Democratic and Popular Republic of Algeria

Ministry of Higher Education and Scientific Research

Ecole supérieure en sciences et technologies de l'informatique et du numérique

B-trees

Presented by : Dr. Daoudi Meroua

Academic year: 2024/2025

B-trees

B-trees

It is one of the **most efficient** access methods known to date for large and highly dynamic files.

Files of the completely balanced m-ary search tree type:

- Same block structures and same declarations.
- Same search mechanisms

Properties :

A B-tree of order **N** is an **m-ary search tree** of order N that satisfies the following properties:

a) All nodes (except the root) must be filled to **at least 50% of their capacity** (minimum degree = $\lfloor \mathbf{N/2} \rfloor$).

b) The **root node** can contain a **minimum of only one value** and two children.

c) All **leaves** are at the **same level.**

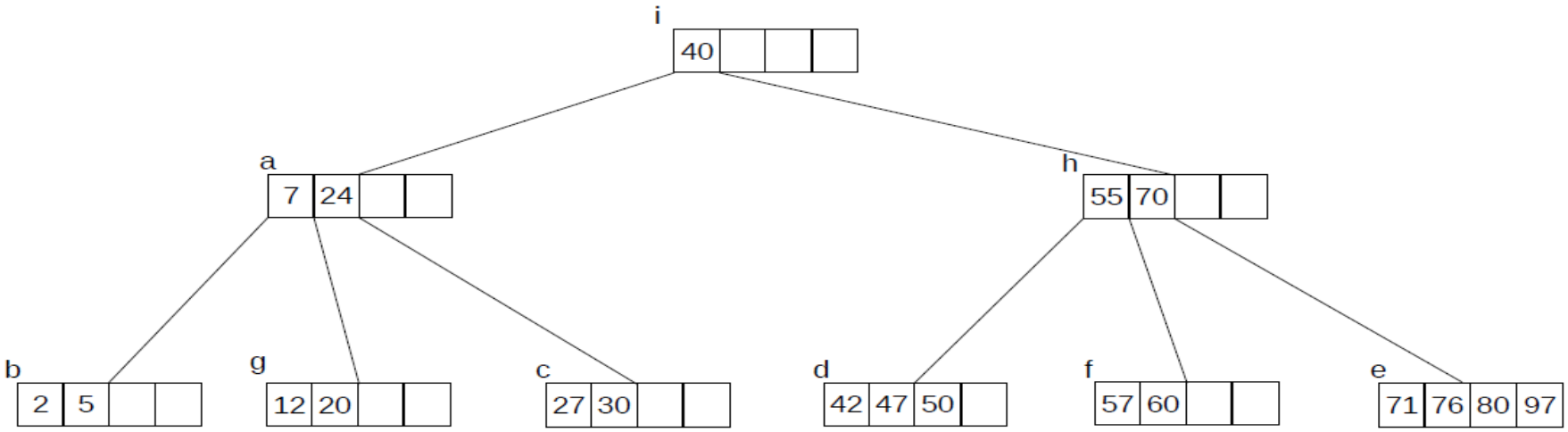
d) In an **internal node**, **all children** (Child1, Child2, ... Childdegree) are **different from -1**, and in a **leaf node**, **all children** (Child1, Child2, ... Childdegree) **are -1.**

B-trees

Example of a B-tree of Order 5

The maximum capacity of a node is: 4 values (and 5 children).

Except for the root, the minimum capacity of a node is: 2 values (and thus 3 children).



B-trees : insertion

Inserting a New Value x:

1. Search for x:
- If x already exists, go to Step 4 (End).
- Otherwise, let P be the last visited node (a leaf), and proceed to Step 2.

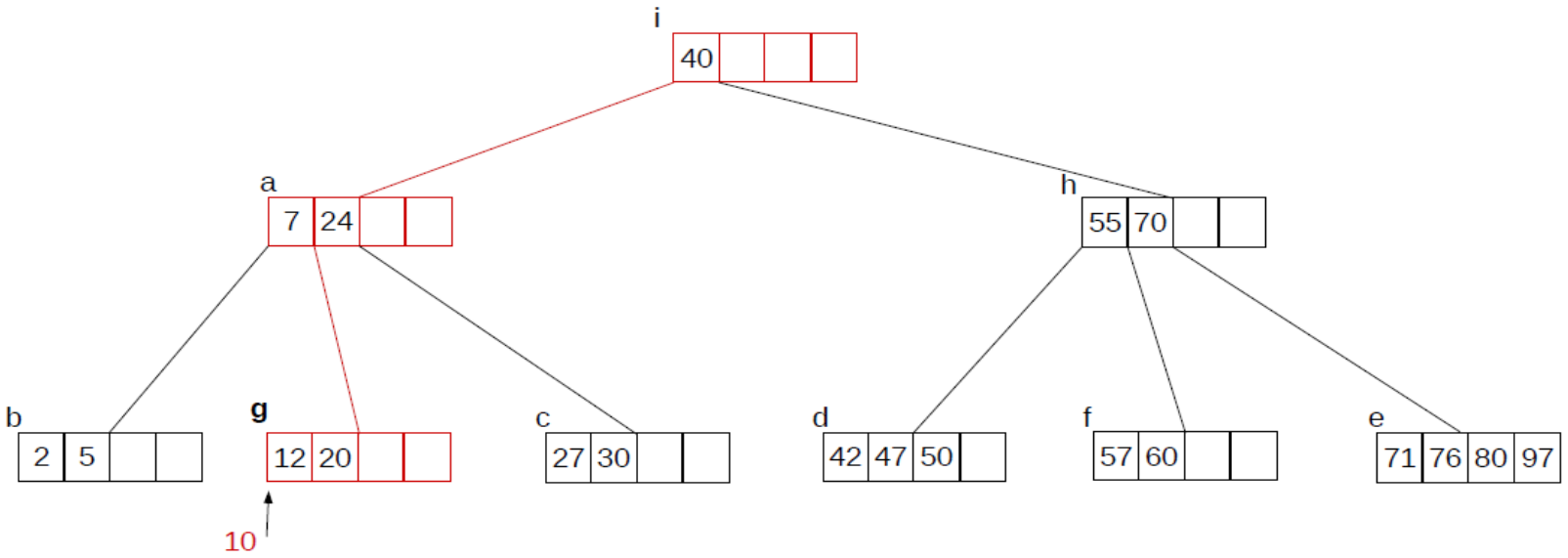
2. if (P is not full):
- Insert x into P (using internal shifts as needed).
- Go to Step 4 (End).

3.If P is full, split P into two nodes:
- Allocate a new node ⇒ Q.
- P: Will contain the first half of the values.
- Q: Will contain the second half of the values.
- Let m be the middle value separating the two halves.

//Now insert m into the parent of P:
- Set x ← m, P ← parent(P).
- If (P == nil), allocate a new root node for P.
- Go to Step 2.
- End.

