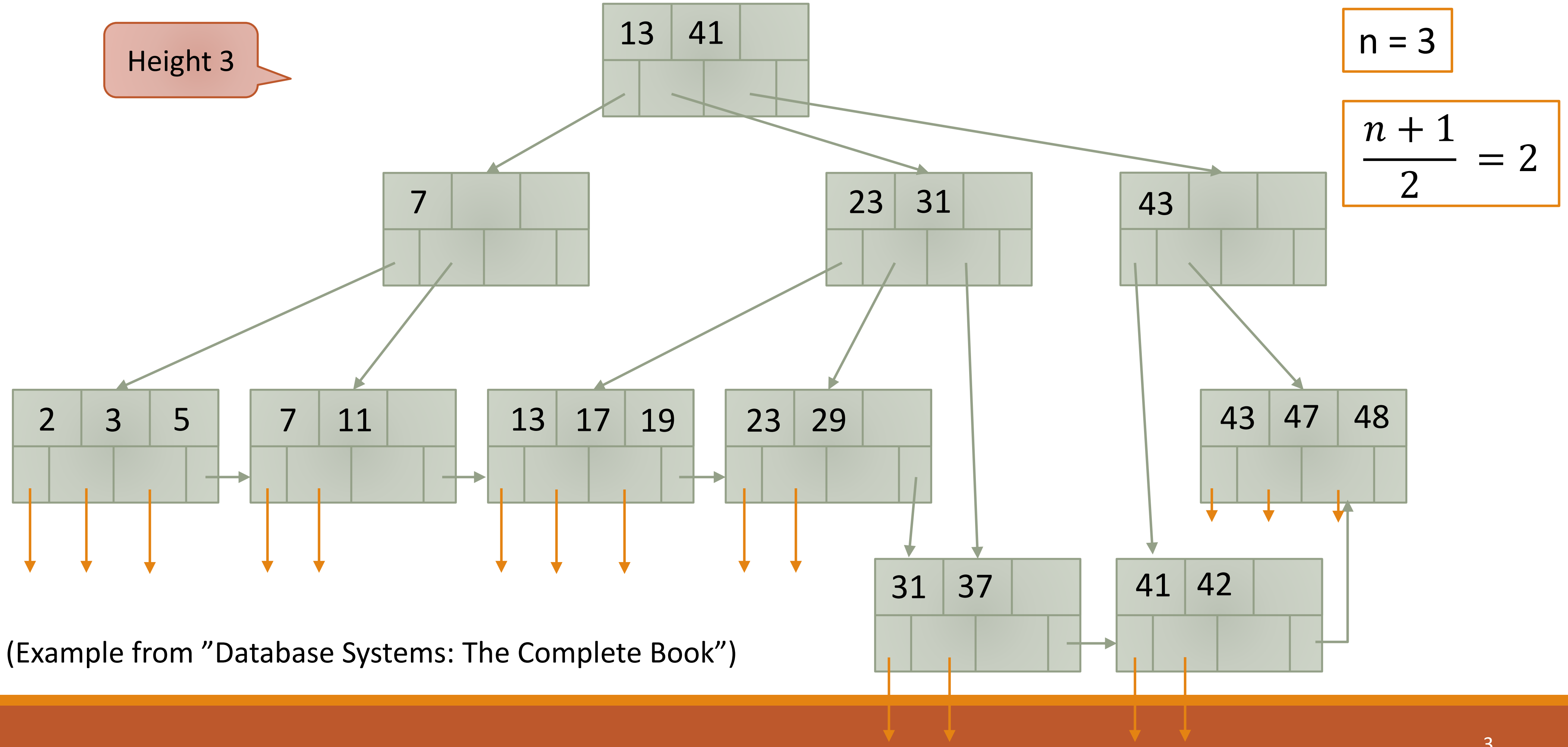
Deletions and conclusions for B+-Trees

Overview of video

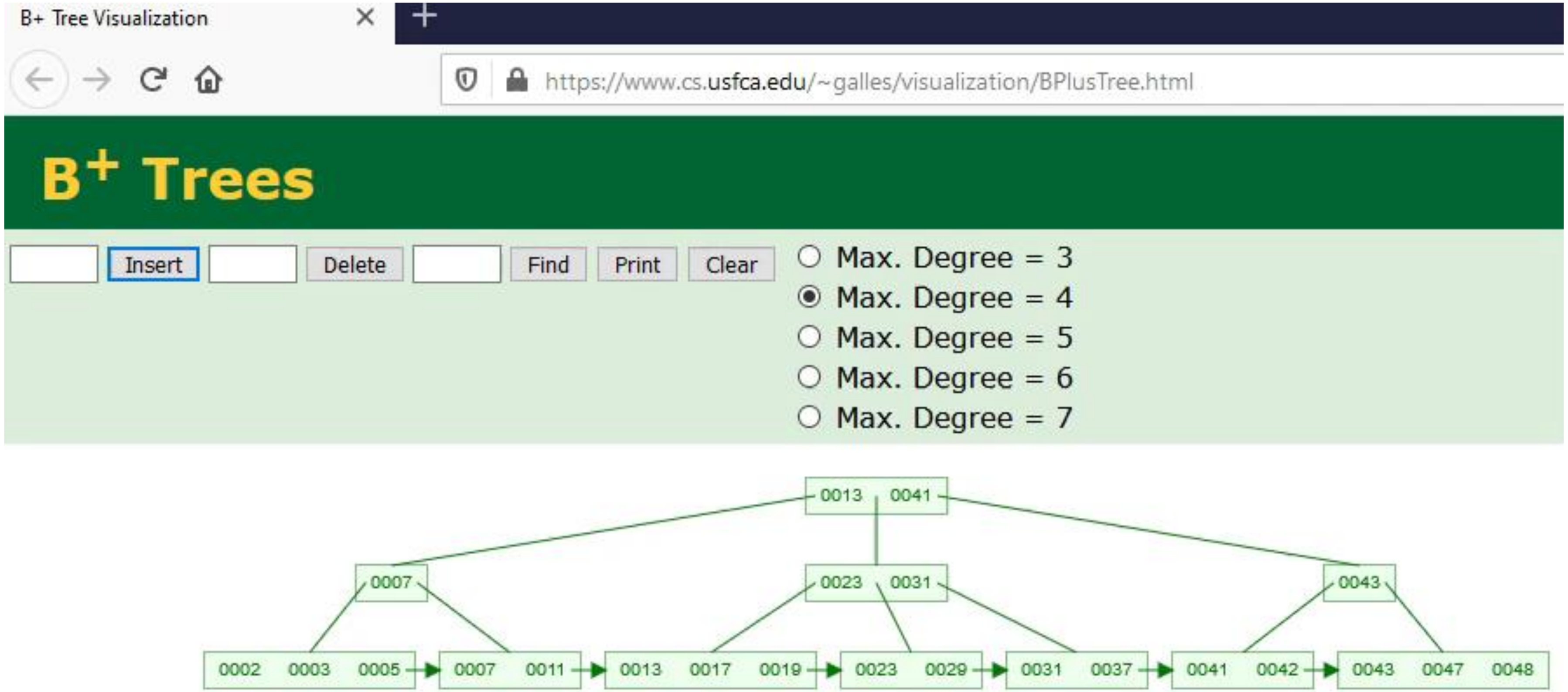
Video discusses how to do deletions as well as a conclusion for B+-trees

Initial tree



(Example from "Database Systems: The Complete Book")

