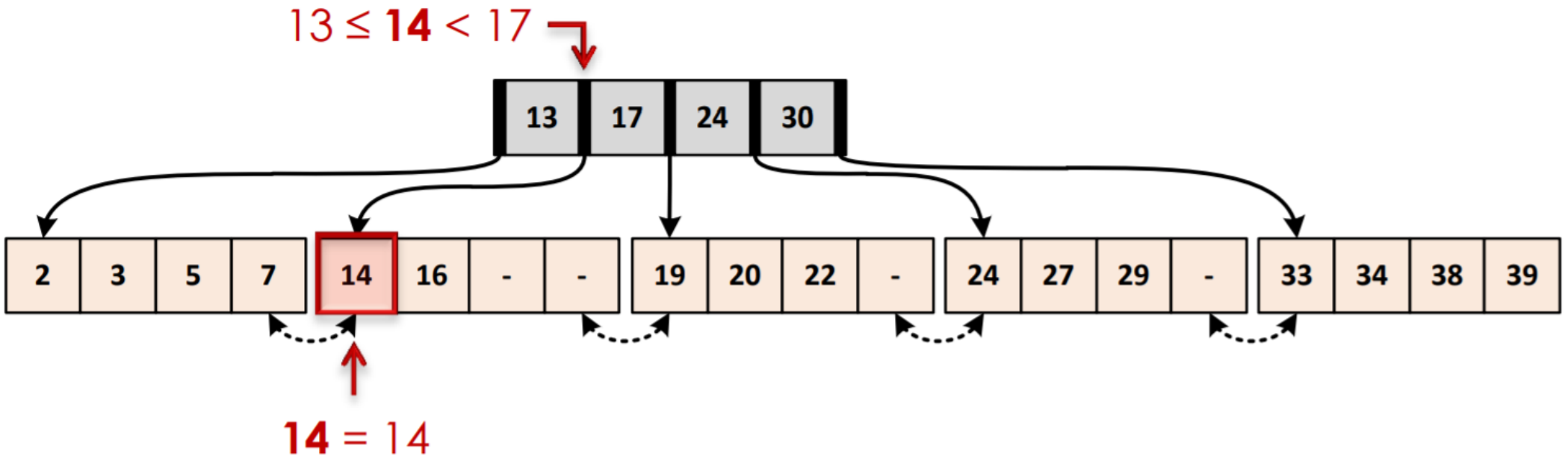
B+ index – example

- **Find** the entry with key **5**.



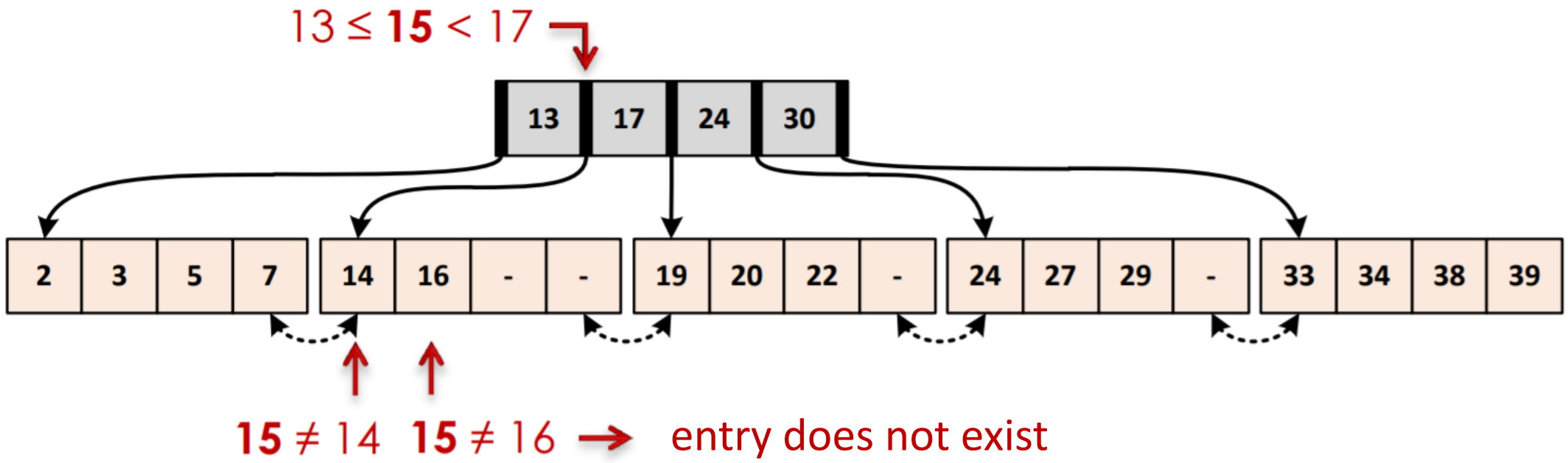
B+ index – example

- **Find** the entry with key **14**.



B+ index – example

- **Find** the entry with key **15**.