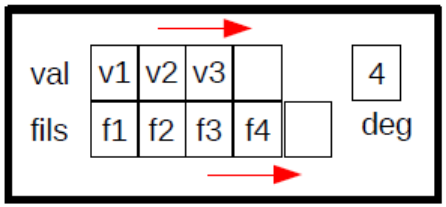
B-trees : insertion

Inserting the value 10:
The search for 10 leads to the leaf node g (position = 1 in g).
Since g is not full, the insertion is done using internal shifts.

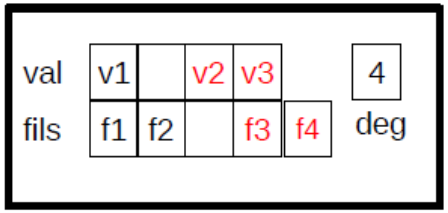


B-trees : insertion

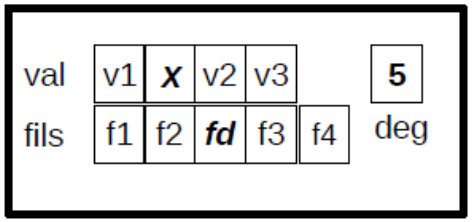
Insertion of x at position 2 (and its right child fd) using internal shifts in a full node.



After internal shifting (values and children).



After inserting x (and its right child fd) and updating the degree.



B-trees : insertion

Result of inserting the value 90:

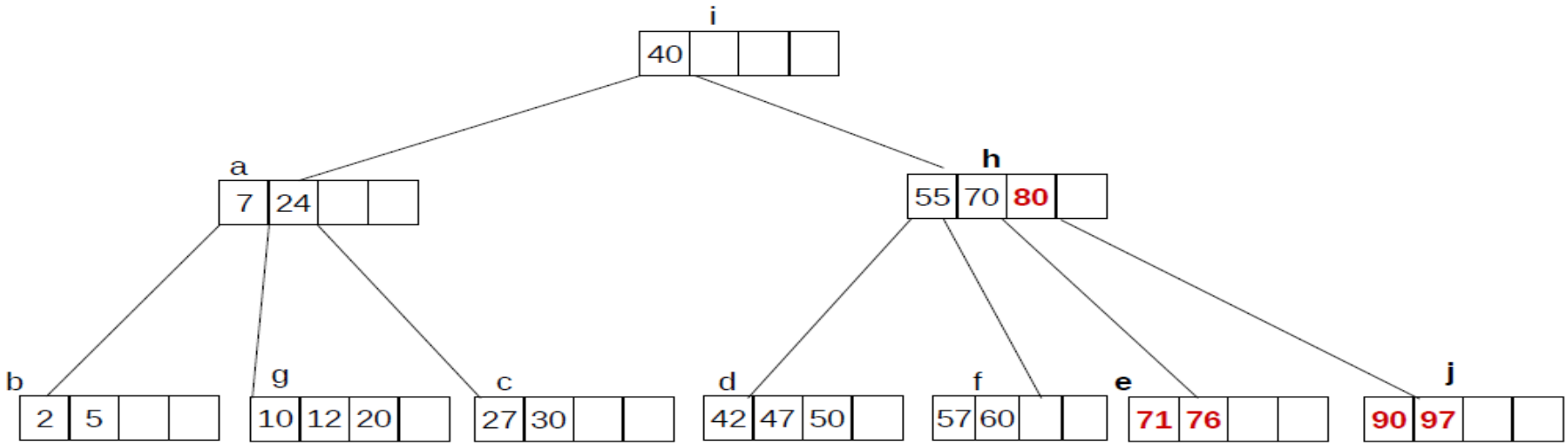
The search for 90 leads to the leaf node e (position = 4 in e).

Since e is already full, the node will be split:

A new node (j) is allocated.

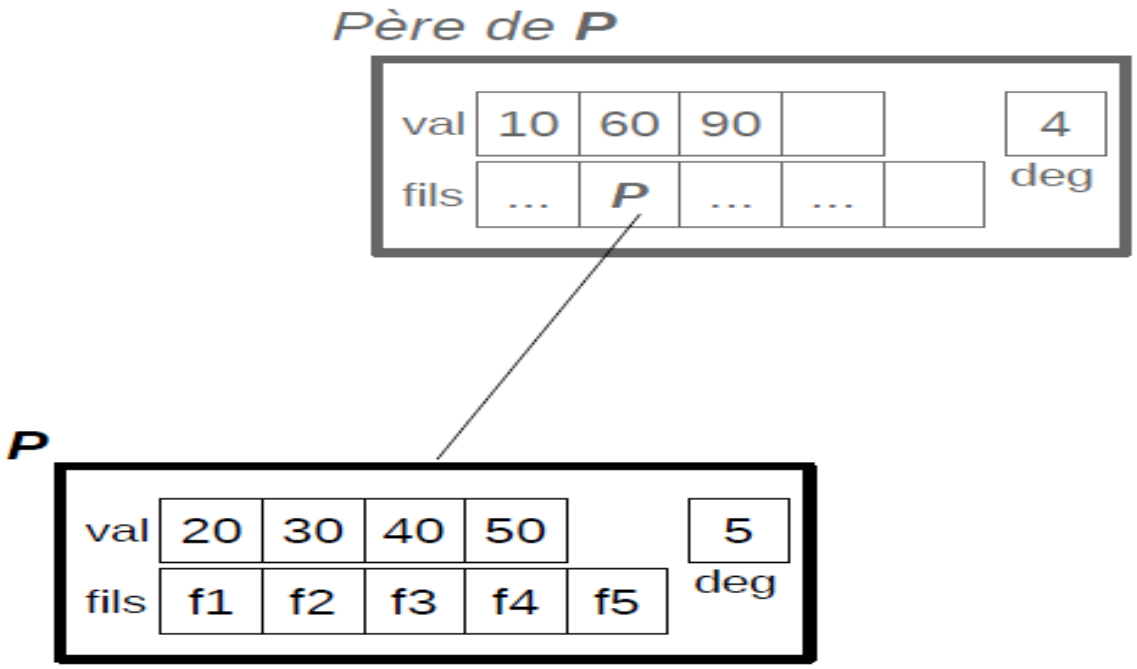
The ordered sequence is formed: 71, 76, 80, 90, 97.

The sequence is split into: [71, 76], [80], [90, 97].