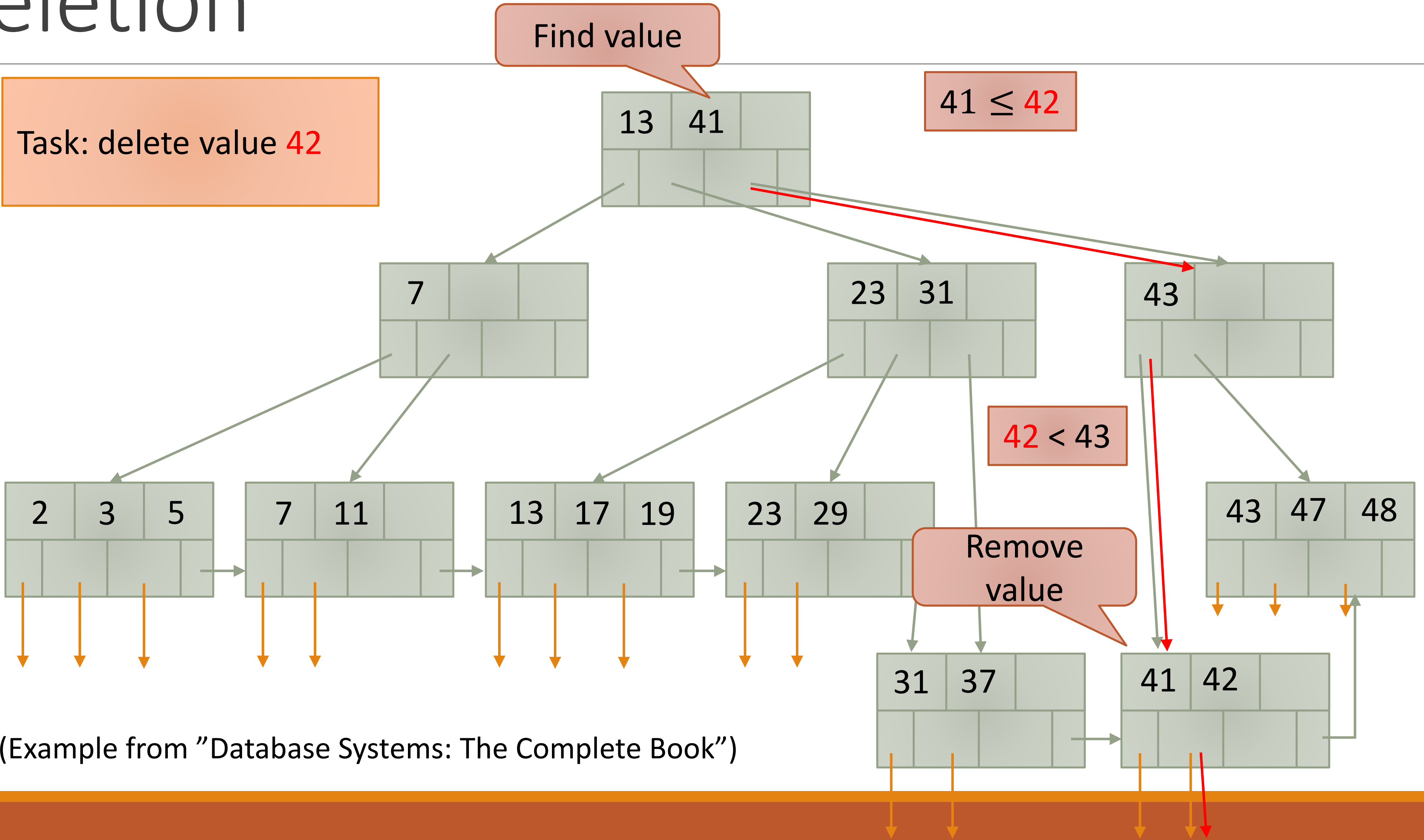
Online version of the initial tree



Delete 42

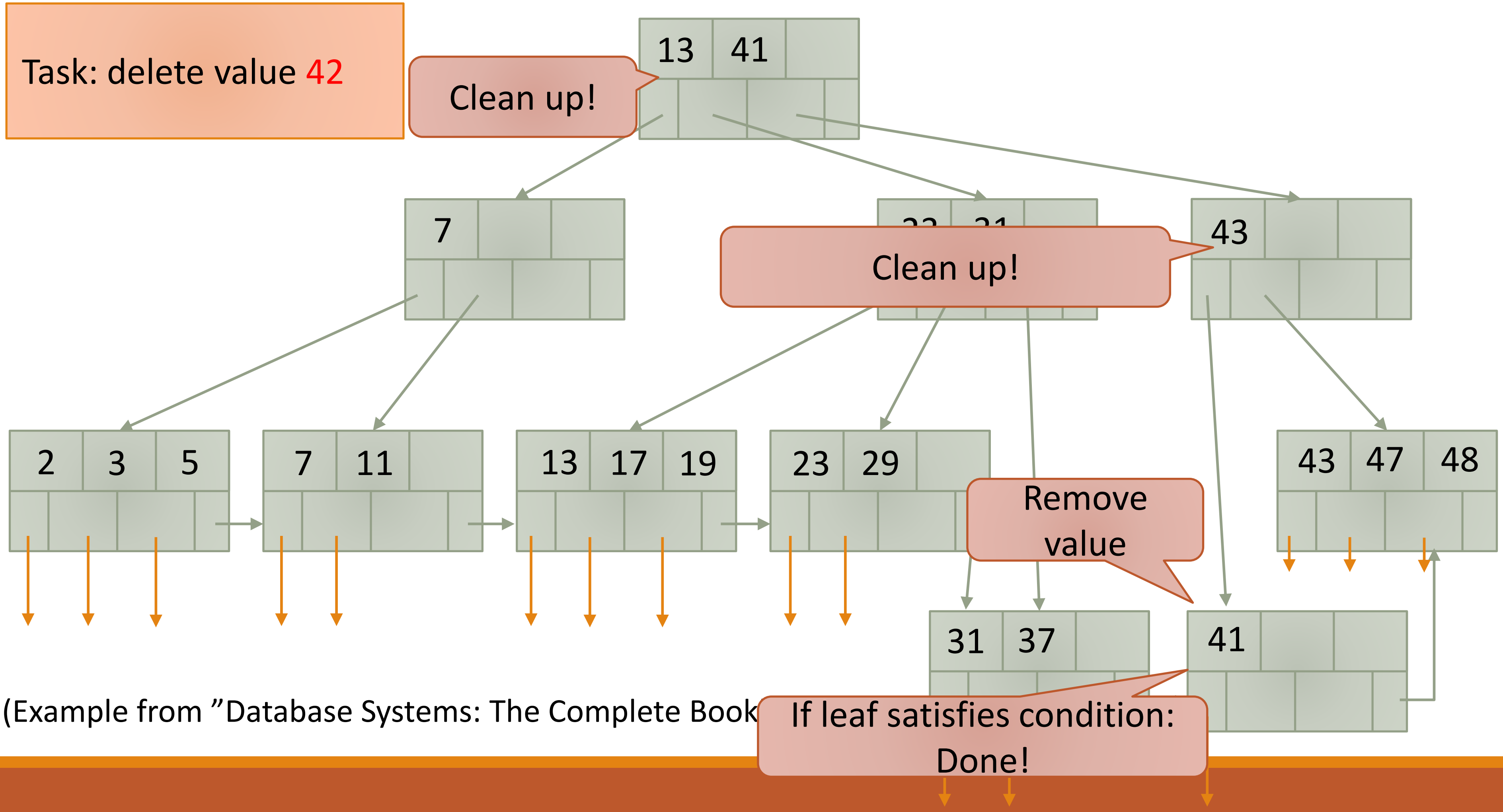
Deletion

Task: delete value 42



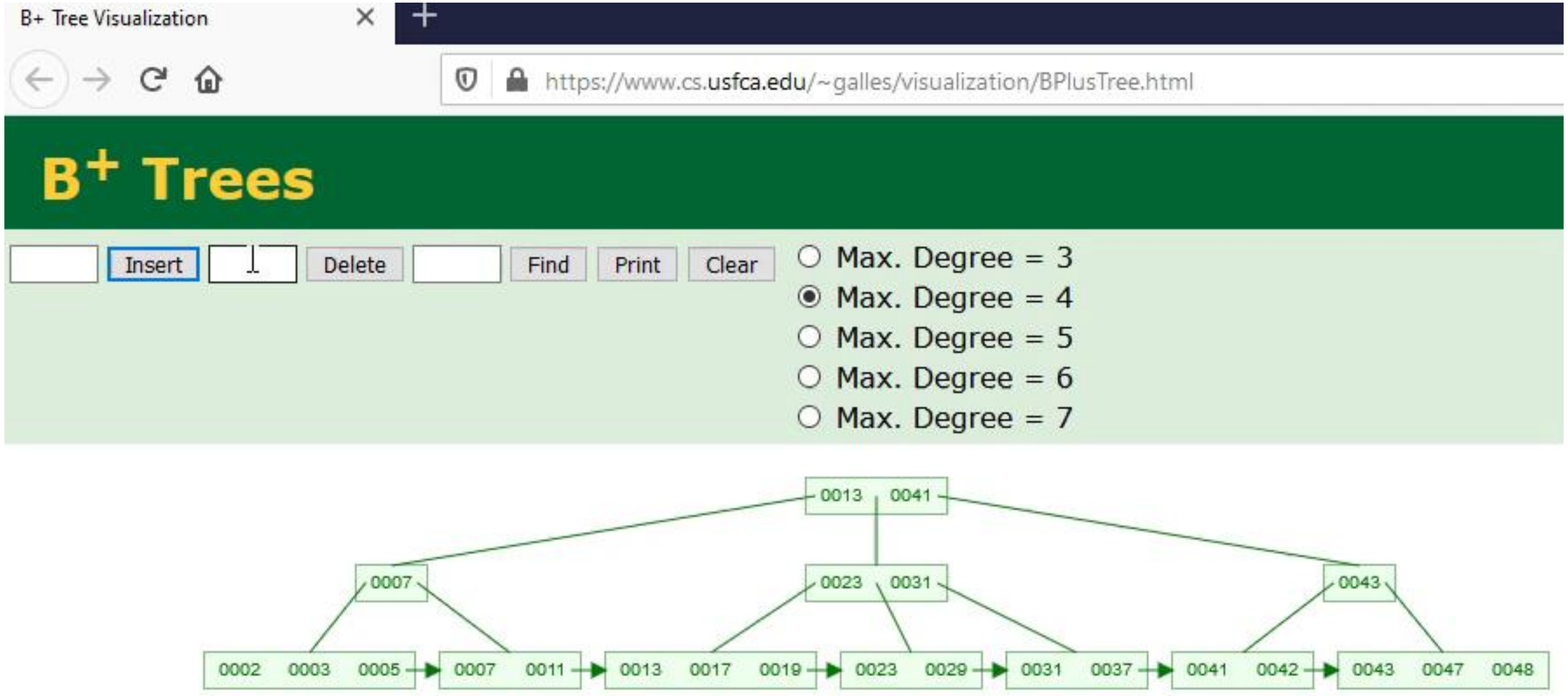
(Example from ”Database Systems: The Complete Book”)

Deletion



(Example from ”Database Systems: The Complete Book

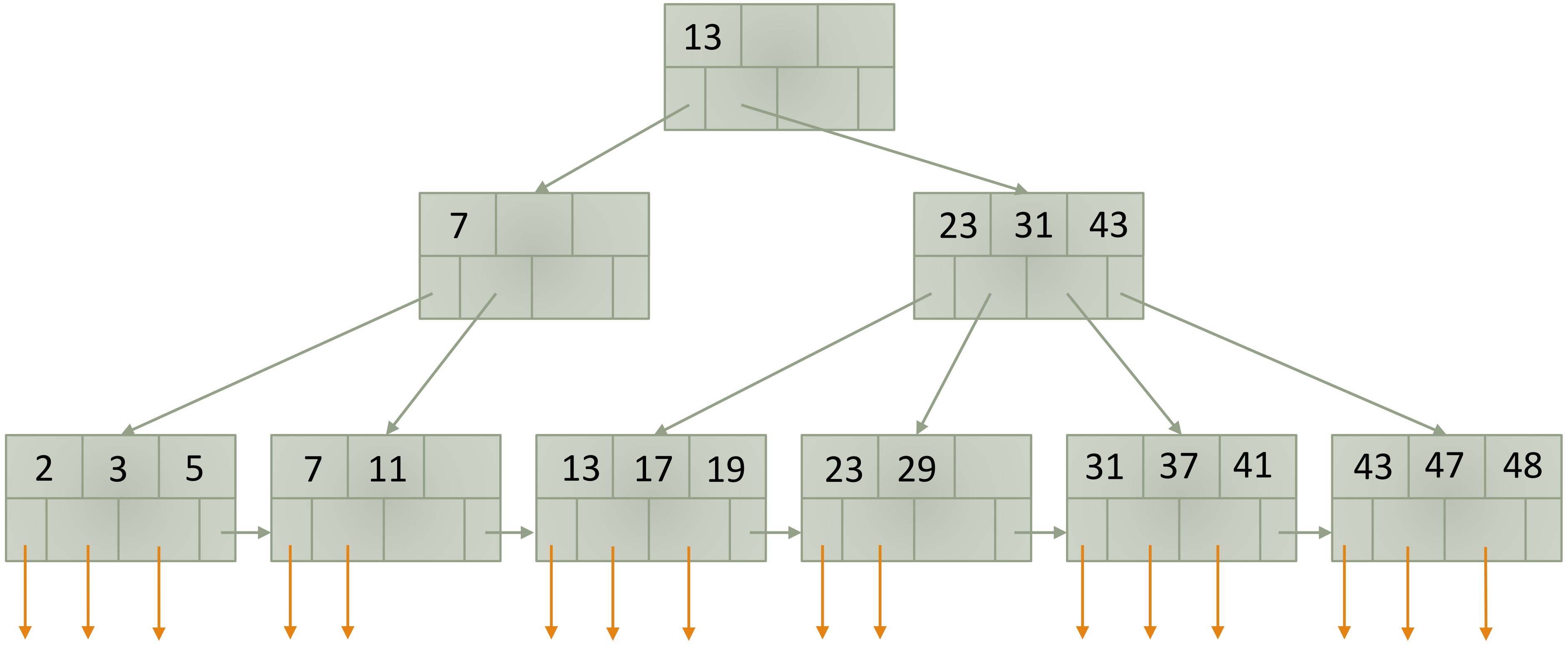
Online version of delete 42



Side note

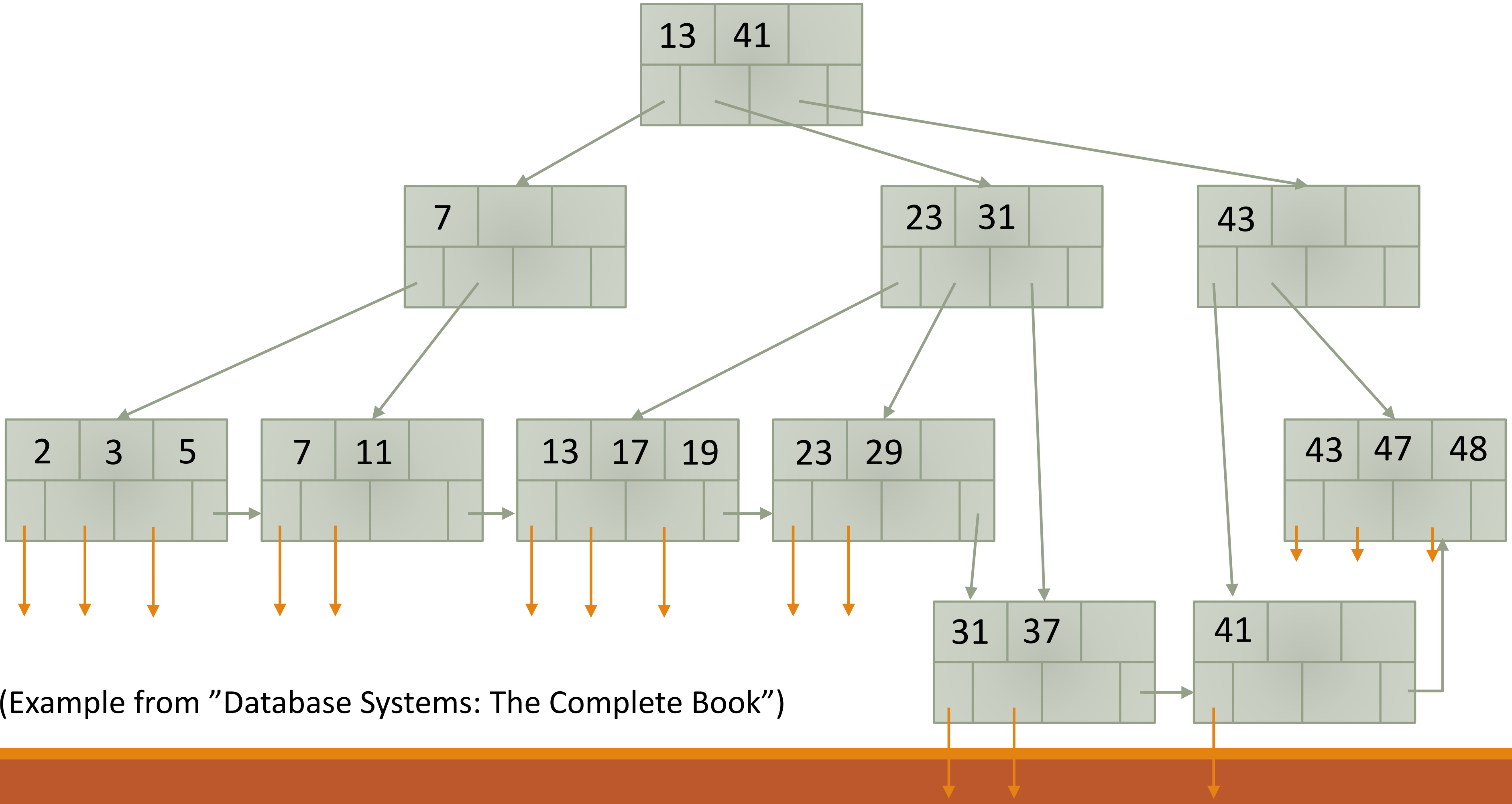
There may be more than one B+ tree with the same leaves