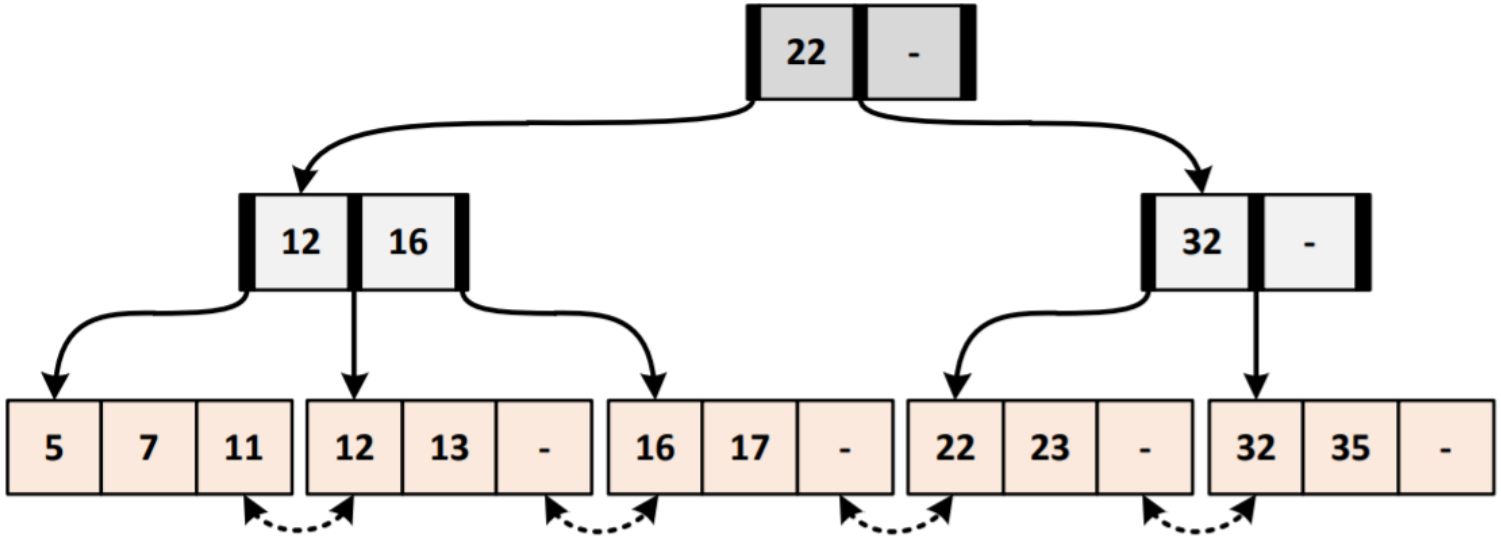


B+ index – inserting data

- Algorithm for adding a record k*:
  - **FIND** the leaf where k* belongs
  - **IF** there si free space in the leaf, insert k*
  - **ELSE** divide the leaf in 2 parts:
    - d elements go to the left leaf
    - the rest go to the right leaf
  - If necessary, correct the split key up the index tree.
  - There are 2 versions: **with** and **without redistribution**
    - If not specified, by default it is used without redistribution.

B+ index – example 1

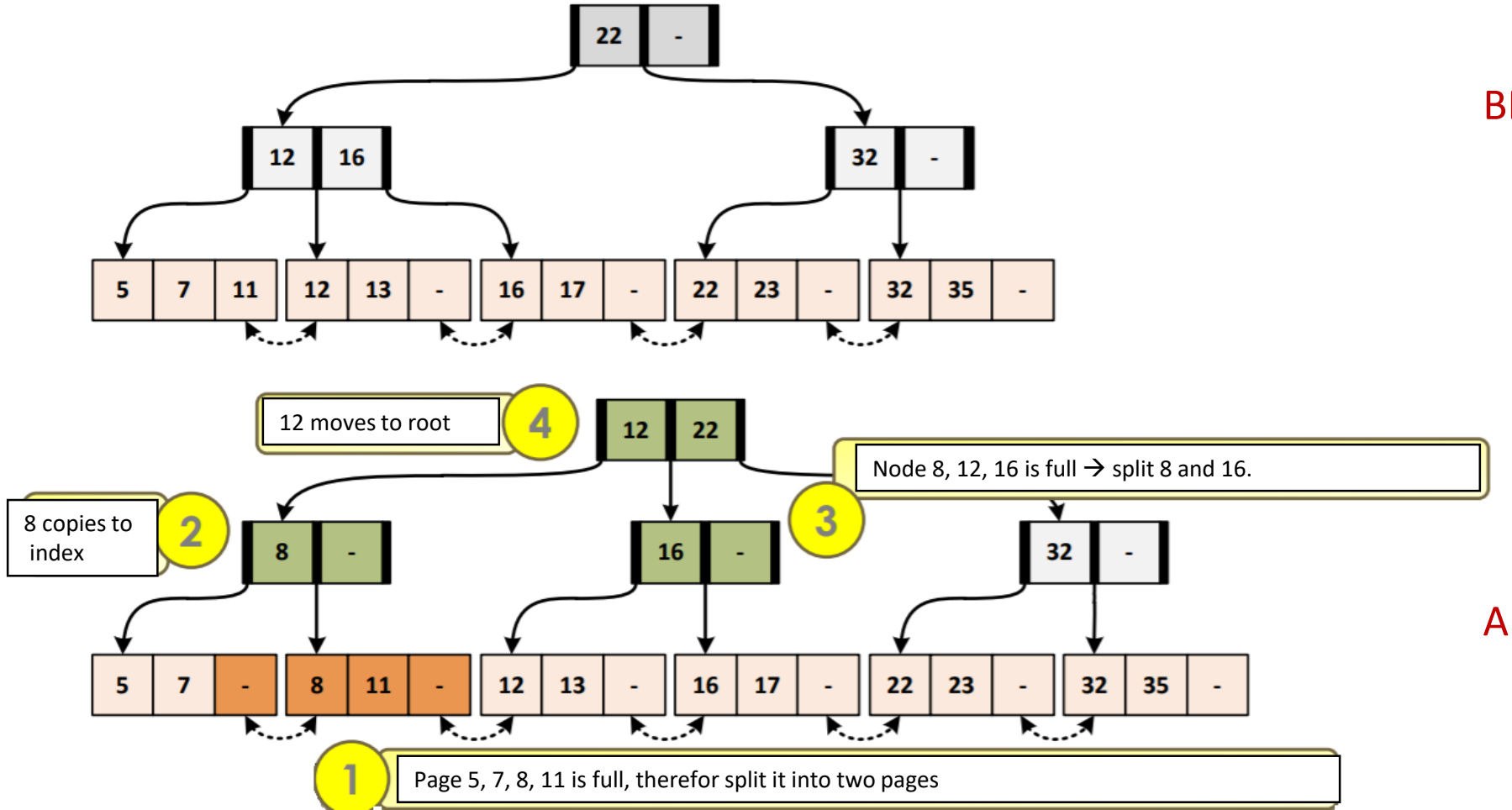
- The constructed B+ index is shown on the figure below:



- Insert the entry with a key **8** once **without redistribution** and once **with redistribution**.

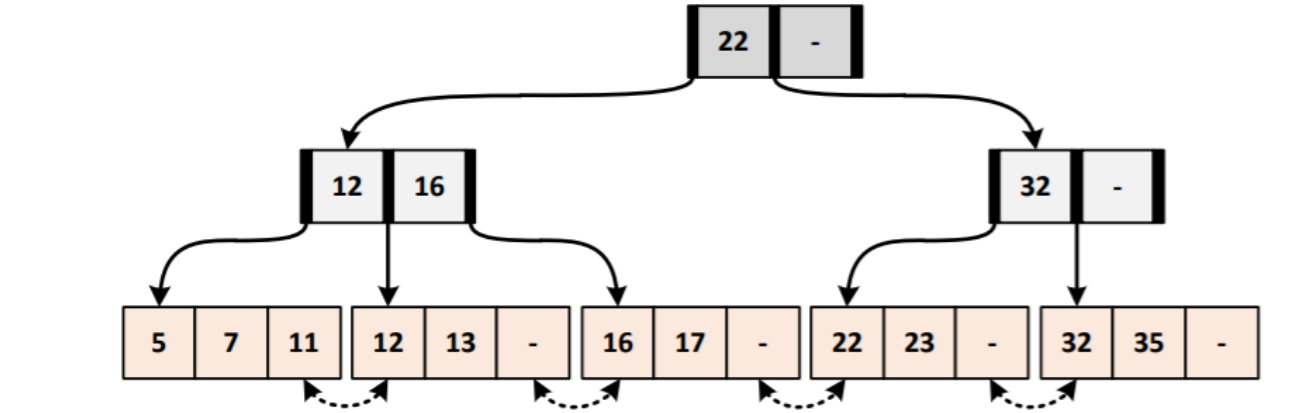
B+ index – example 1

- Inserting the entry with key **8 (without redistribution)**:

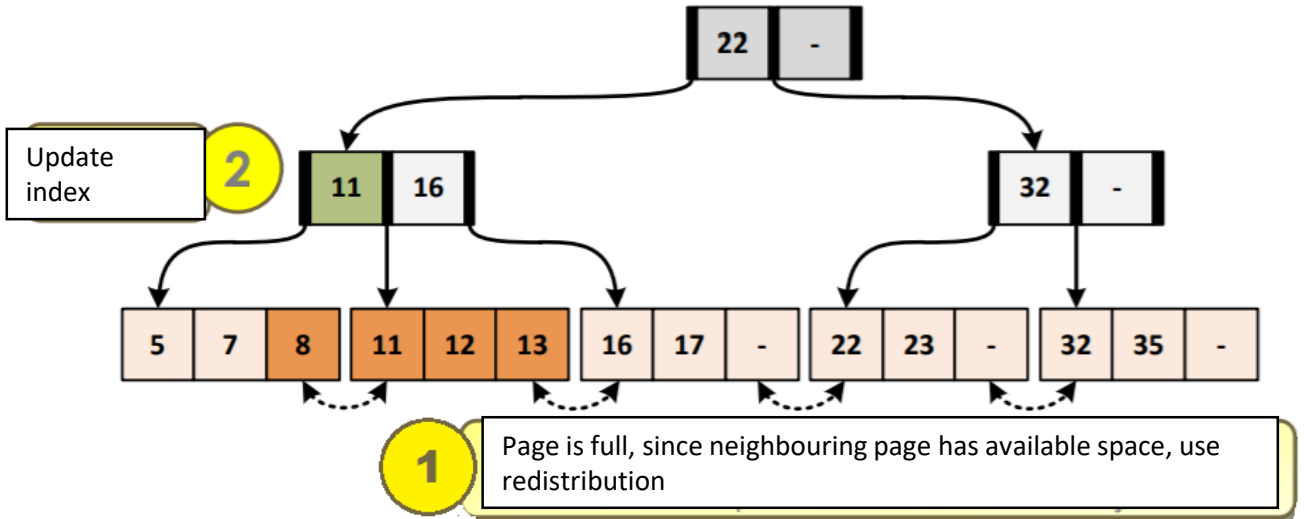


B+ index – example 1

- Inserting the entry with key **8 (with redistribution)**::