B-trees : insertion

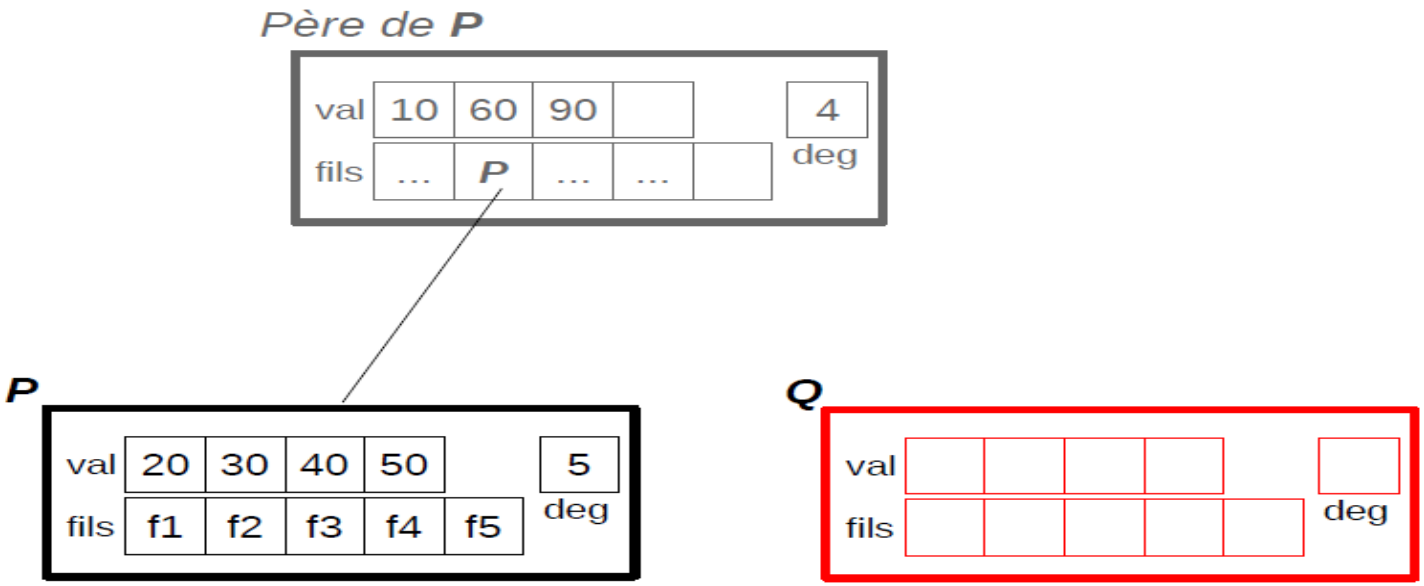
Insertion of x = 25 (and its right child fd), through splitting, in a full node P:



B-trees : insertion

Insertion of x = 25 (and its right child fd), through splitting, in a full node P:

Allocation of a new block: Q.

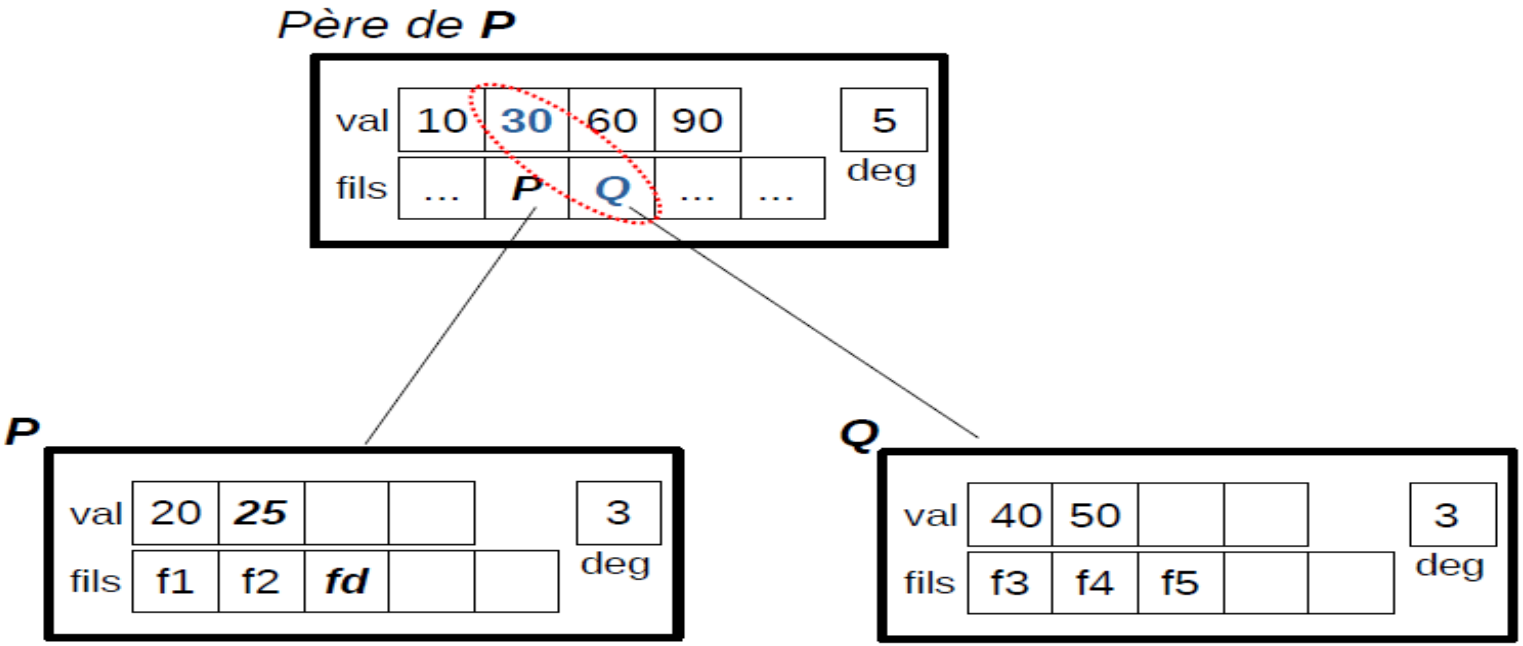


B-trees : insertion

Insertion of x = 25 (and its right child fd), through splitting, in a full node P:

Splitting the sequence between:P (first half) and Q (second half)

[20,25,30,40,50]



B-trees : Deletion

Deletion of c (Del(c)):

1. Search(c) → (i, j) with stacking of the visited nodes.
2. If i is an internal node:
   - Replace c with its in-order successor c' (which must be in a leaf).
   - Let (i, j) be the address of c'.
3. Delete the value at position j (and its right child) by internal shifts in i.
4. If i becomes underfilled (underflow):

If one of i's siblings is more than 50% full (contains more than the minimum):

➢ Redistribute with this sibling.

Otherwise: // The siblings are therefore filled to the minimum.