Before insertion + deletion of 42



(Example from ”Database Systems: The Complete Book”)

After insertion + deletion of 42

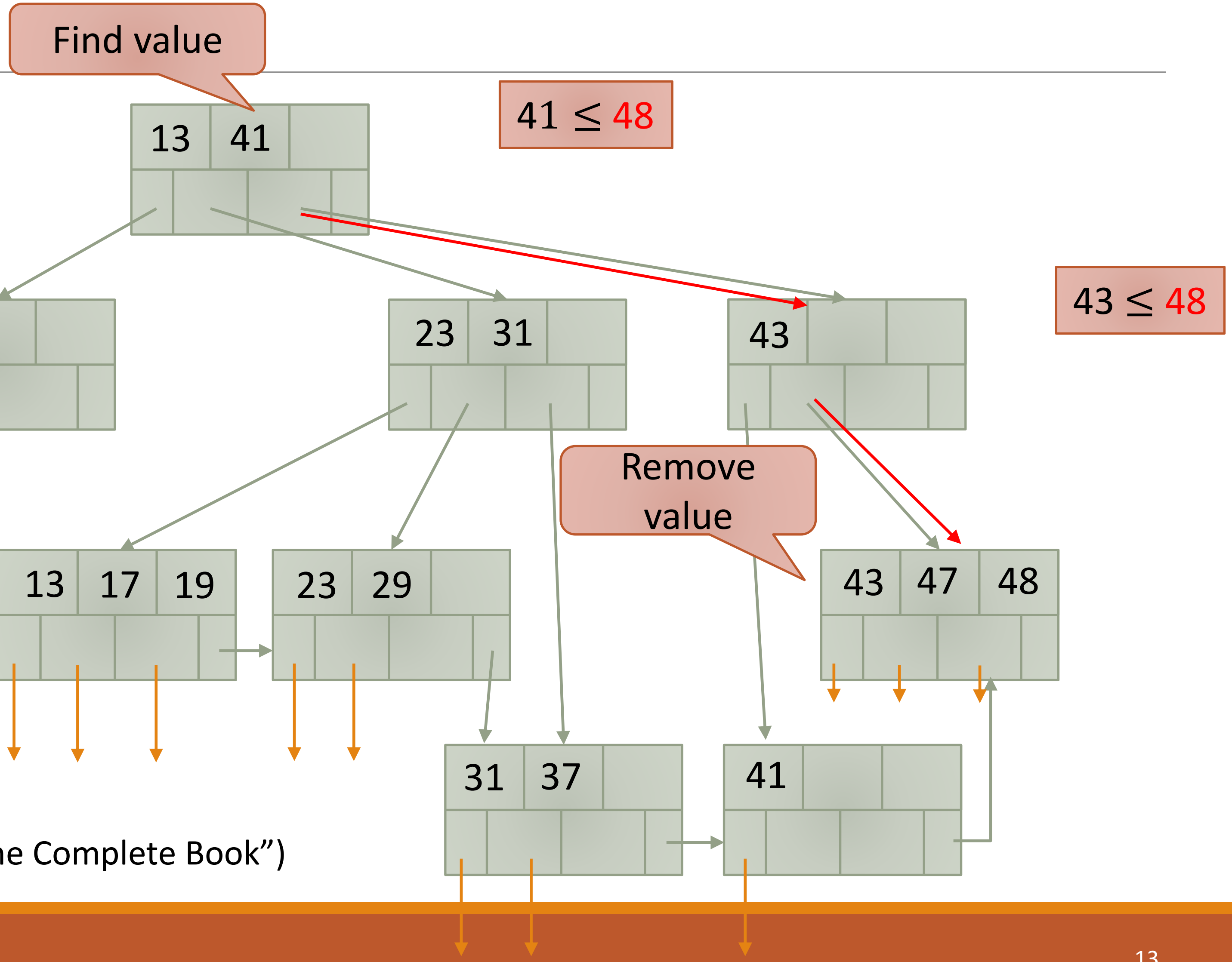


(Example from ”Database Systems: The Complete Book”)

Delete 48

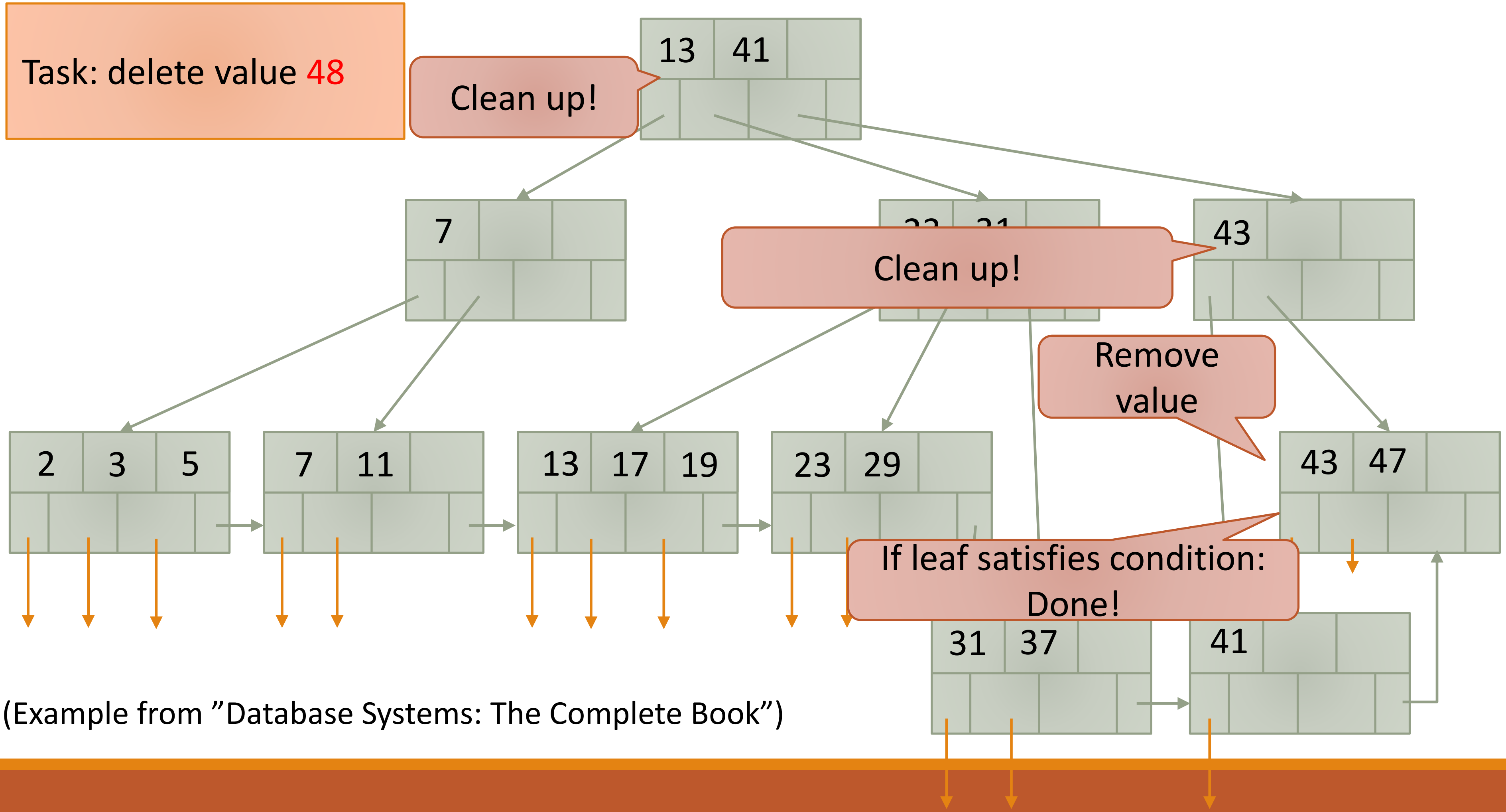
Deletion 2

Task: delete value 48



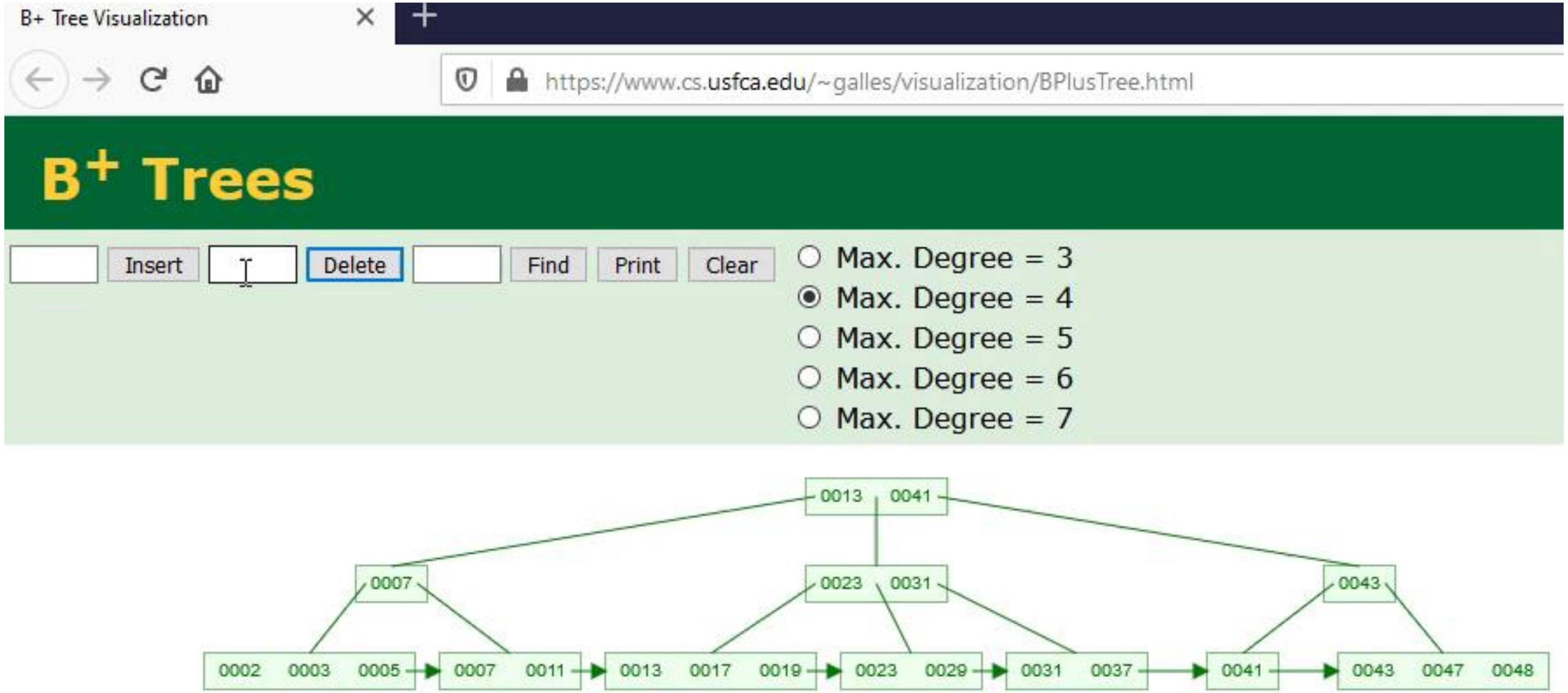
(Example from "Database Systems: The Complete Book")