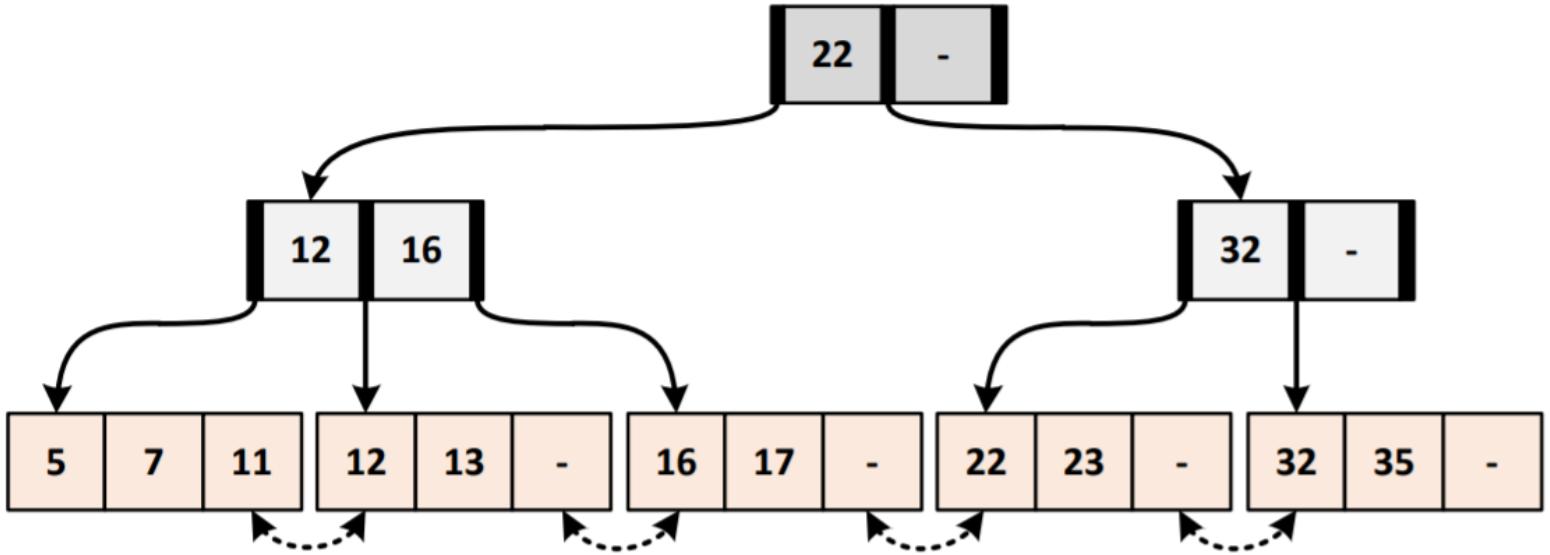
BEFORE



AFTER

B+ index – example 2

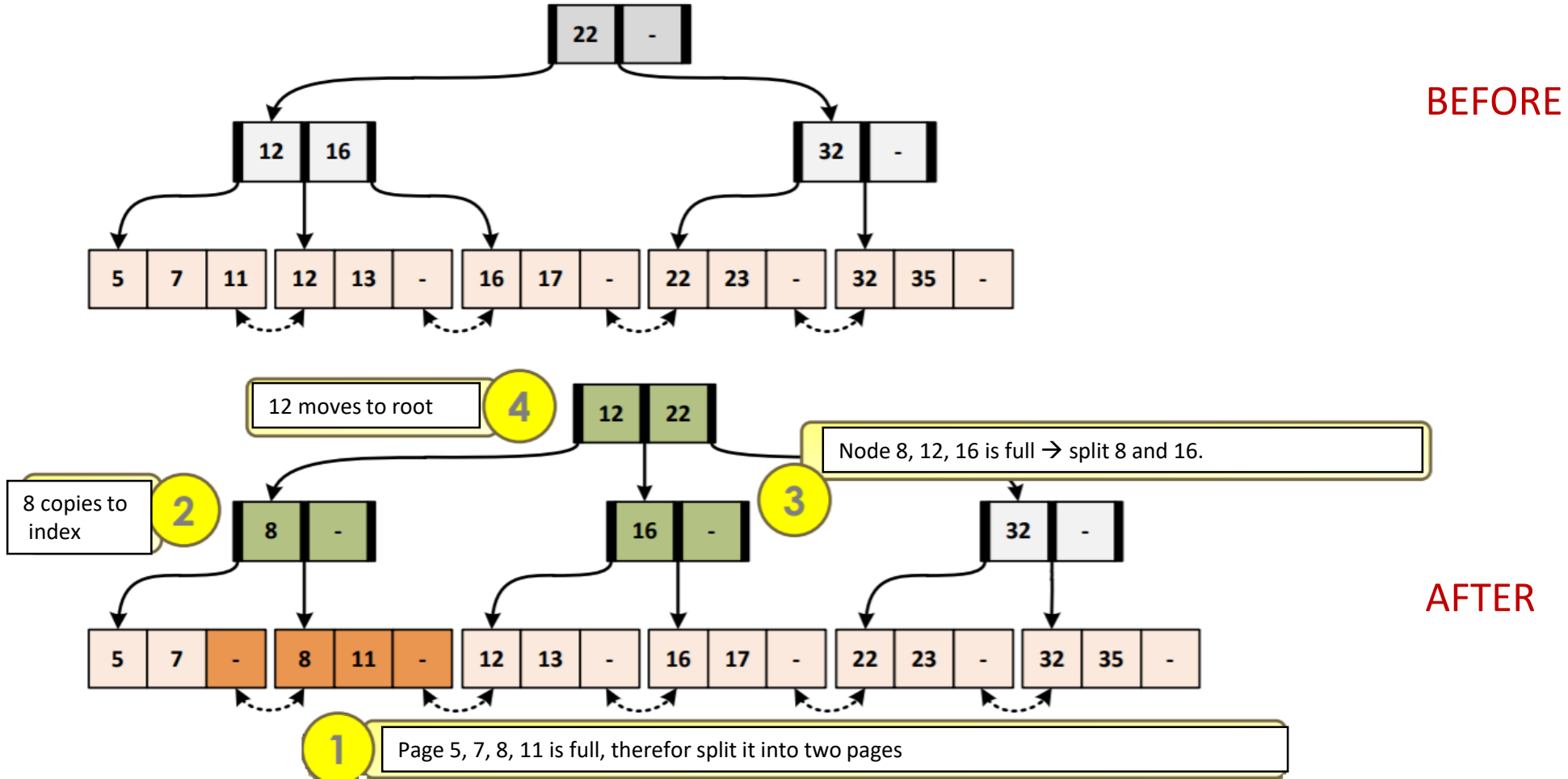
- The constructed B+ index is shown on the figure below:



- First **insert** the record with **key 8**, then **delete** the record with **key 32**.

