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

m-ary trees

Presented by : Dr. Daoudi Meroua

Academic year: 2024/2025

m-ary trees

Definitions

An m-ary search tree of order NNN is a tree where each node can contain at most:

- N−1 ordered values (val1,val2,…,valN−1)
- N children (Child1,Child2,…,ChildN)

For a given node:

- The degree represents the current number of children.
- The number of current values is always equal to degree−1
- Minimum degree: 2, and maximum degree: N.

m-ary trees

Properties

a) All values in the subtree Child1 are <val1
b) All values in the subtree Childj are >valj−1 and <valj (for j∈[2,degree−1])
c) All values in the subtree Childdegree are >valdegree−1

m-ary trees

Example of a 5-ary Search Tree

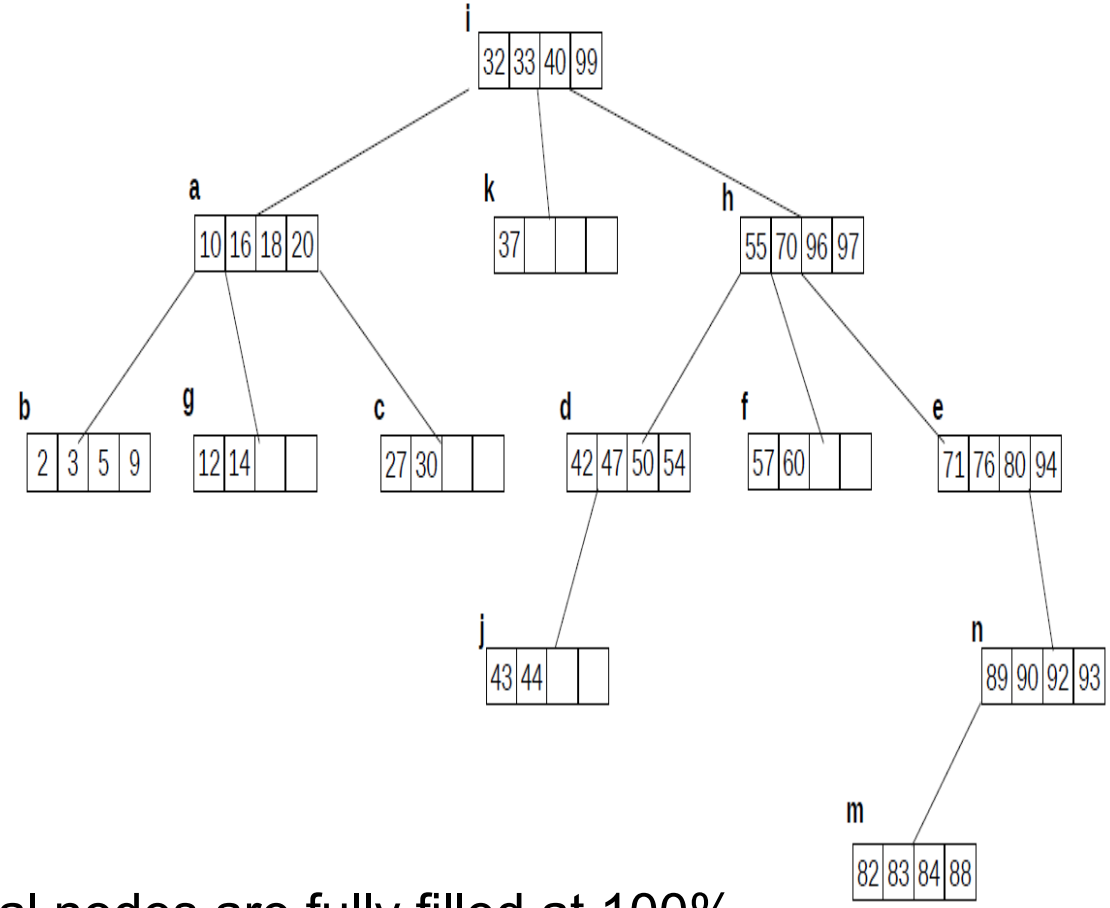
The root node: **i**

The internal nodes: **i, a, h, d, e, n**

The leaf nodes: **b, g, c, k, j, f, m**

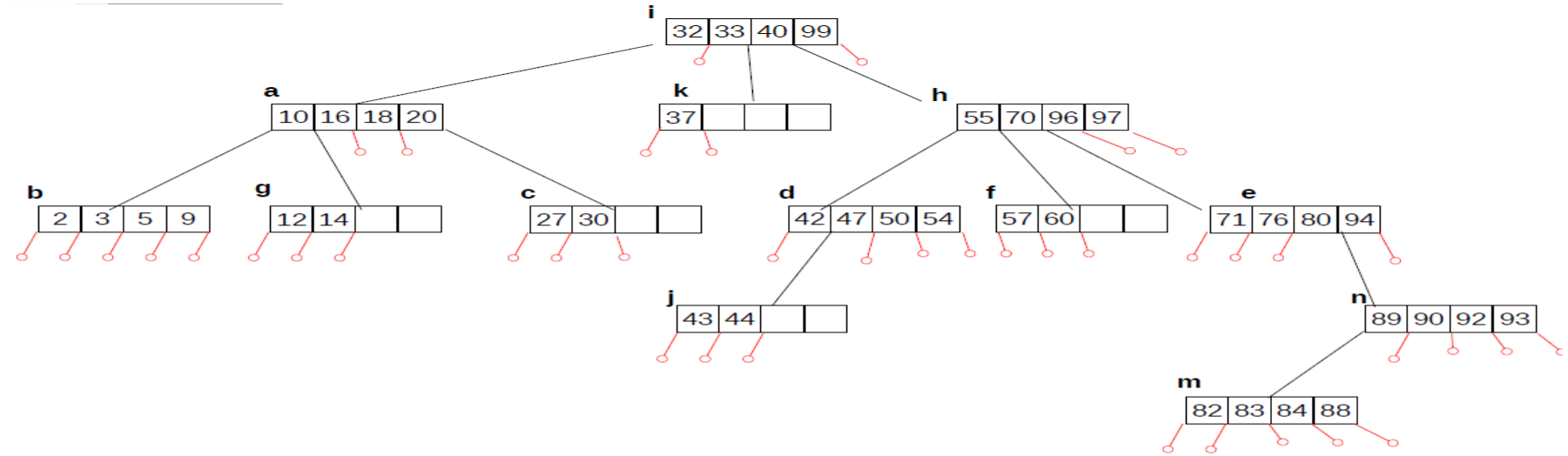
The depth (or height) of the tree = **4** (the level of the farthest leaf)

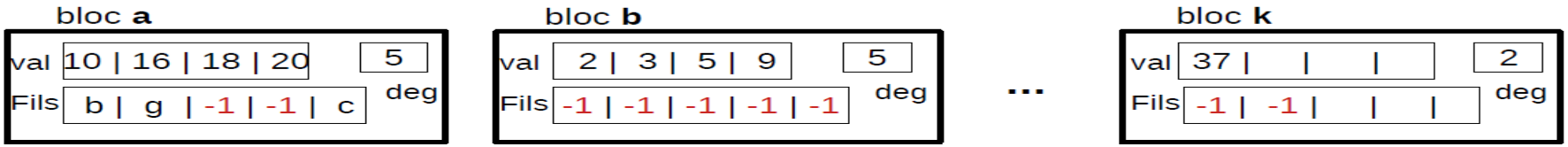
degree(a) = **5**, degree(b) = **5**, degree(c) = **3**, degree(d) = **5**, degree(e) = **5**, degree(f) = **3**, degree(g) = **3**, … etc



Top-Down Property: All internal nodes are fully filled at 100% (optionally verified)