Deletion 2



(Example from ”Database Systems: The Complete Book”)

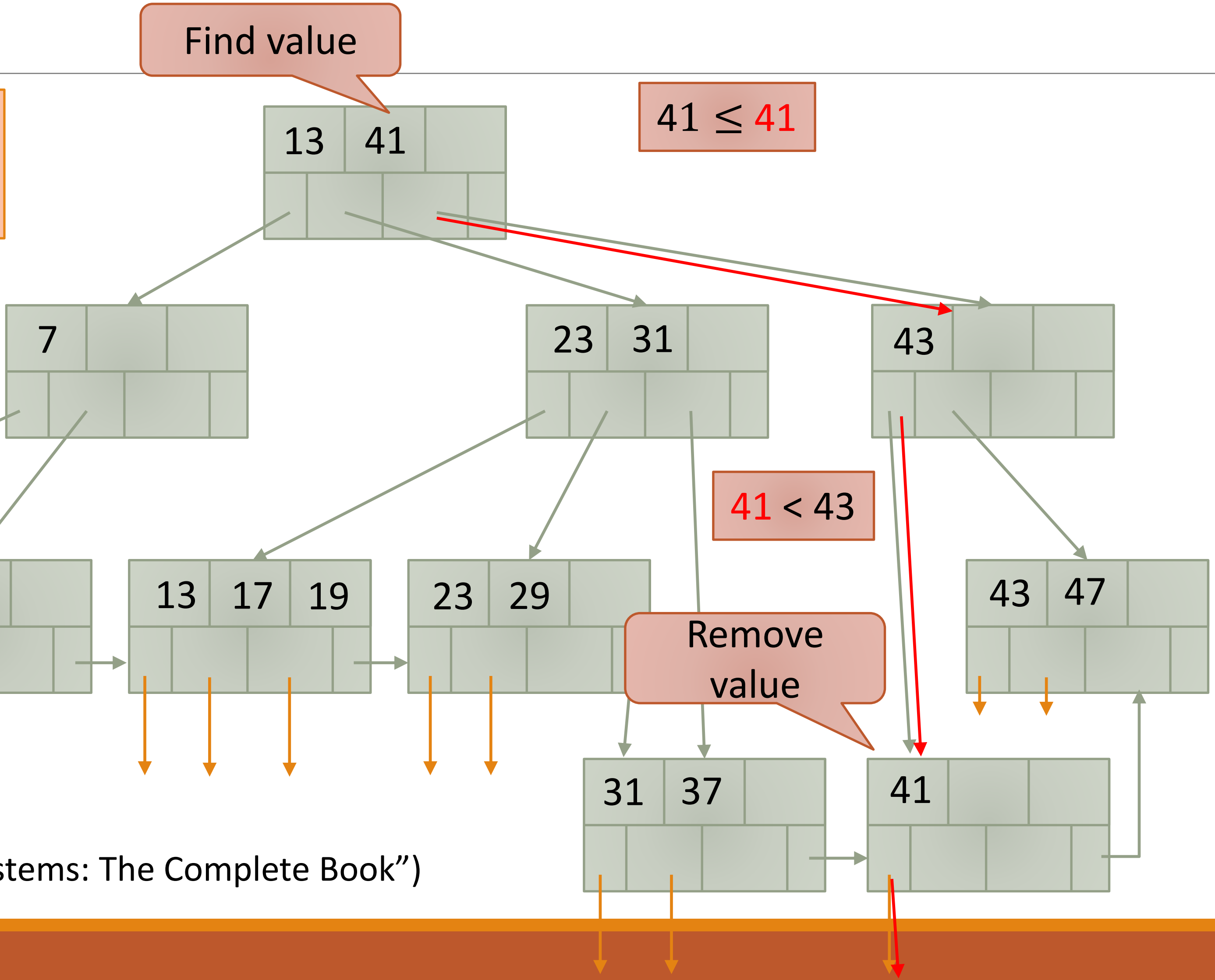
Online version of delete 48



Delete 41

Deletion 3

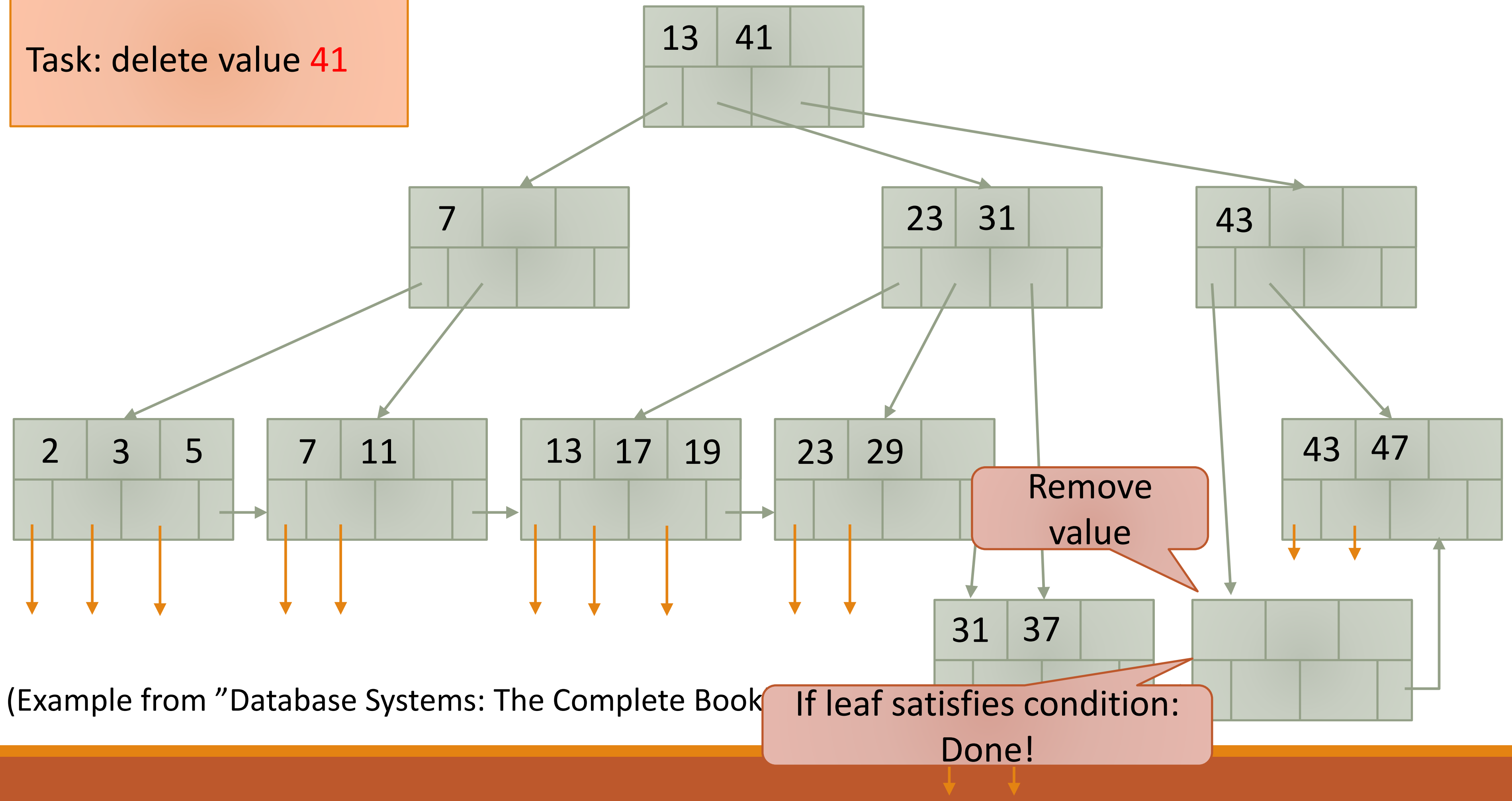
Task: delete value 41



(Example from "Database Systems: The Complete Book")