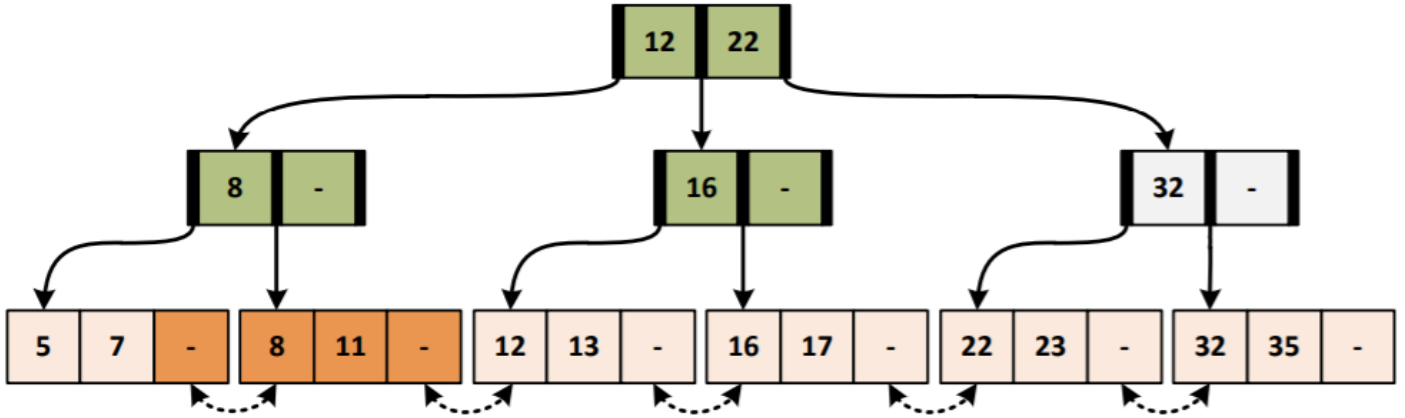
B+ index – example 2

- Inserting the entry with key **8.**

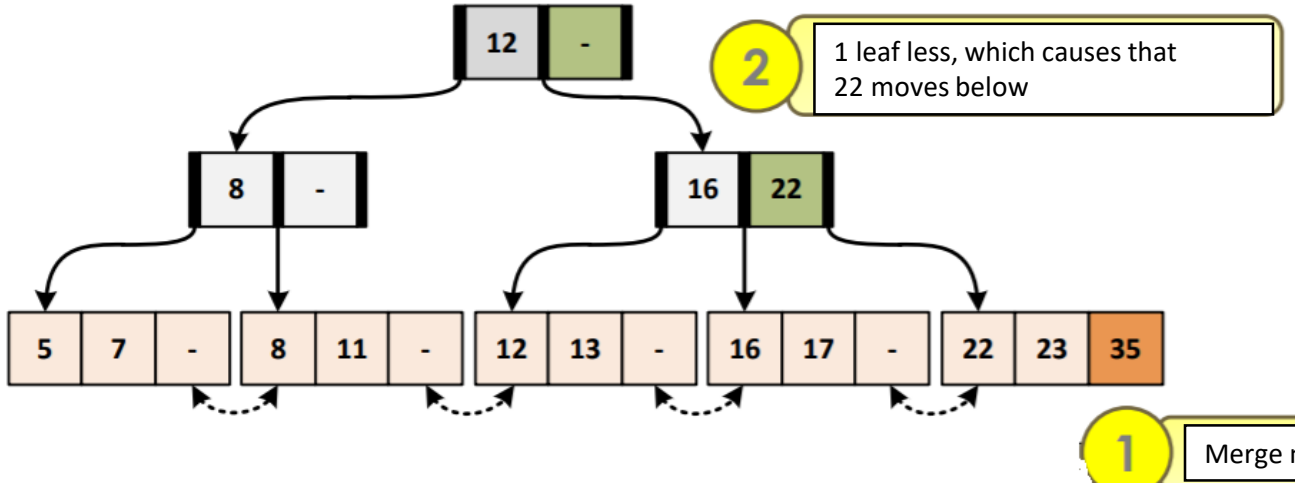


B+ index – example 2

- **Delete** the entry with key **32**.



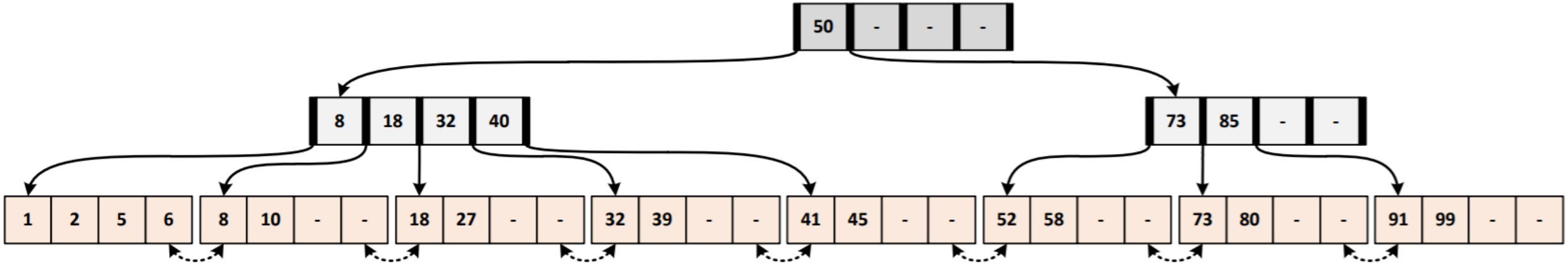
PRED operacijo



PO operaciji

B+ index – Exercise 1

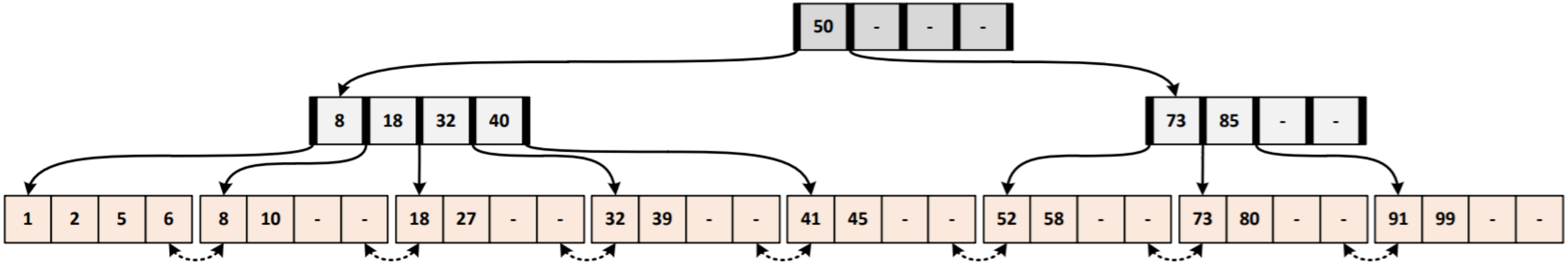
- The constructed B+ index is shown on the figure below.
- You will have to perform certain operations and display the result in the form of a tree.



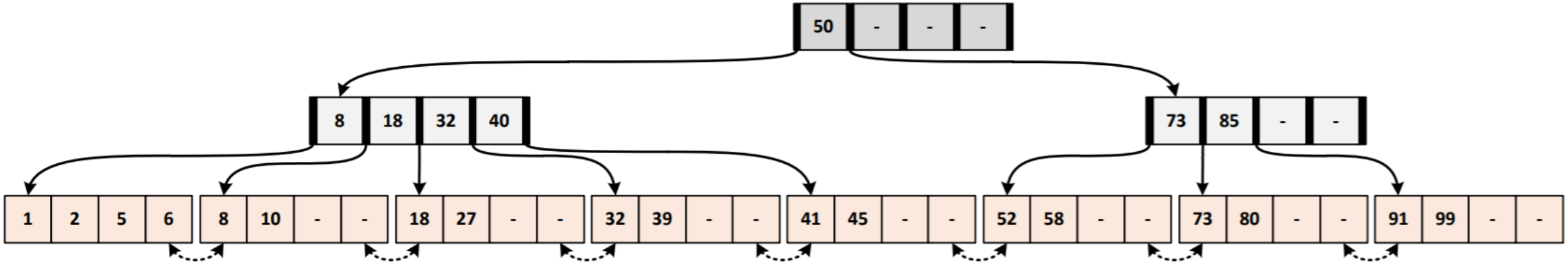
B+ index – Exercise 1

- **Insert** the entry with key **9**.
- **Insert** the entry with key **3**.
- **Delete** the entry with key **8** (redistribution with left side).
- **Delete** the entry with key **8** (redistribution with right side).
- **Insert** the entry with key **46** and then **delete** the entry with key **52**.
- **Delete** the entry with key **91**.
- **Delete** the entries with key **32**, **39**, **41**, **45, 73**.