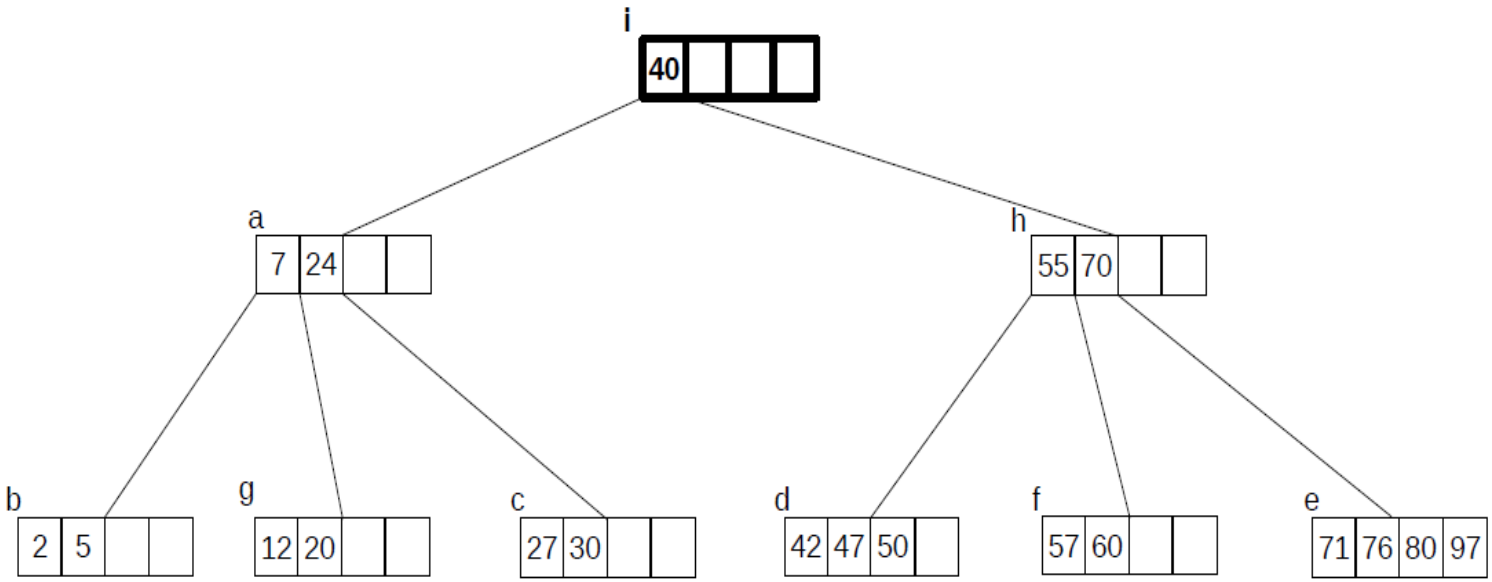
➢ Fusion:

- Merge i with one of its siblings.
- Remove the middle value from the parent:
- (Unstack <i, j ...>; go to Step 3).

B-trees : Deletion

Example 1: Deletion of 40

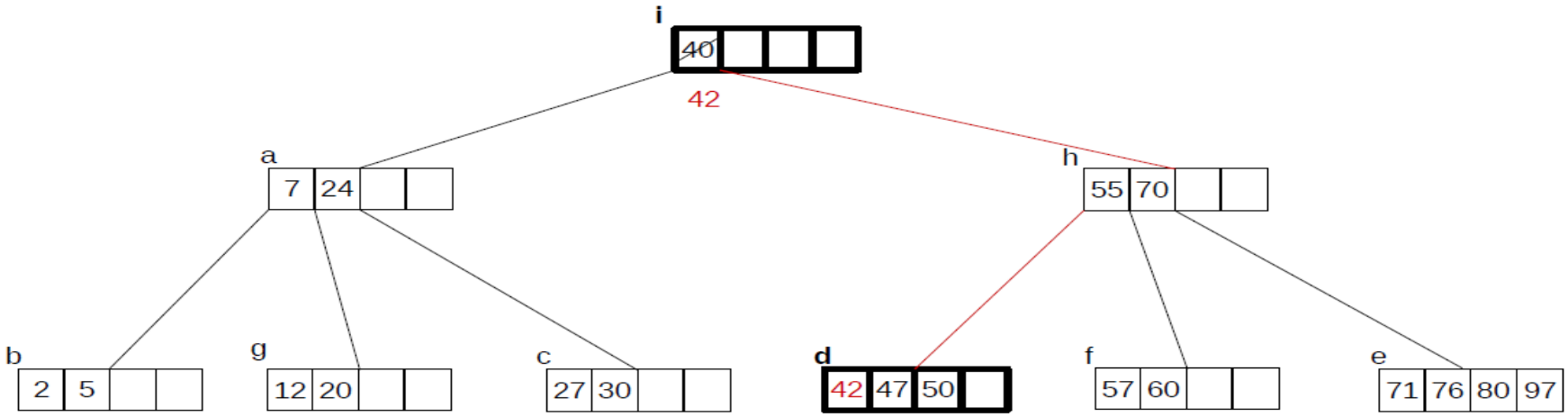
1. The search for 40 ends at (i, 1).



B-trees : Deletion

Example 1: Deletion of 40

2. Since it is an internal node, 40 in (i, 1) is replaced by its in-order successor, 42 → (d, 1).

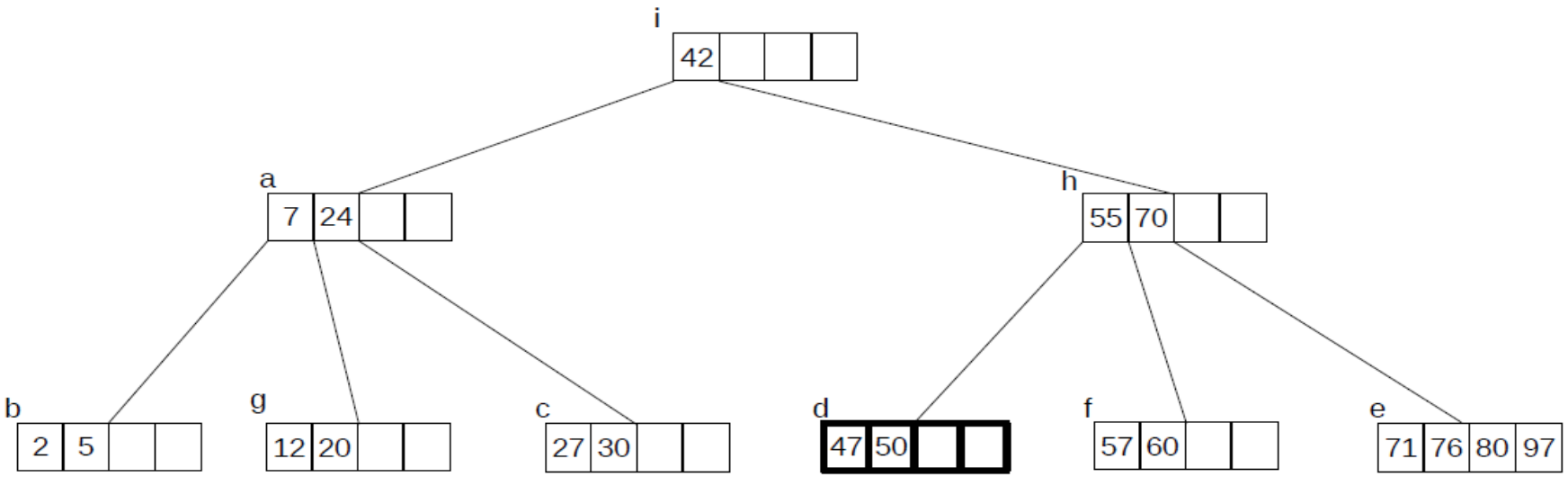


B-trees : Deletion

Example 1: Deletion of 40

3. 42, i.e., (d, 1), is deleted using internal shifts.