Deletion 3

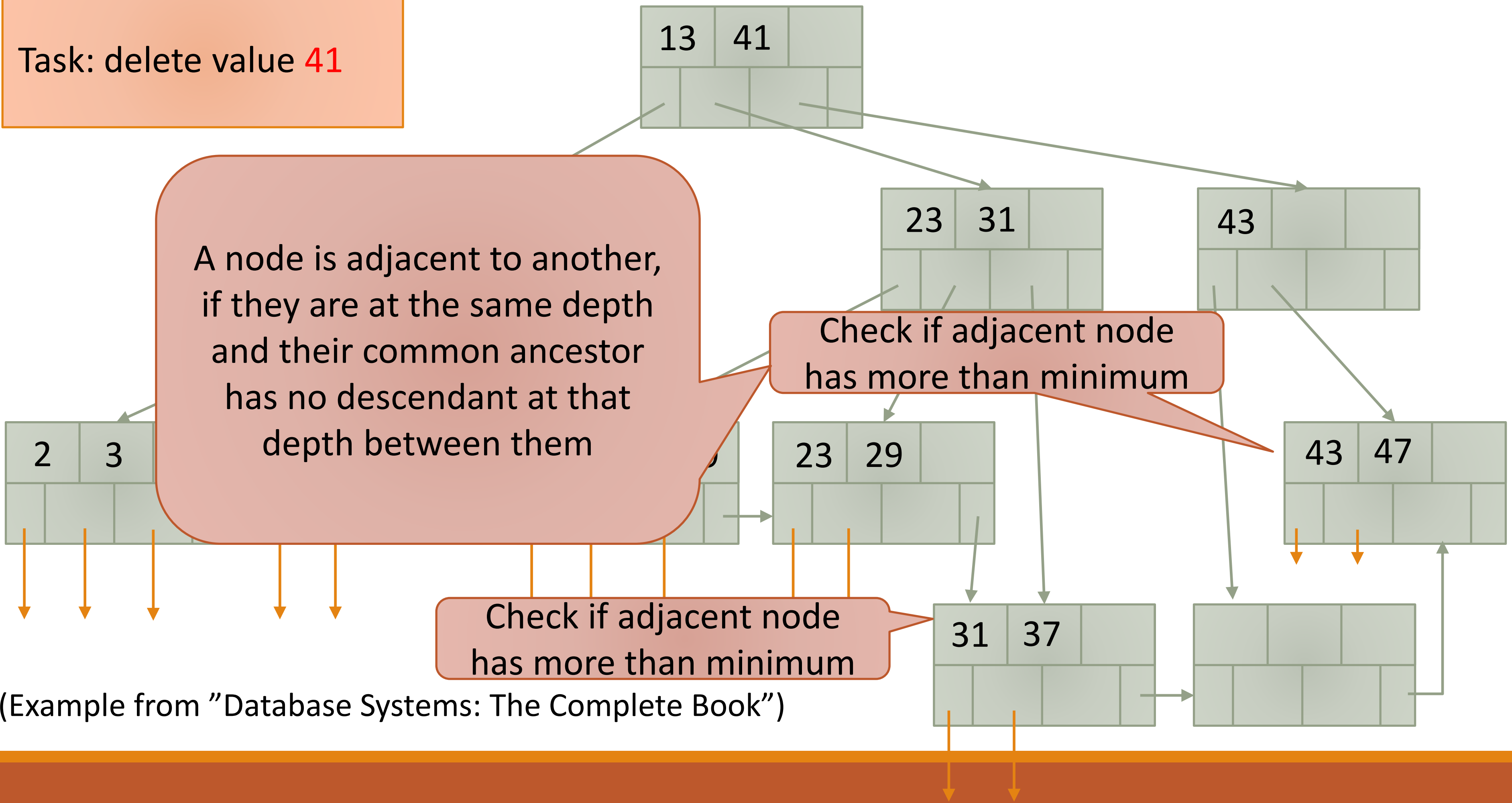
Task: delete value 41



(Example from ”Database Systems: The Complete Book

Deletion 3

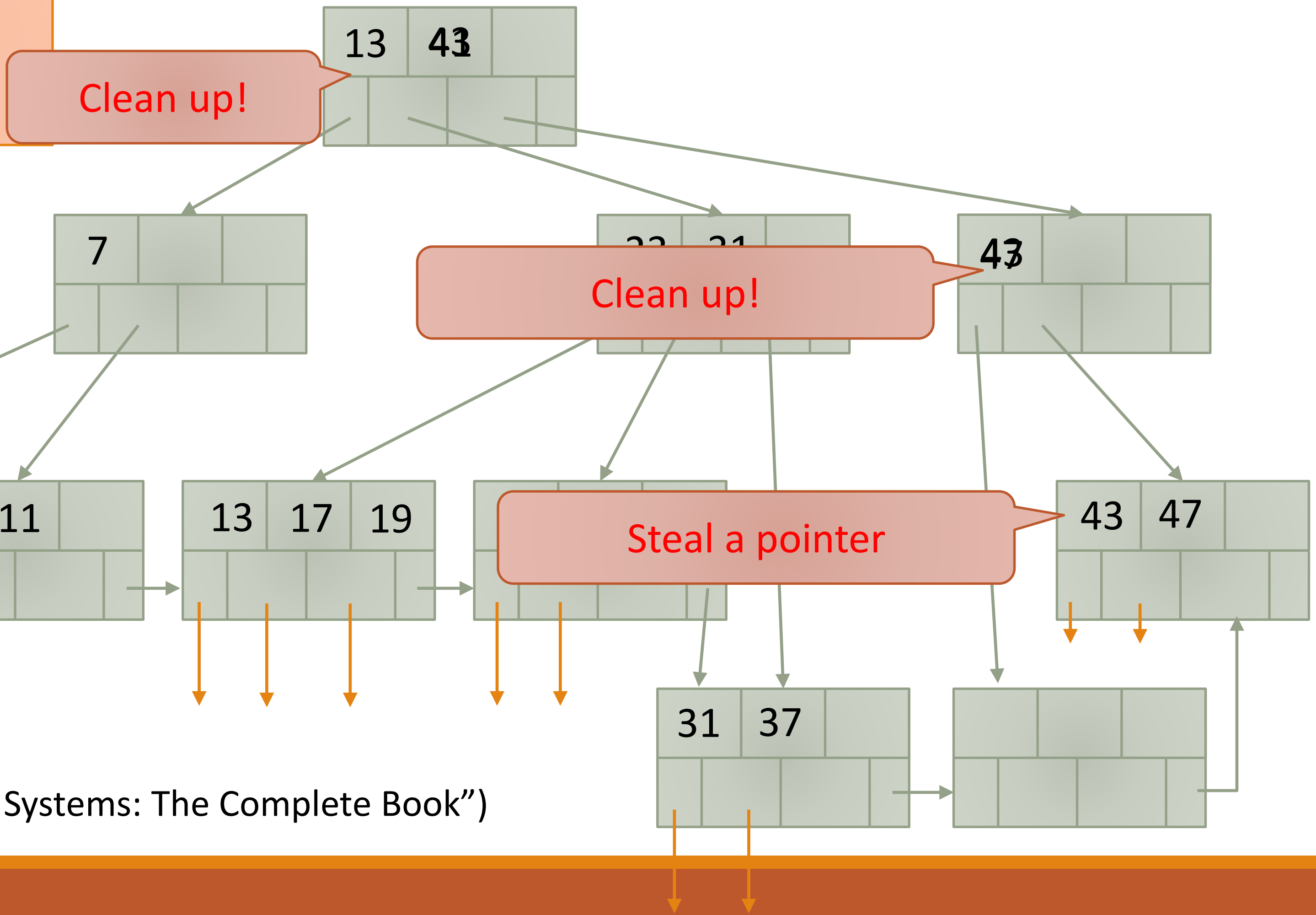
Task: delete value 41



(Example from ”Database Systems: The Complete Book”)

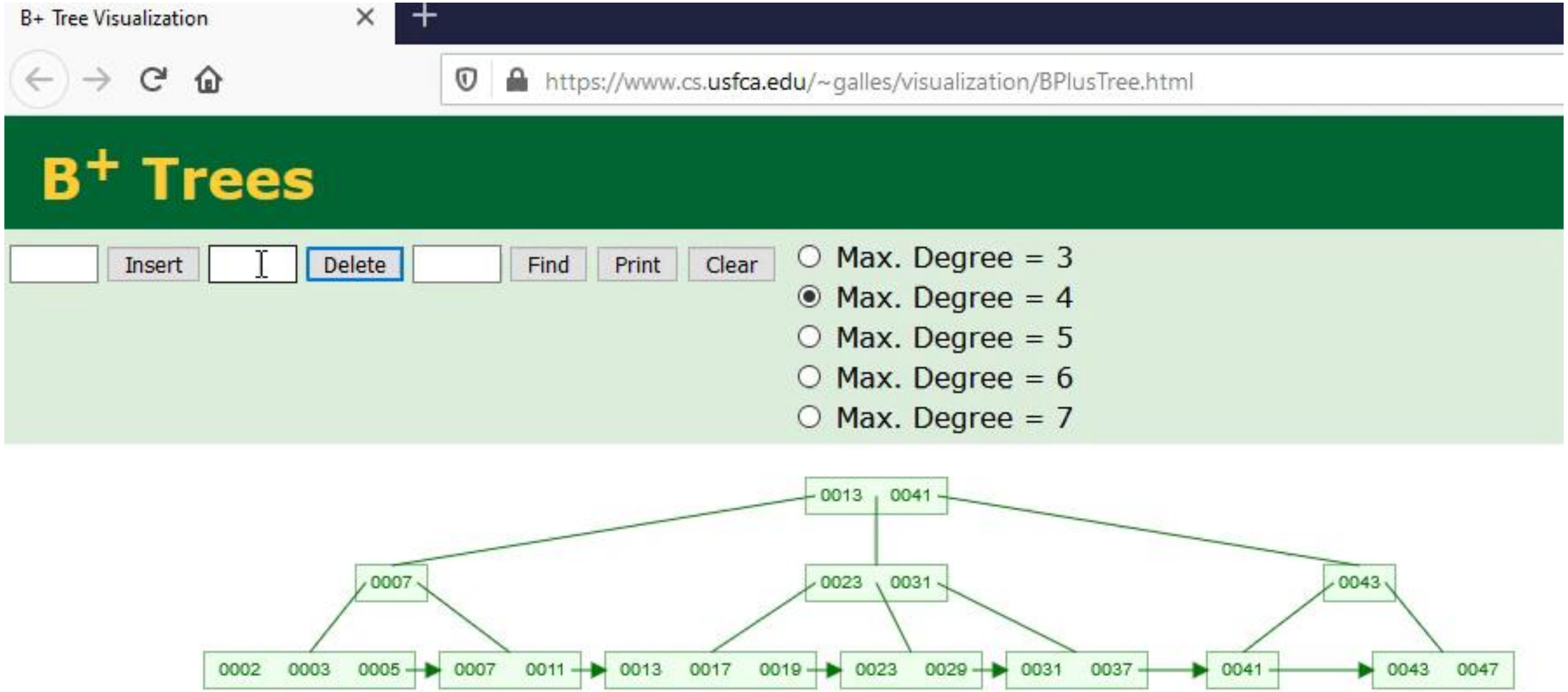
Deletion 3

Task: delete value 41



(Example from ”Database Systems: The Complete Book”)