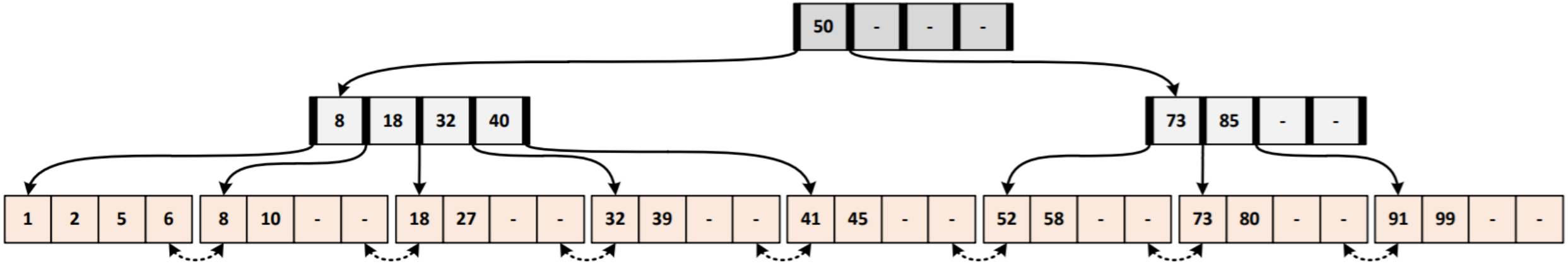
Insert the entry with key **9**.



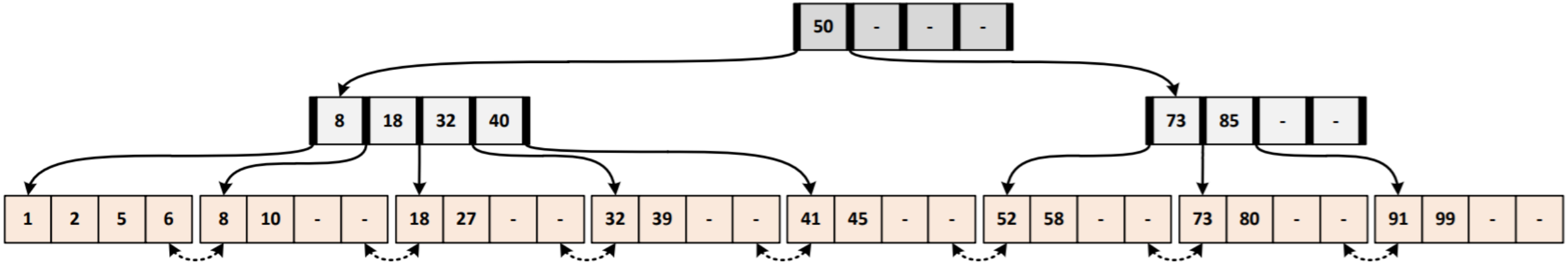
Insert the entry with key **3**.



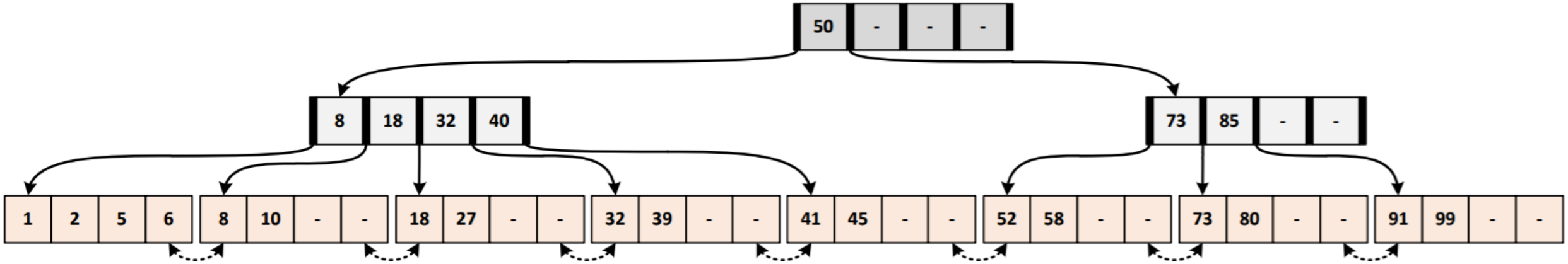
Delete the entry with key **8** (redistribution with left side).



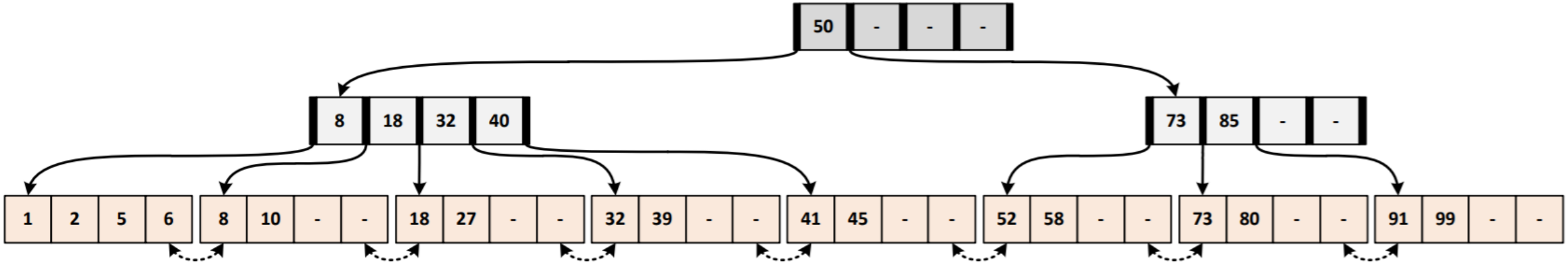
Delete the entry with key **8** (redistribution with right side).



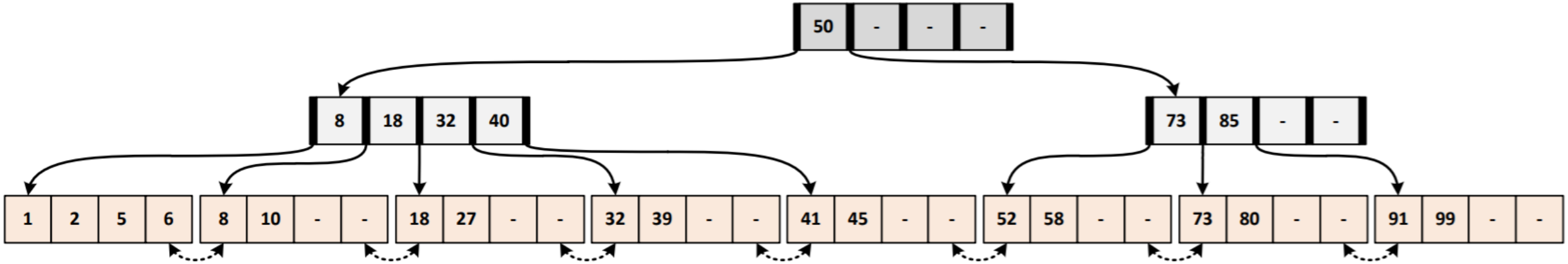
Insert the entry with key **46** and then **delete** the entry with key **52**.