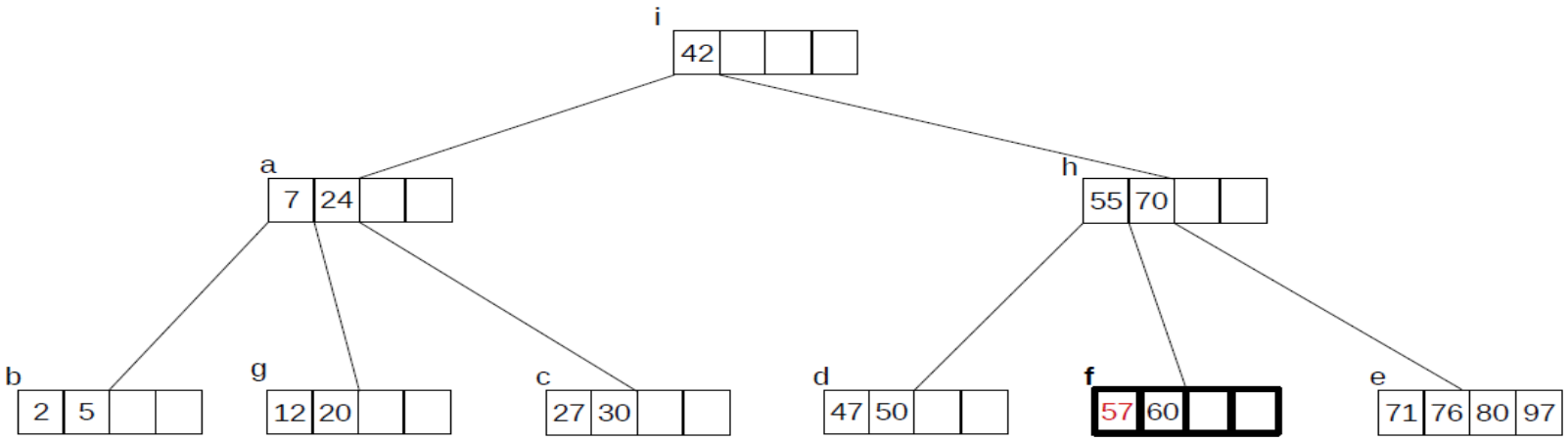
End of the deletion of 40.



B-trees : Deletion

Example 2: Deletion of 57

1. The search for 57 ends at (f, 1).



B-trees : Deletion

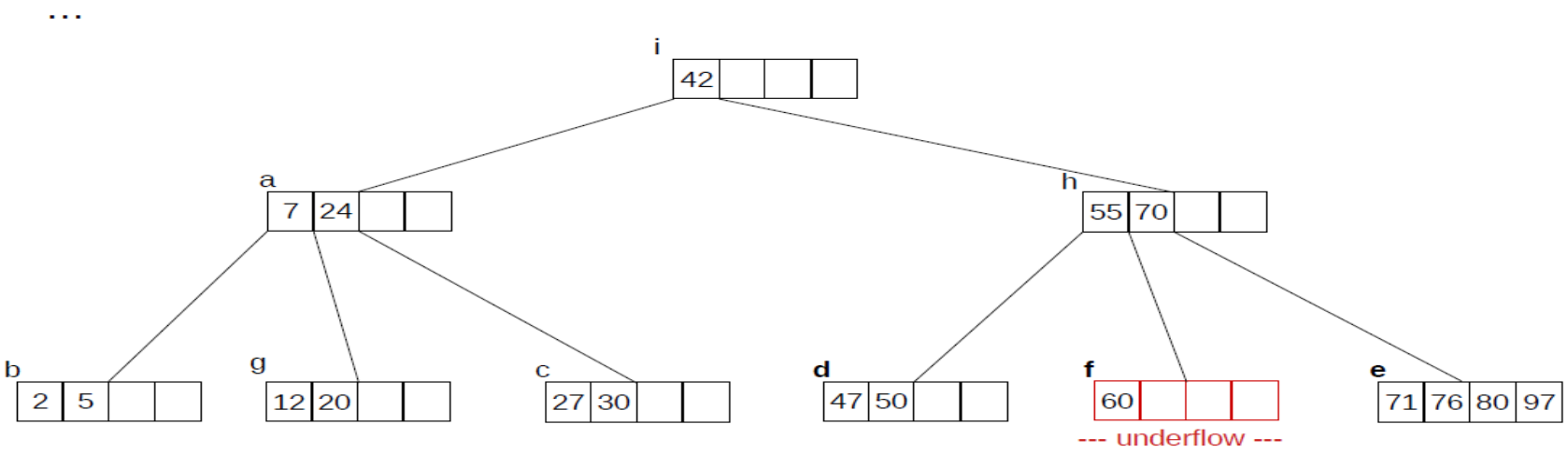
Example 2: Deletion of 57

2. Since f is a leaf node, 57 is deleted using internal shifts.

4. The node f becomes underfilled (in a state of underflow).

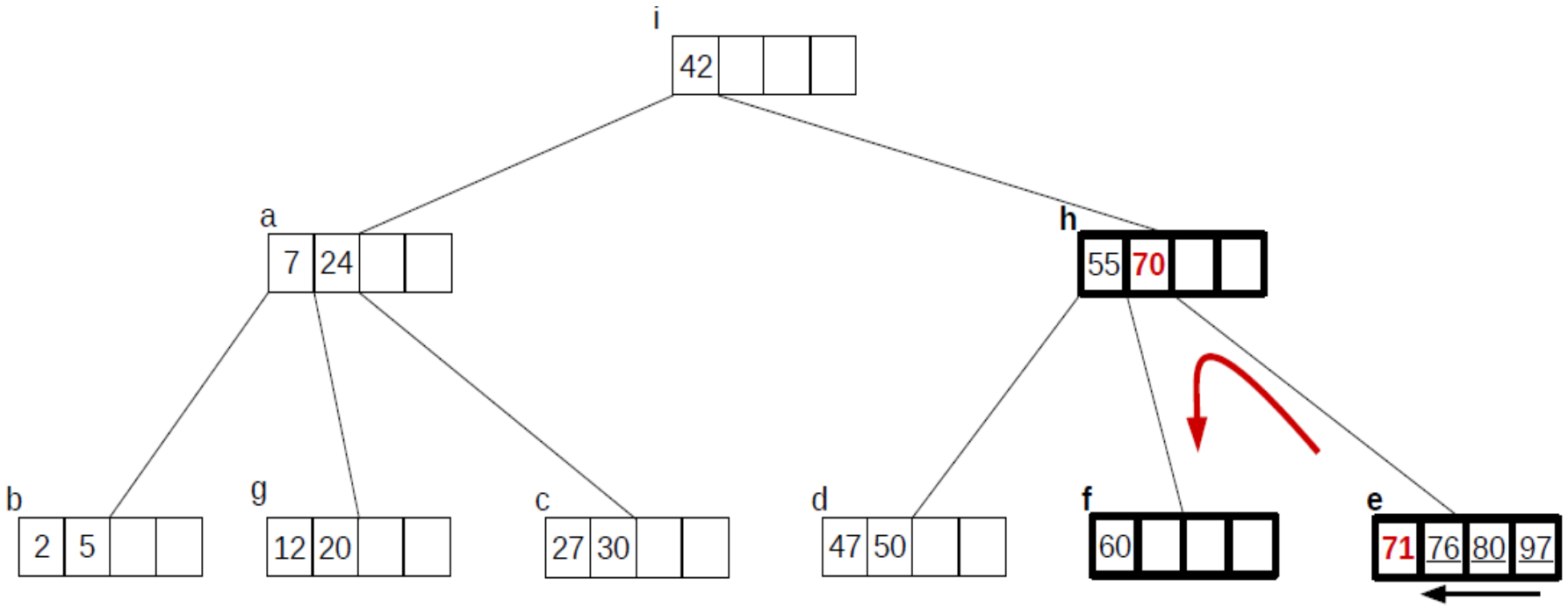
Its left sibling (d) is filled to its minimum capacity (50%).