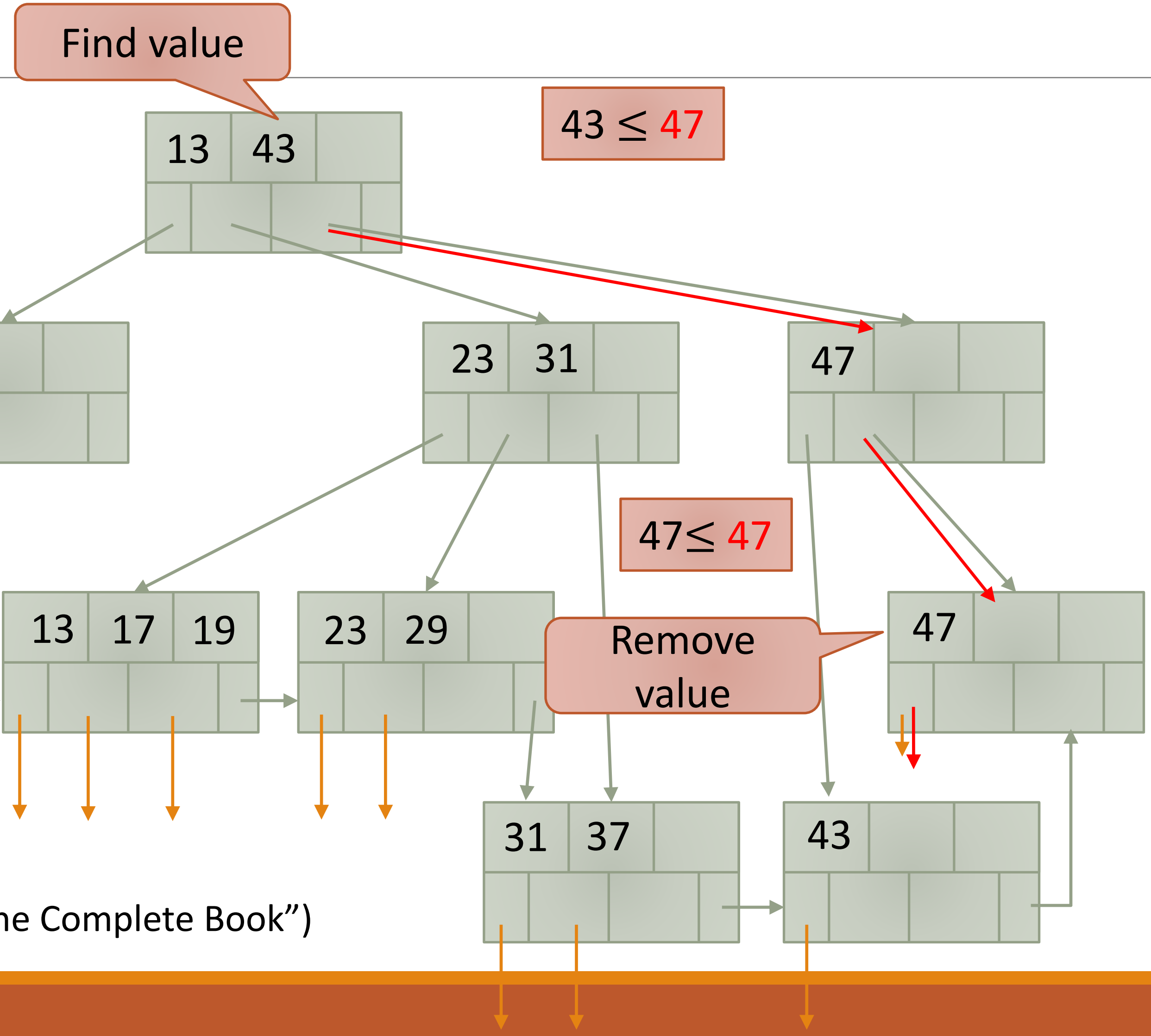
Online version of delete 41



Delete 47

Deletion 4

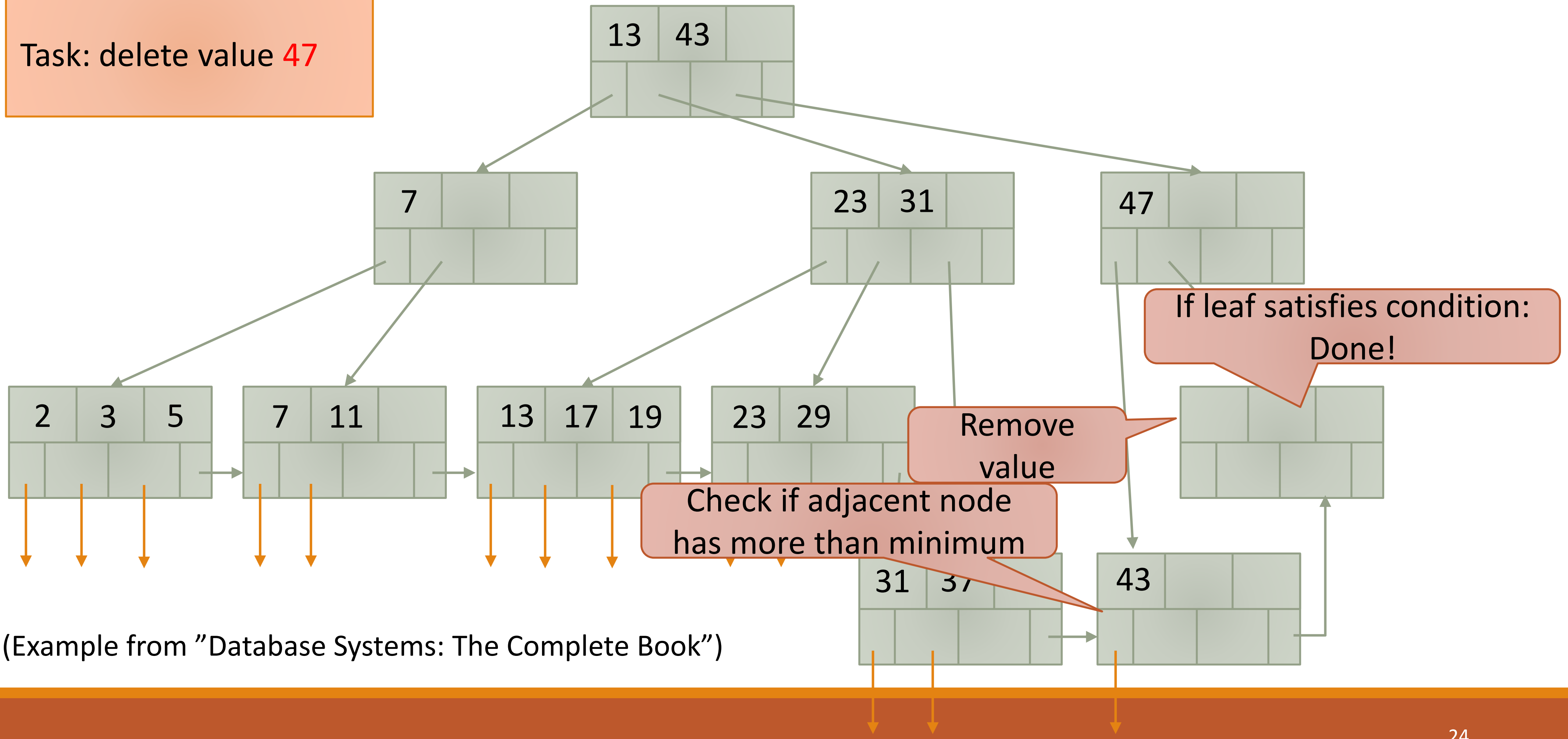
Task: delete value 47



(Example from ”Database Systems: The Complete Book”)

Deletion 4

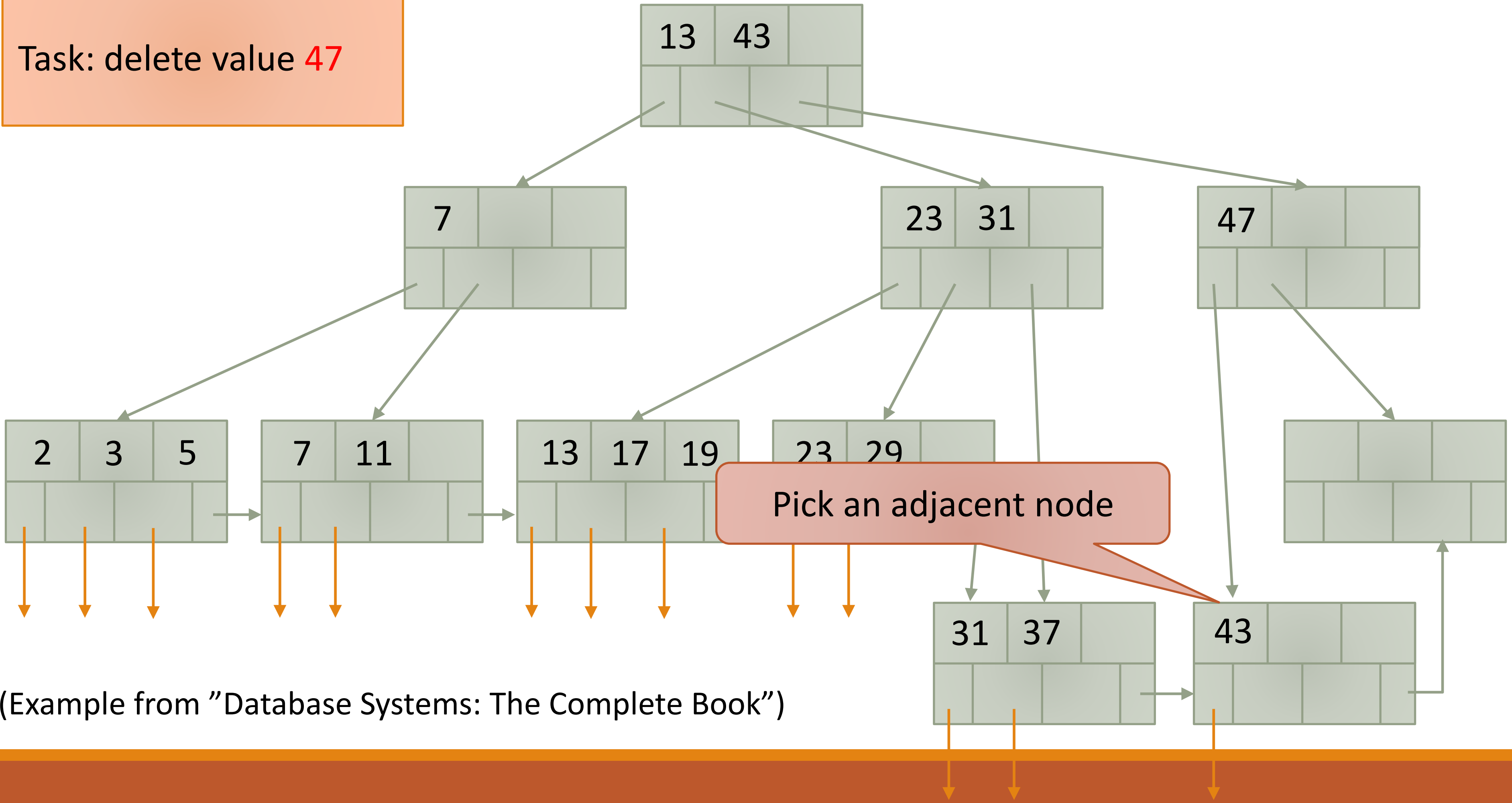
Task: delete value 47



(Example from "Database Systems: The Complete Book")

Deletion 4

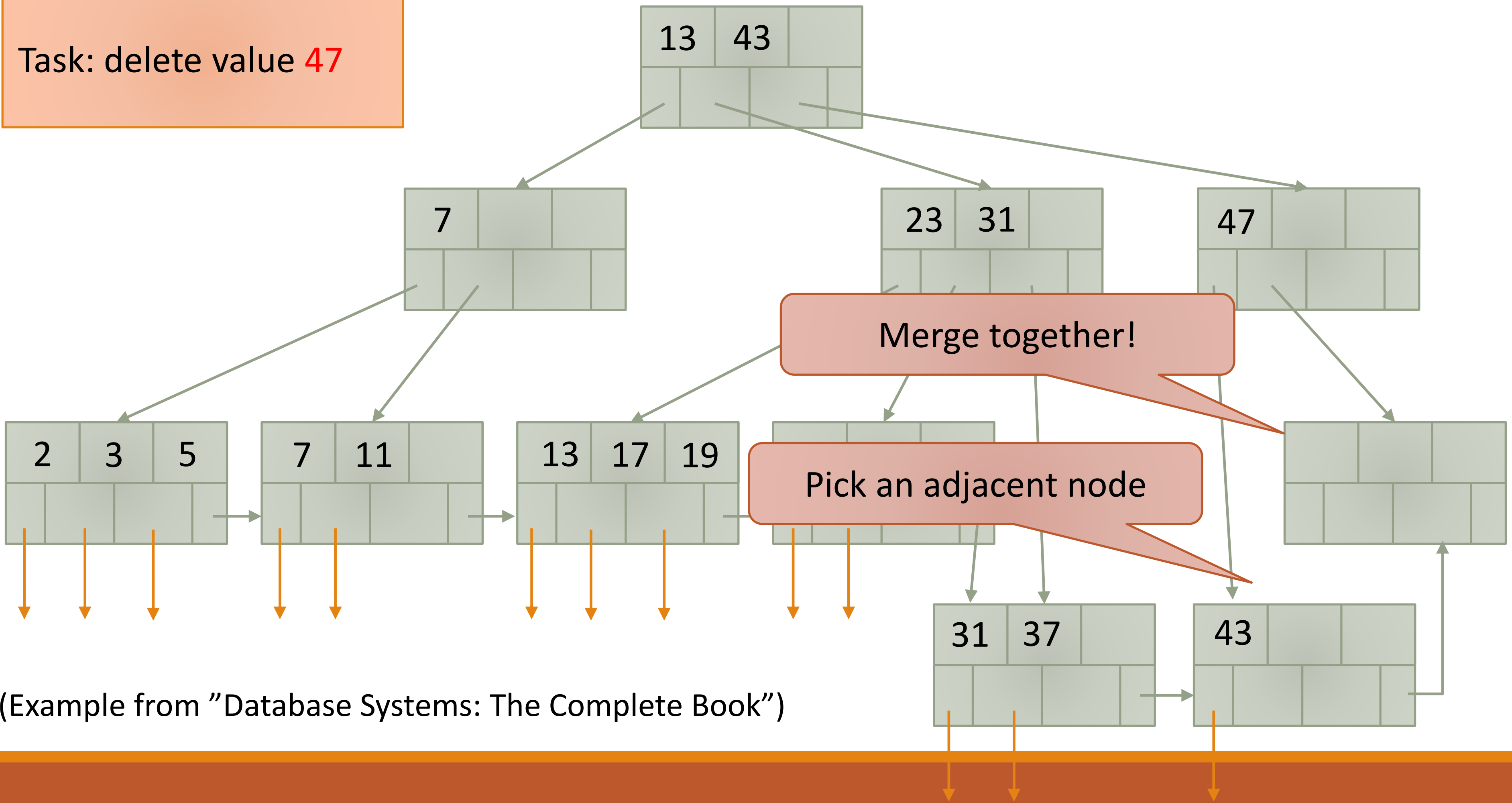
Task: delete value 47



(Example from "Database Systems: The Complete Book")

Deletion 4

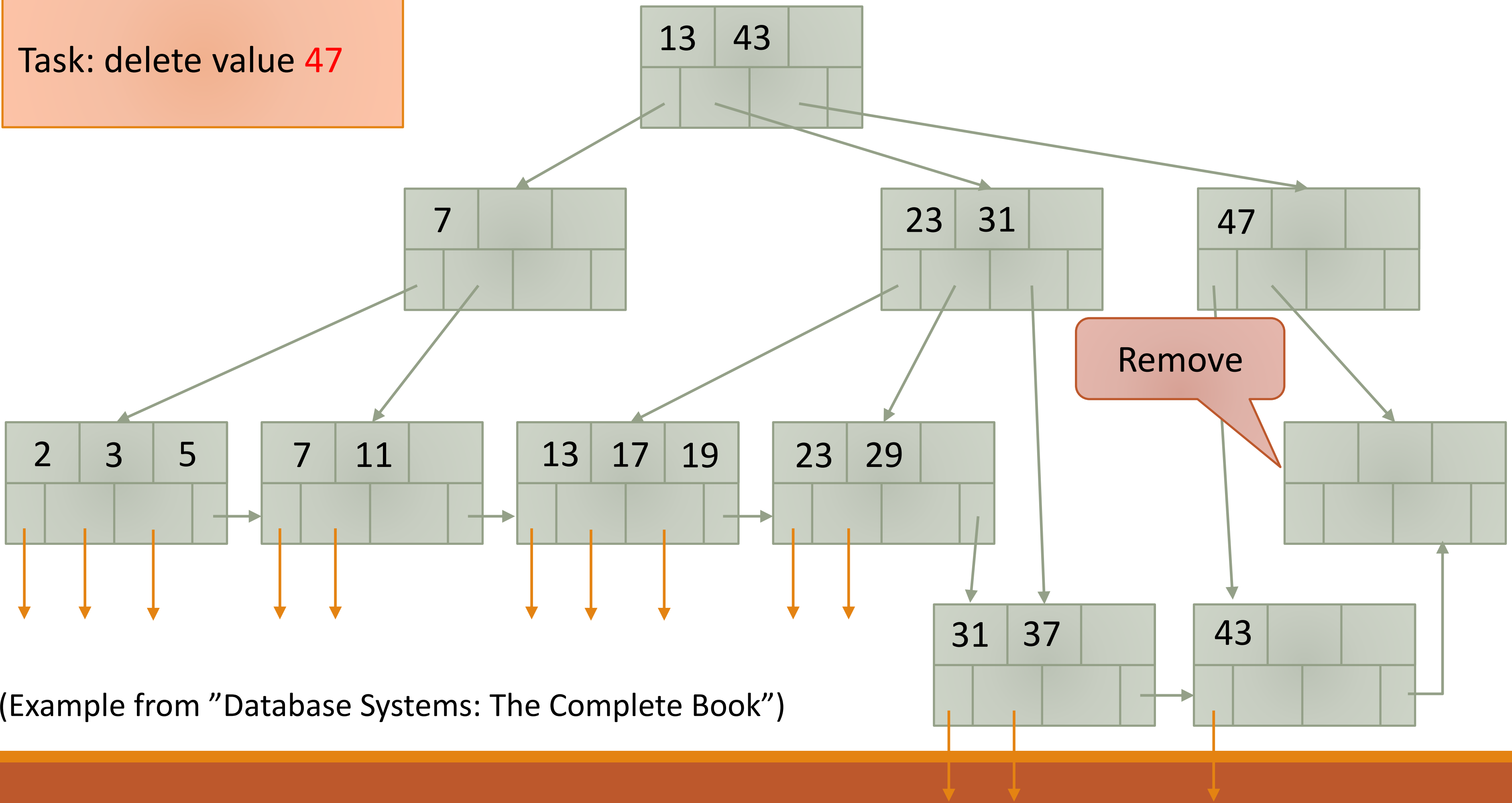
Task: delete value 47



(Example from ”Database Systems: The Complete Book”)