Delete the entry with key **91**.



Delete the entries with key **32**, **39**, **41**, **45, 73**.



B+ index – Exercise 2

a) Construct a B+-tree for the following set of key values:

(2, 3, 5, 7, 11, 17, 19, 23, 29, 31)

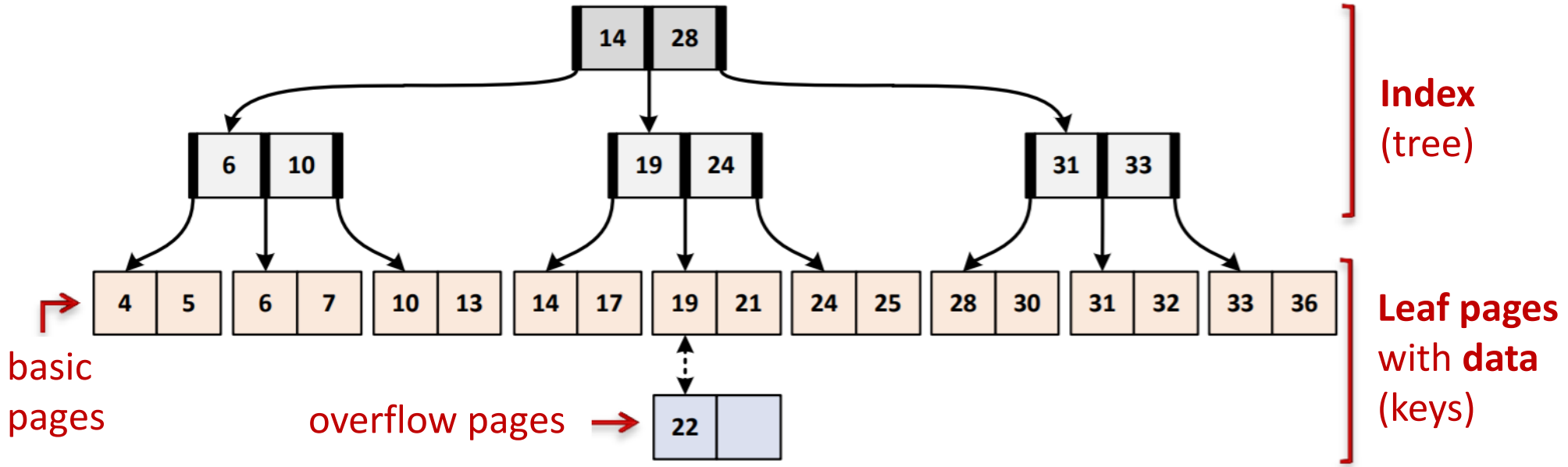
Assume that the tree is initially empty and values are added in ascending order. Construct B+-tree where the number of pointers that will fit in one node is 6.

b) Show the form of the tree (constructed in the example (a)) after each of the following series of operations:

1. Insert 9.
2. Insert 10.
3. Insert 8.
4. Delete 23.
5. Delete 19.

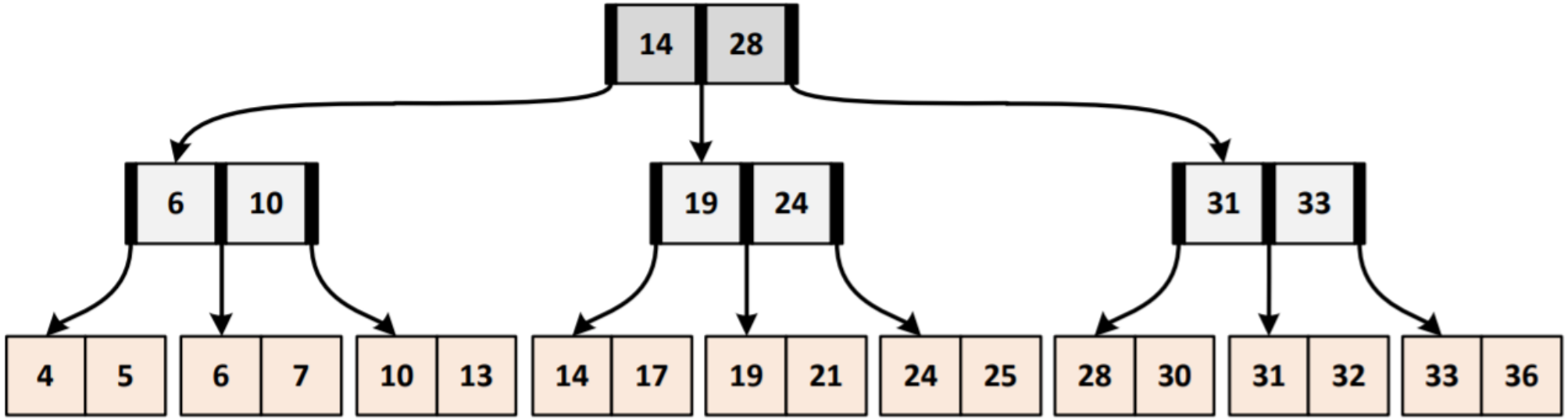
ISAM index

- **Static** (with the exception of overflow pages) = need to be reorganised.
- The data in the overflow pages are not sorted → due to the effectiveness of adding the records.



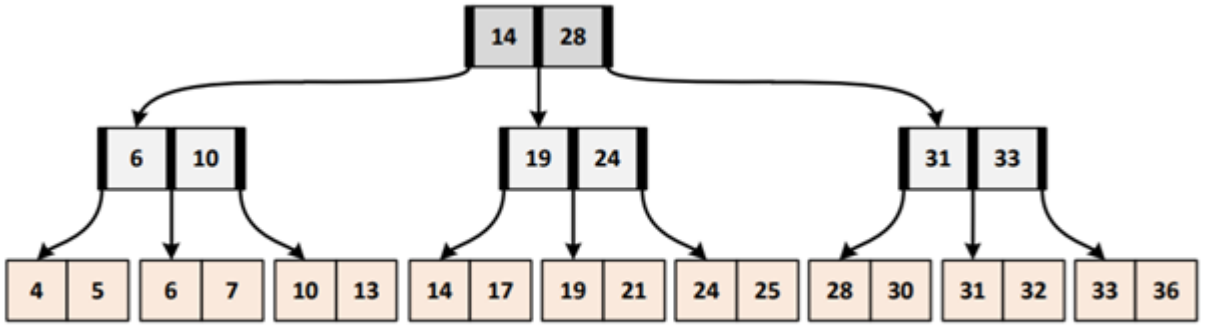
ISAM index – Exercise 3

- The constructed ISAM index is show on the figure below.
- You will have to perform certain operations and display the result in the form of a tree.