Its right sibling (e) is filled to more than 50%.



B-trees : Deletion

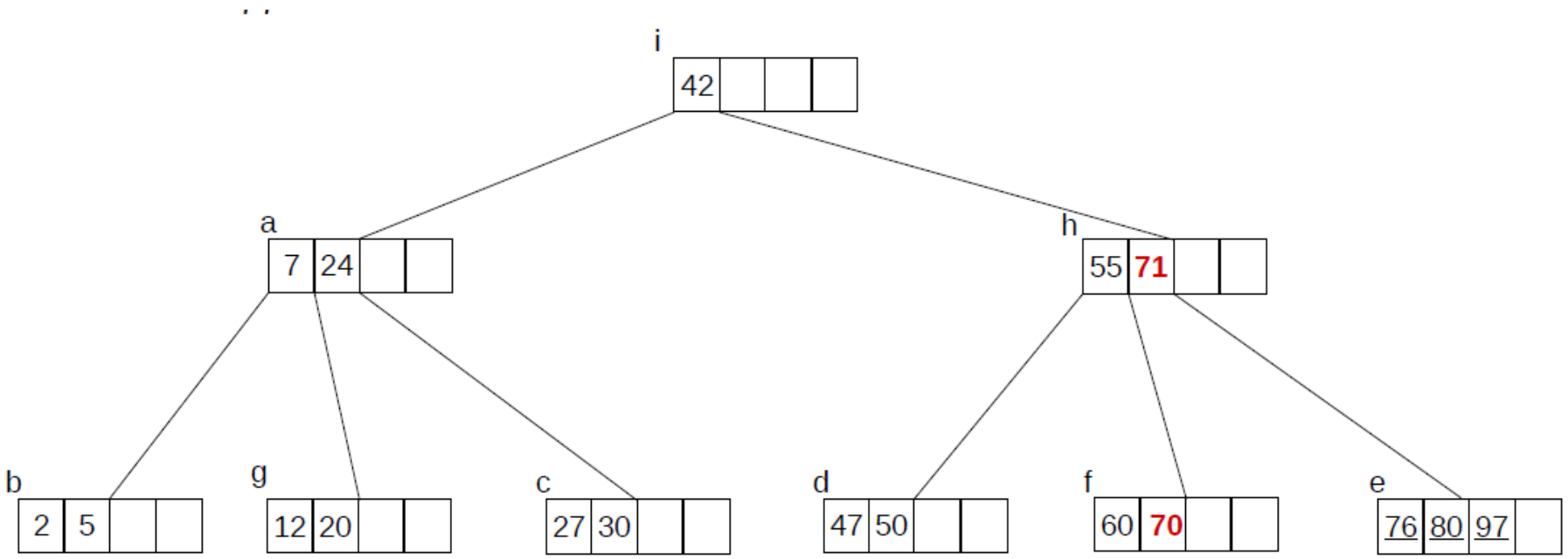
→ Right-to-left redistribution between f and e (via h).