Deletion 4

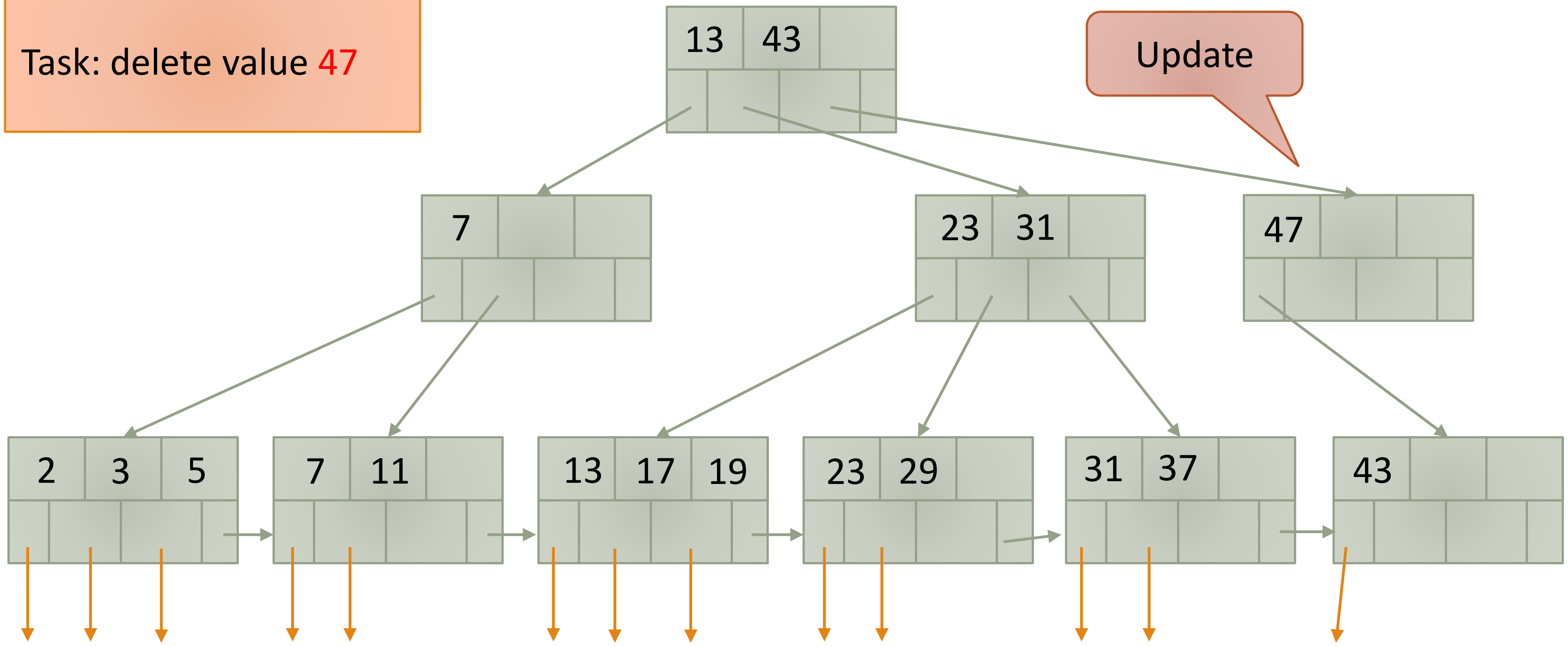
Task: delete value 47



(Example from ”Database Systems: The Complete Book”)

Deletion 4

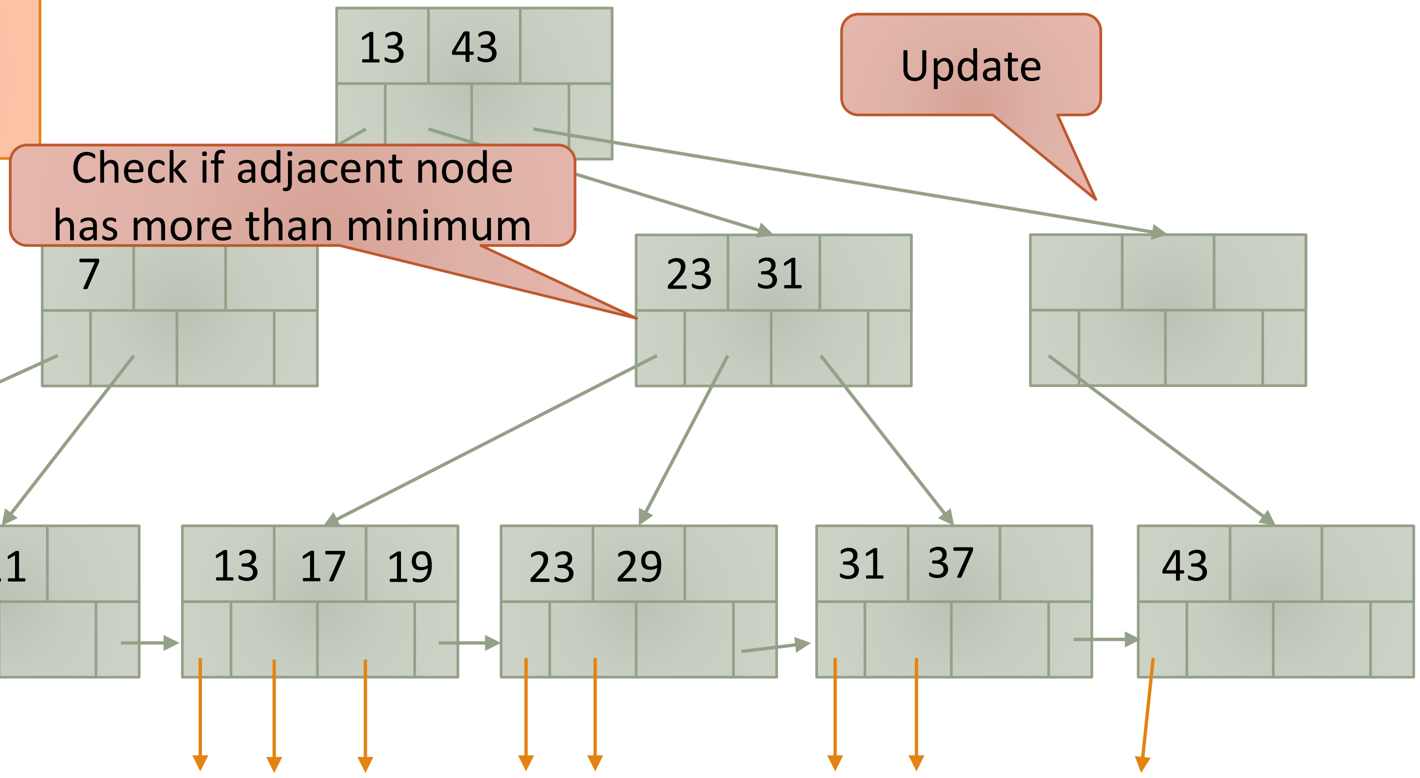
Task: delete value 47



(Example from "Database Systems: The Complete Book")

Deletion 4

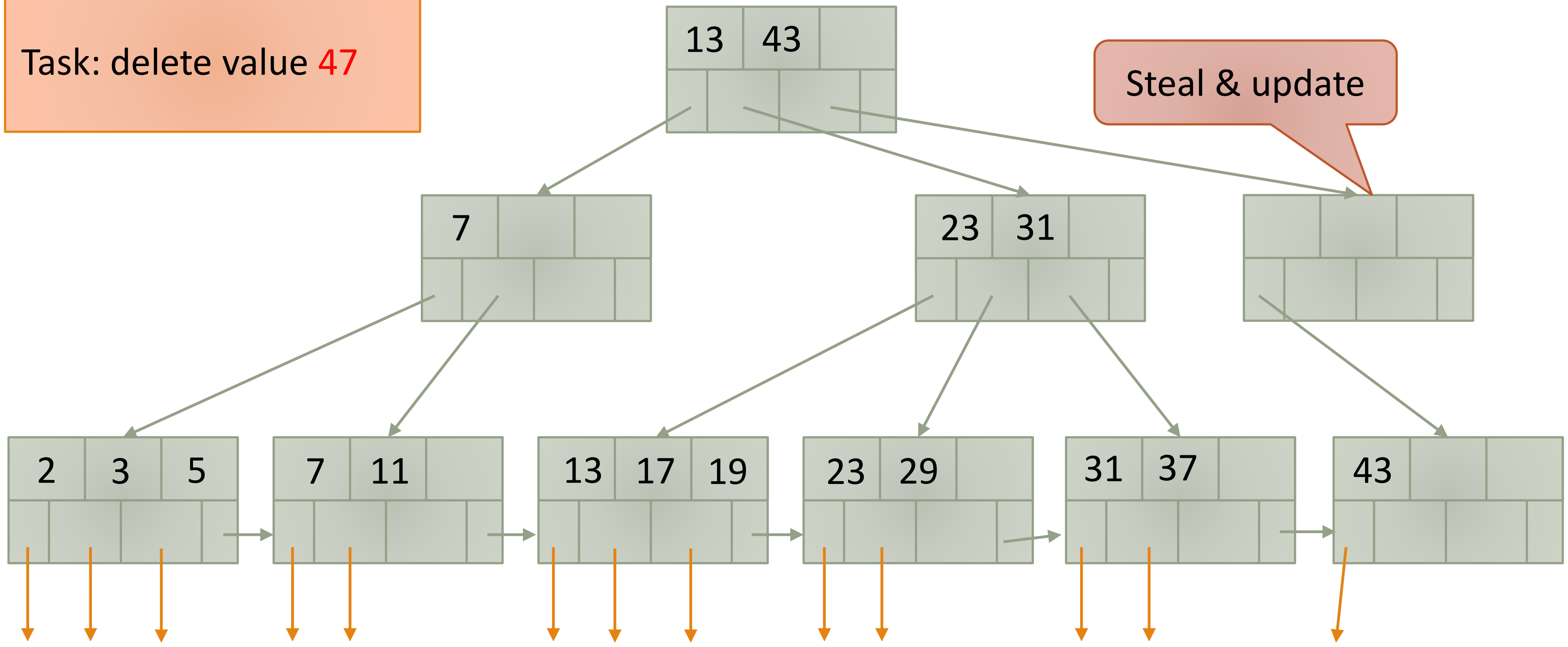
Task: delete value 47



(Example from "Database Systems: The Complete Book")