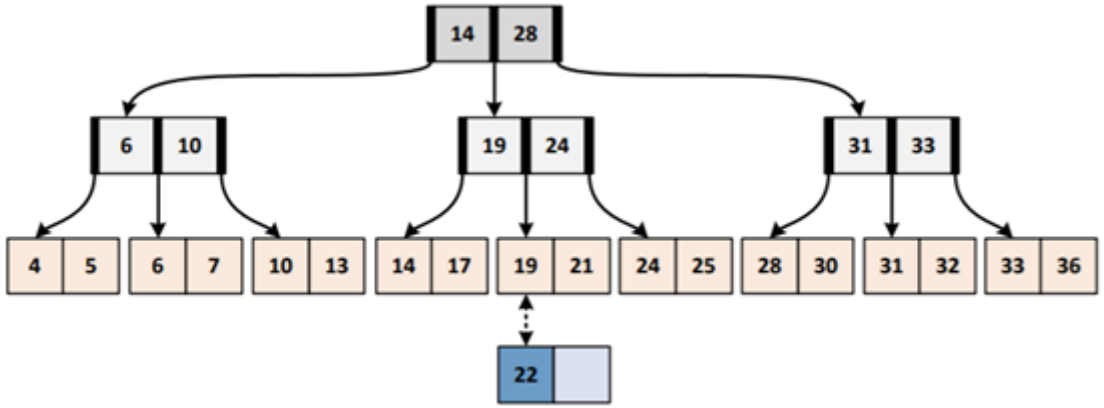
ISAM index – Exercise 3

- **Insert** entry with the key **22**.



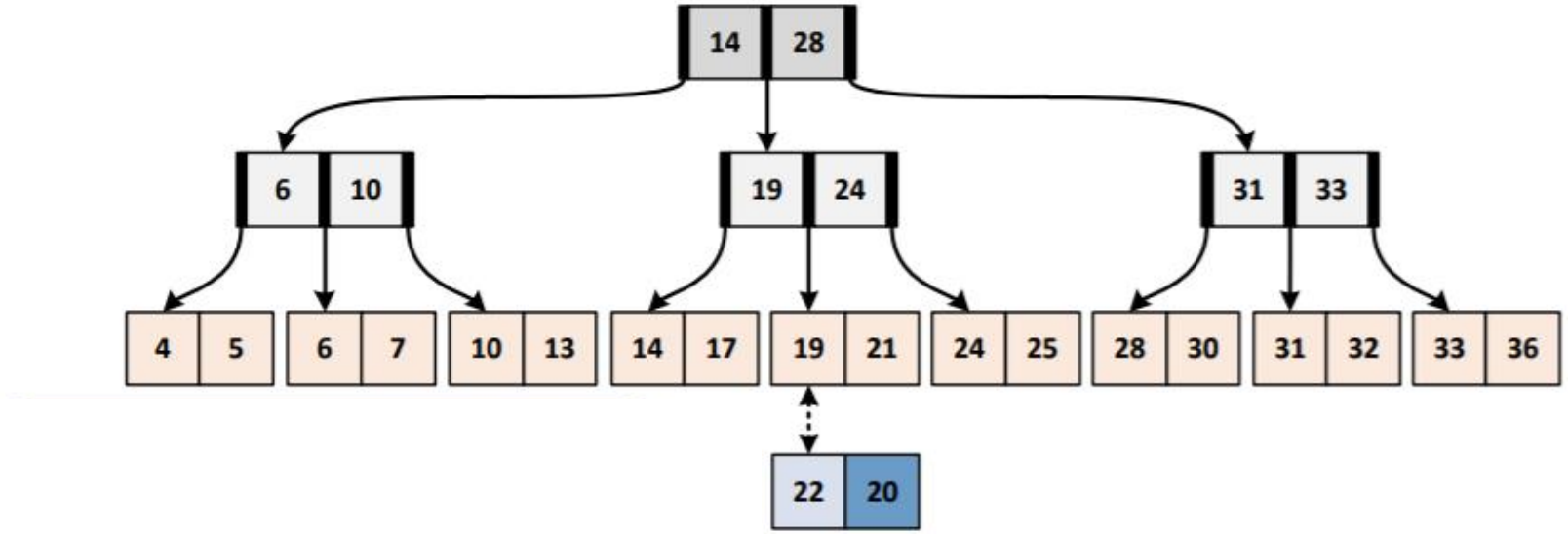
ISAM index – Exercise 3

- **Insert** entry with key **20**.



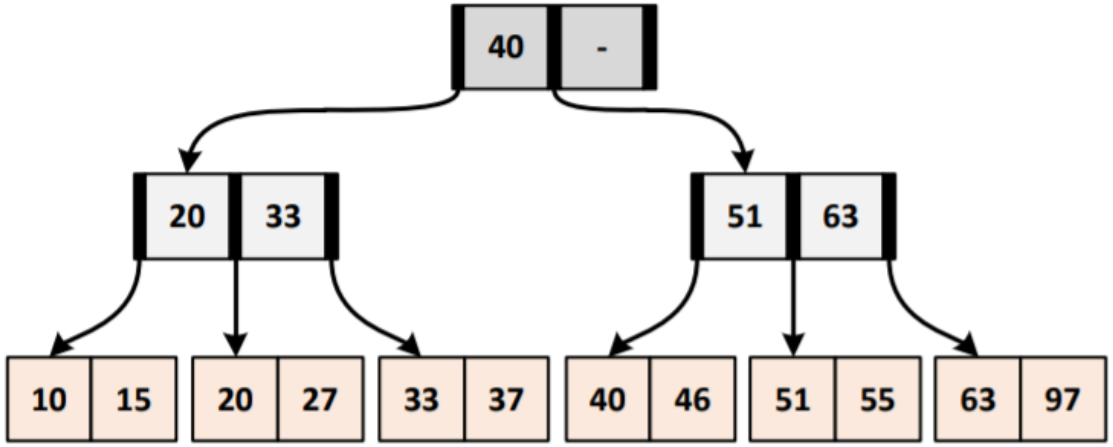
ISAM index – Exercise 3

- **Delete** entries with key **22**, **20, 21, 19, 31**



ISAM and B+ index – Exercise 4

- Two constructed indexes (ISAM and B+) are shown in the figure below (they look the same at the beginning).



- How do the indexes change, if we first **add** entry with the key **41** and after that entry with the key **23**?