B-trees : Deletion

Example 1: Deletion of 57

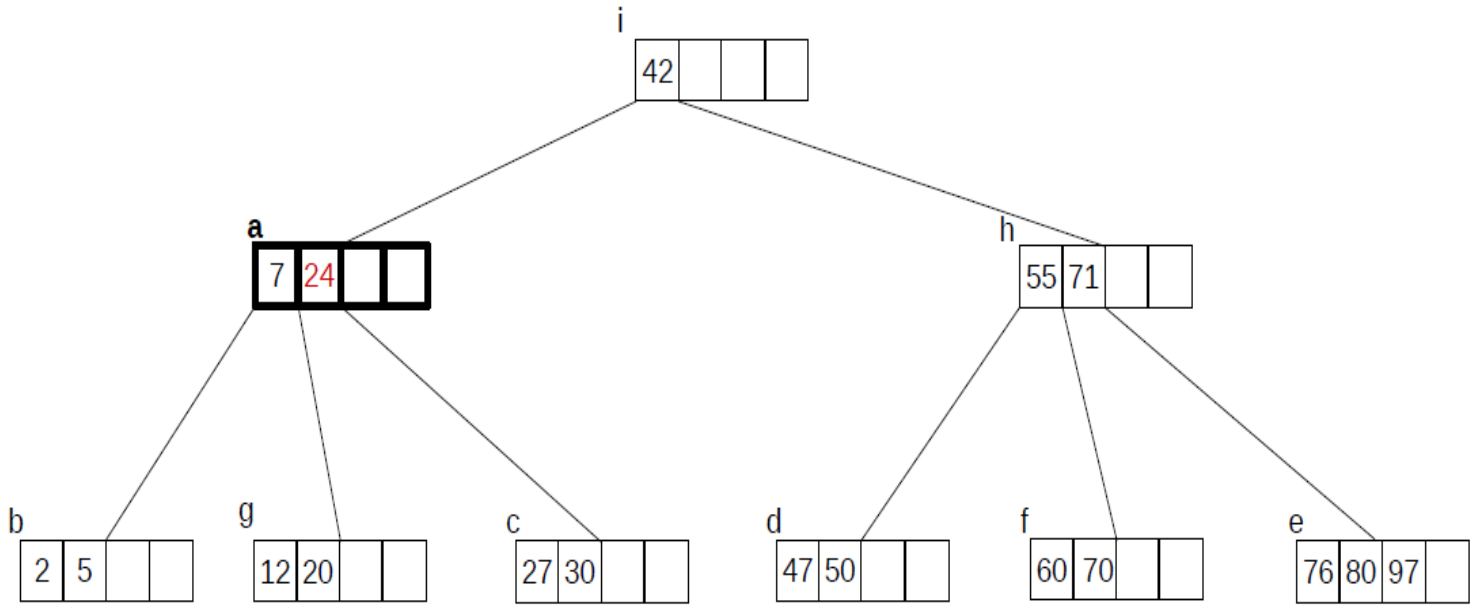
→ End deletion of 57



B-trees : Deletion

Example 3: Deletion of 24

1. The search for 24 ends at (a, 2).

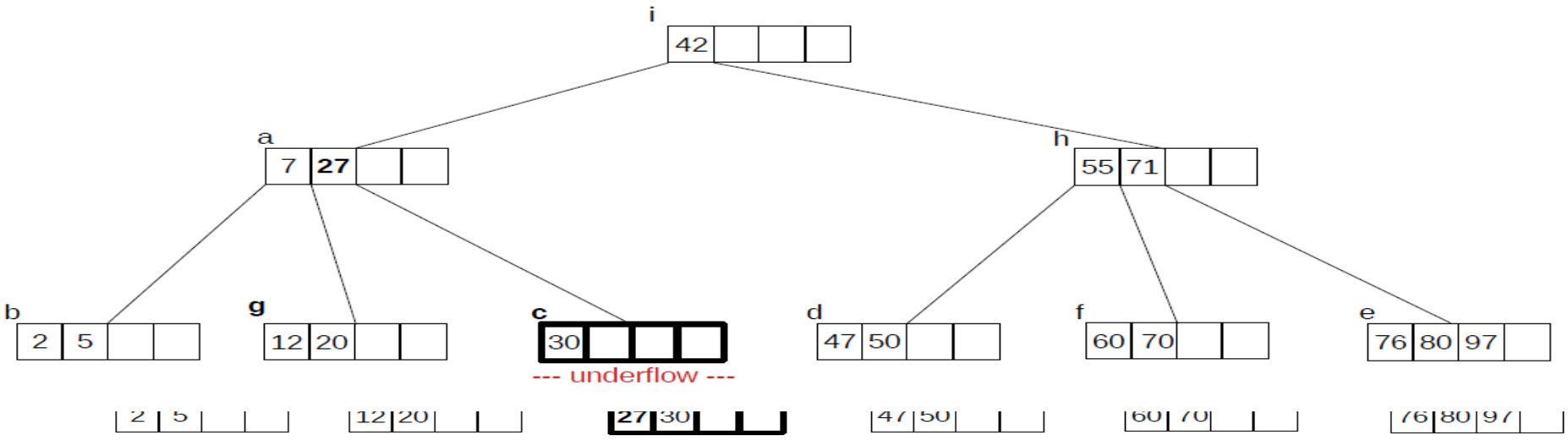


B-trees : Deletion

Example 3: Deletion of 24

2. a is an internal node, so 24 is replaced by the next in-order successor: 27 (c, 1).

3. 27 is deleted using internal shifts in c.



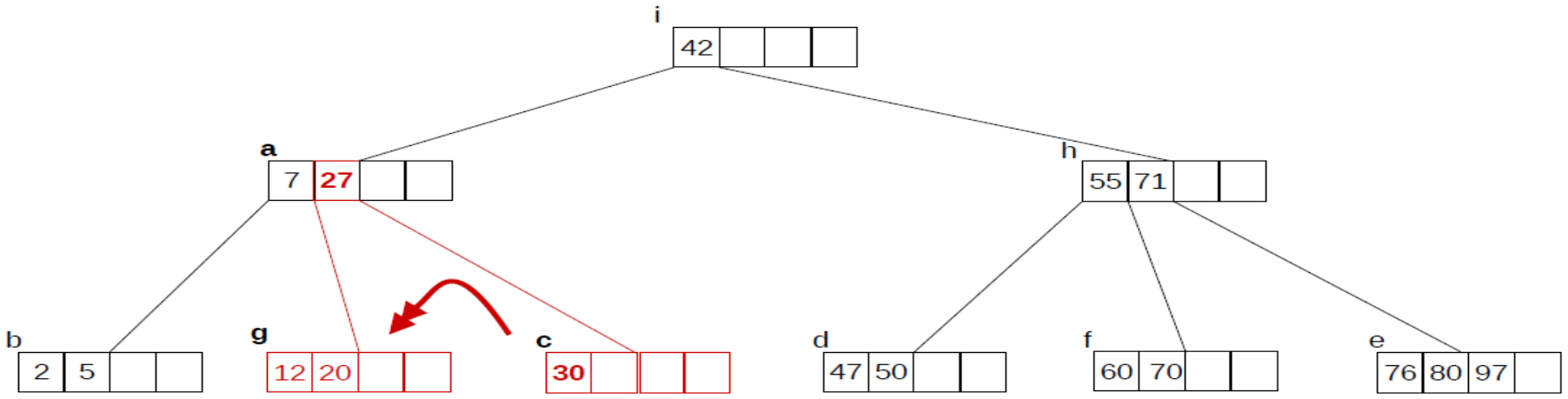
B-trees : Deletion

Example 3: Deletion of 24

4. c becomes underfilled (underflow):

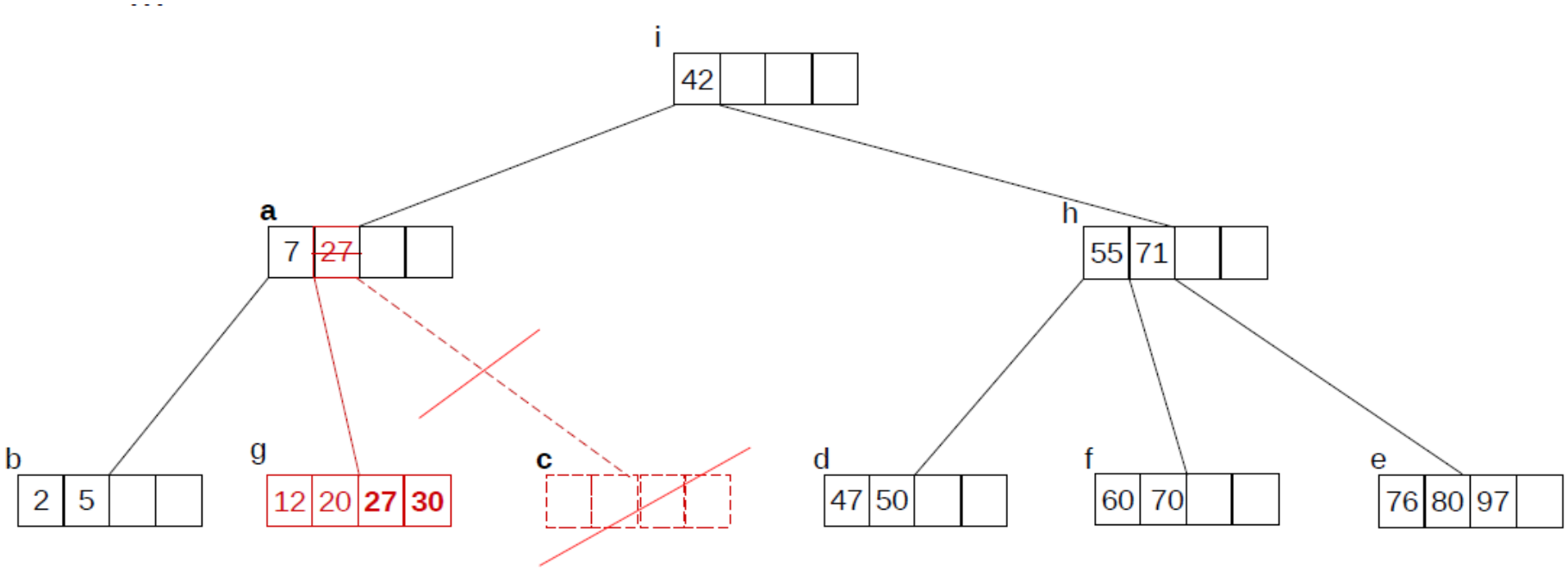
Its only sibling (g) is filled to its minimum capacity.