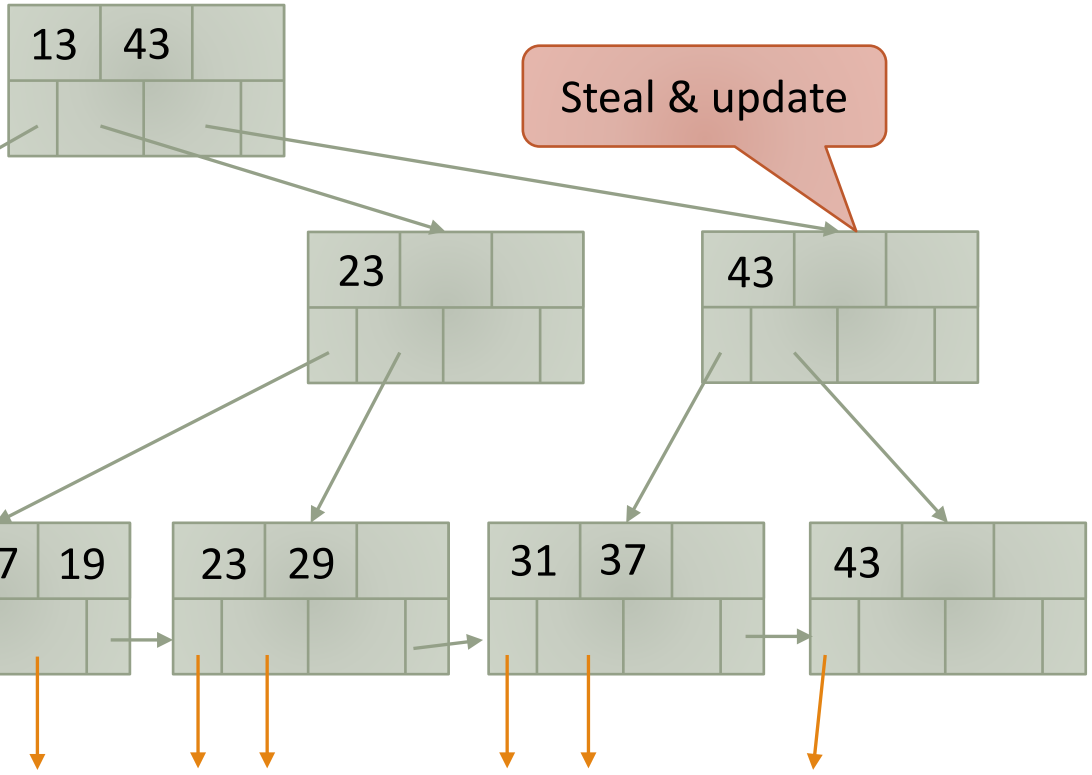
Deletion 4

Task: delete value 47



Deletion 4

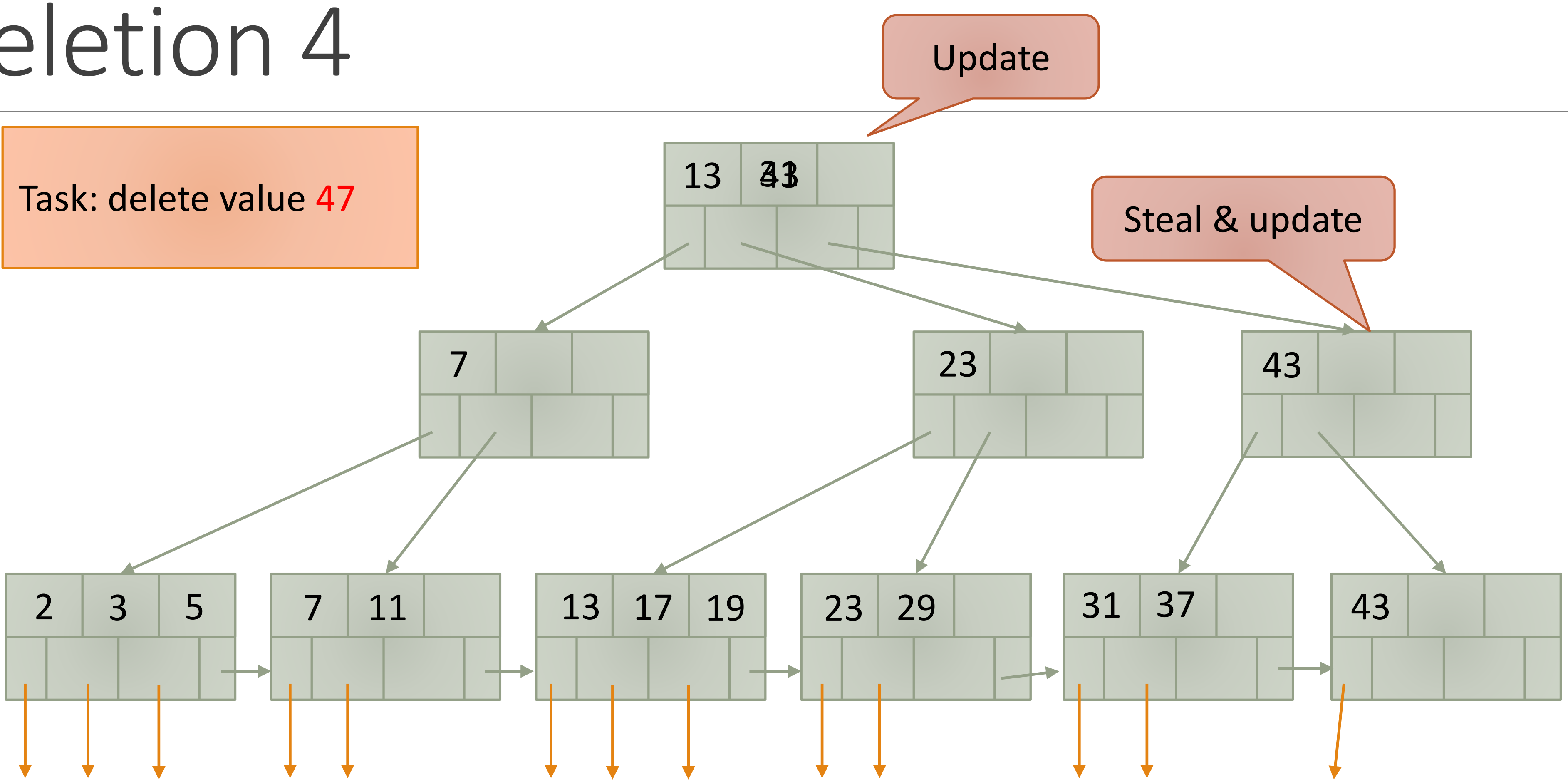
Task: delete value 47



(Example from ”Database Systems: The Complete Book”)

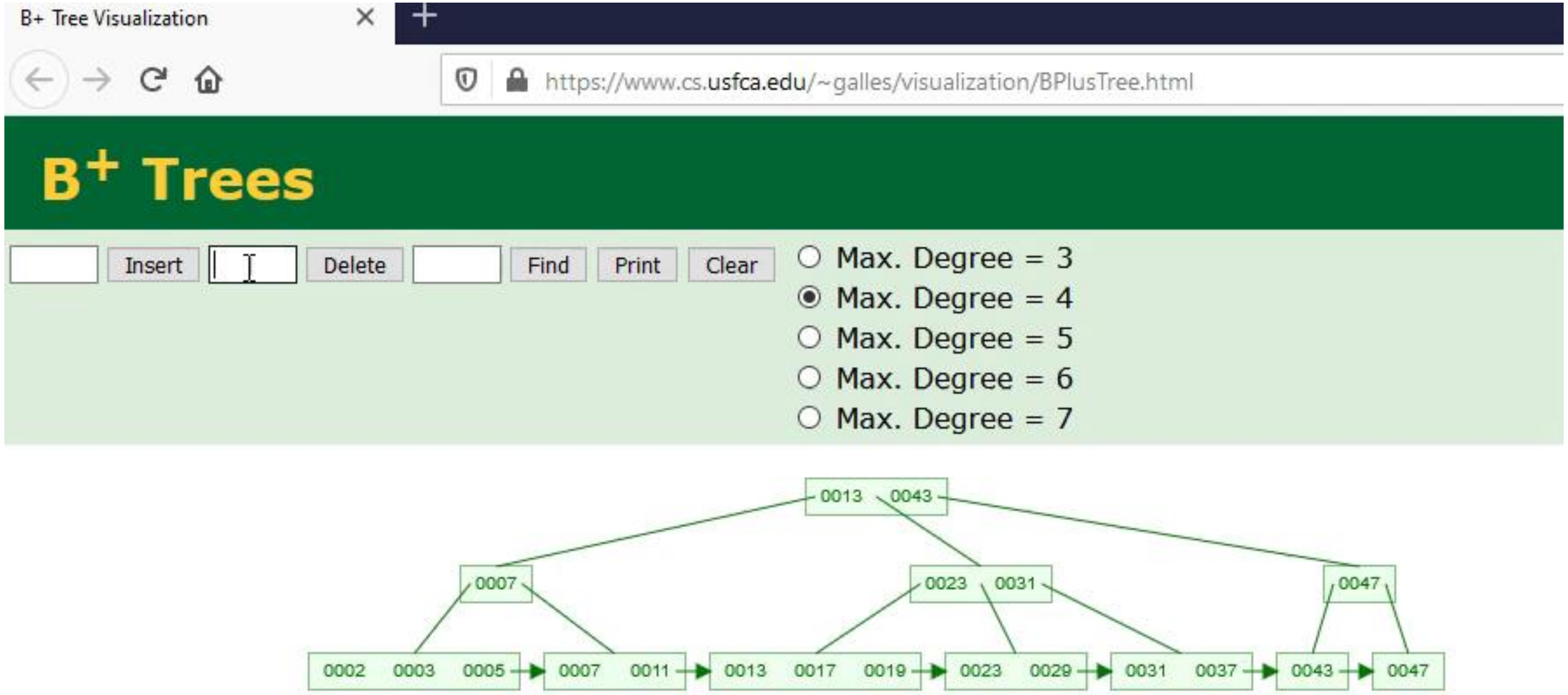
Deletion 4

Task: delete value 47



(Example from "Database Systems: The Complete Book")

Online version of delete 47



Review: Deletion

Goal: delete a value/pointer pair

Procedure:

- Find the leaf that should contain the value
  - If not there: Done
- Remove the value/pointer pair
- Let the current node C
- Let x be $\begin{cases} 2 & \text{if } C \text{ is root} \\ \left\lceil \frac{n+1}{2} \right\rceil & \text{if } C \text{ is internal node} \\ \left\lfloor \frac{n+1}{2} \right\rfloor & \text{if } c \text{ is leaf} \end{cases}$
- If C has above x pointers: Fix ancestors (if necessary) and you are done
- If C is the root but not a leaf: Remove it (and let the child of the root be the new root)
- Otherwise, check if an adjacent node has at least $x + 1$ pointers
- If so: take one, fix ancestors (if necessary) and you are done
- Otherwise, merge with sibling and go to line 3 with the parent as current node

See definition slide 19

The B+ tree **remains balanced**!

Running time: $O(h \times \log_2 n)$ "real" running time $O(h \times D)$

Time for a disk operation

Height of the B+ tree

Properties of B+ Tree Indexes

Fast lookups, insertions, deletions in time:

n ≈ B

$$O(\text{height of B+ tree} \times \log_2 n) = O(\log_n N \times \log_2 n)$$
$$= O(\log_2 N)$$

Remain **balanced**

Huge capacity even with height 3 if blocks large enough

- Block size: 16386 bytes (16 kilobytes)
- Values stored in index: 4 bytes
- Pointers: 8 bytes
- Largest n so that each B+ tree node fits into a block (i.e., $4n + 8(n+1) \leq 16386$) is 1364
- B+ trees with height 3 can store > $n^3 = 2.5 \cdot 10^9$ values

Properties of B+ Tree Indexes

Can be implemented **efficiently with respect to number of disk accesses**

- Number of disk accesses typically ≈ 2 + height of B+ tree

Most of the B+ tree can be kept in memory

- Specifically, the upper levels
- Even with block size of 16384 bytes and n = 1364:
  - Level 1 (root) ≈ 16 KB
  - Level 2 (children of root) ≈ (n+1) × 16 KB ≈ 21 MB
  - Level 3 ≈ (n+1) × (n+1) × 16 KB ≈ 28 GB
  - Level 4 ≈ (n+1) × (n+1) × (n+1) × 16 KB ≈ 38 TB

Typically, these are the leaf nodes and can be loaded from disk on demand

Summary

We can do each operation (search, insertion and deletions) in a B+-tree in time

- O(height of B+ tree $\times \log_2 n$) or
- O(height of B+ tree $\times$ D)

Slide 34 contains a summary of how to do deletions