→ Merge c and g into g (with the value 27 being moved down from the parent node a).



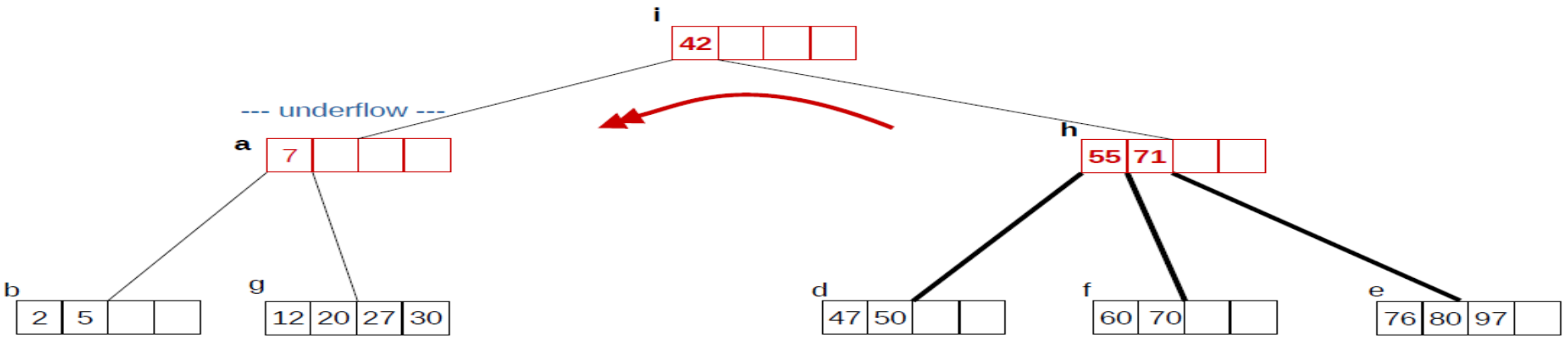
B-trees : Deletion

→ Delete 27 in a and release the node c (as it is now empty).



B-trees : Deletion

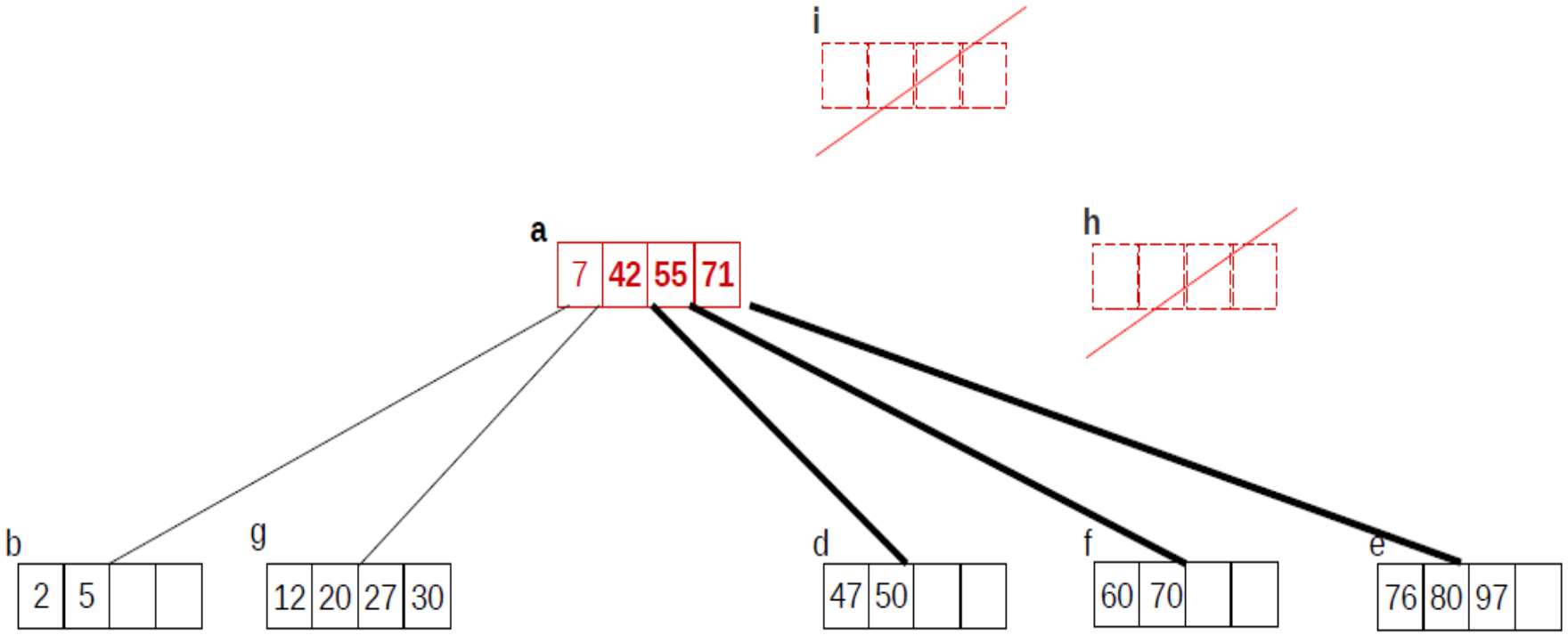
a becomes underfilled. Its only sibling (h) is at 50%.
→ Merge a and h into a (with the value 42 being moved down from the parent node i).



B-trees : Deletion

Exemple 3 : Suppression de 2

Release of node i and h:



B-trees : variations

There are several variations of the basic method

1) B+-tree

2) Prefixed B+-tree

3) B*-tree

See the review article:

Goetz Graefe, Modern B-Tree Techniques

Foundations and Trends in Databases, Vol. 3, No. 4 (2010) 203–402

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.219.7269&rep=rep1&type=pdf