m-ary trees

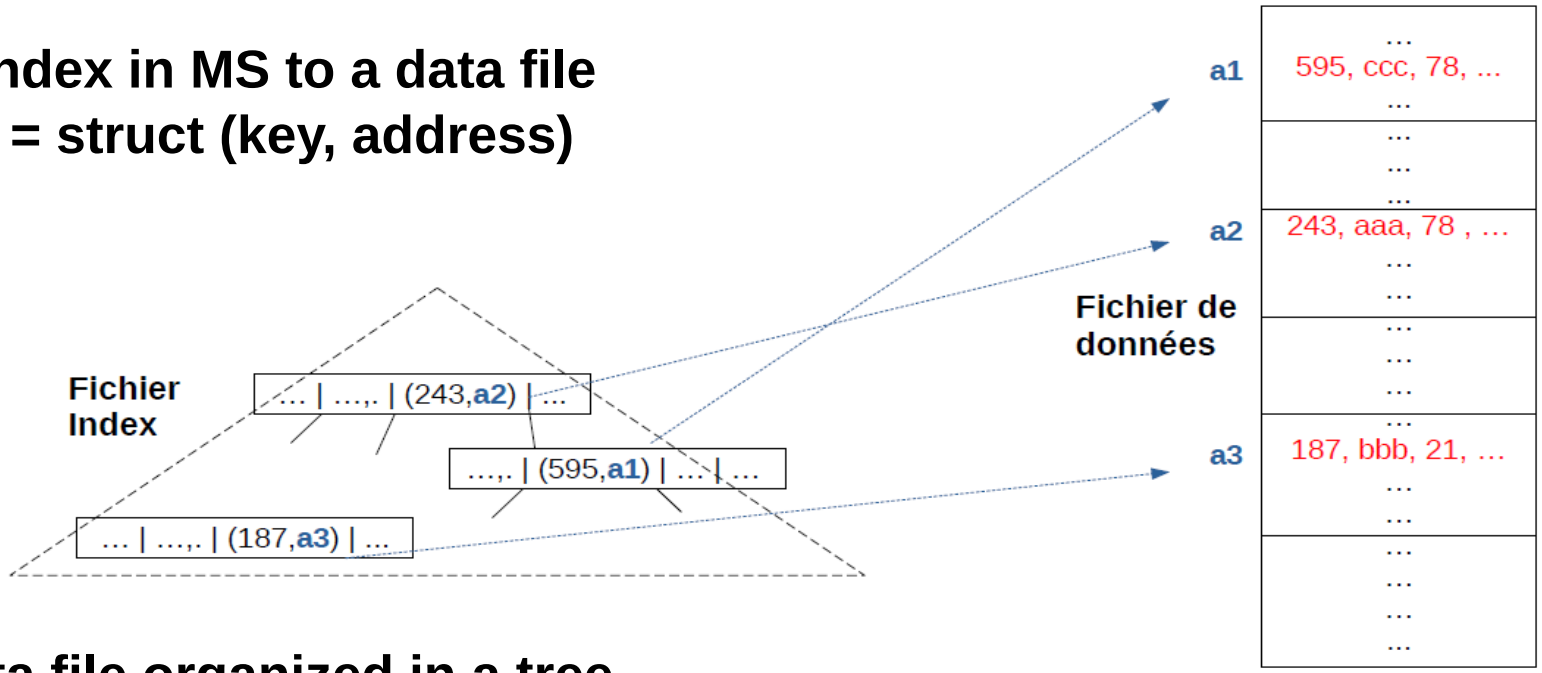




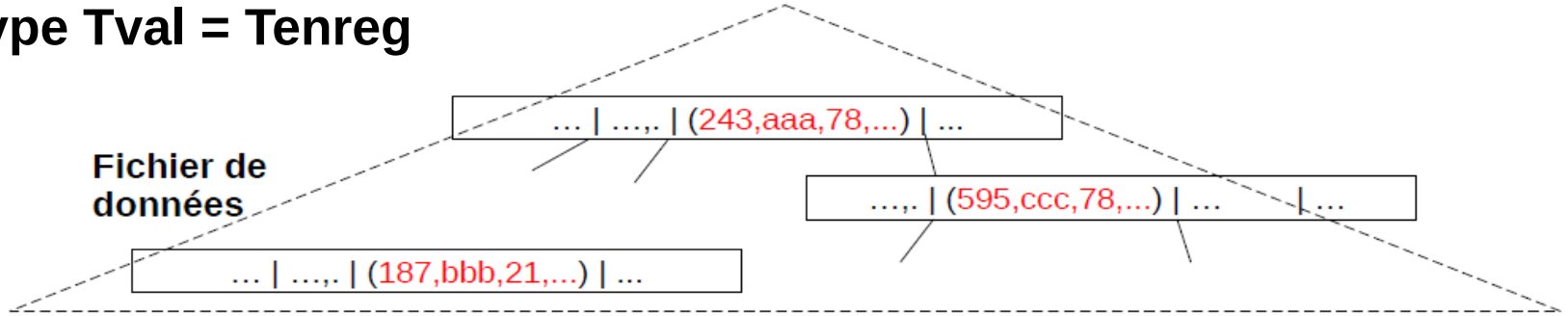
Tbloc = structure
Val : tableau[N-1] de typeqlq; // enregistrements
ou (cles-adr)
Fils : tableau[N] d'entier; // numeros de blocs
degre : entier; // nb d'elt dans le tableau Fils
Fin;

Uses of m-ary trees

1- As an index in MS to a data file
Type Tval = struct (key, address)



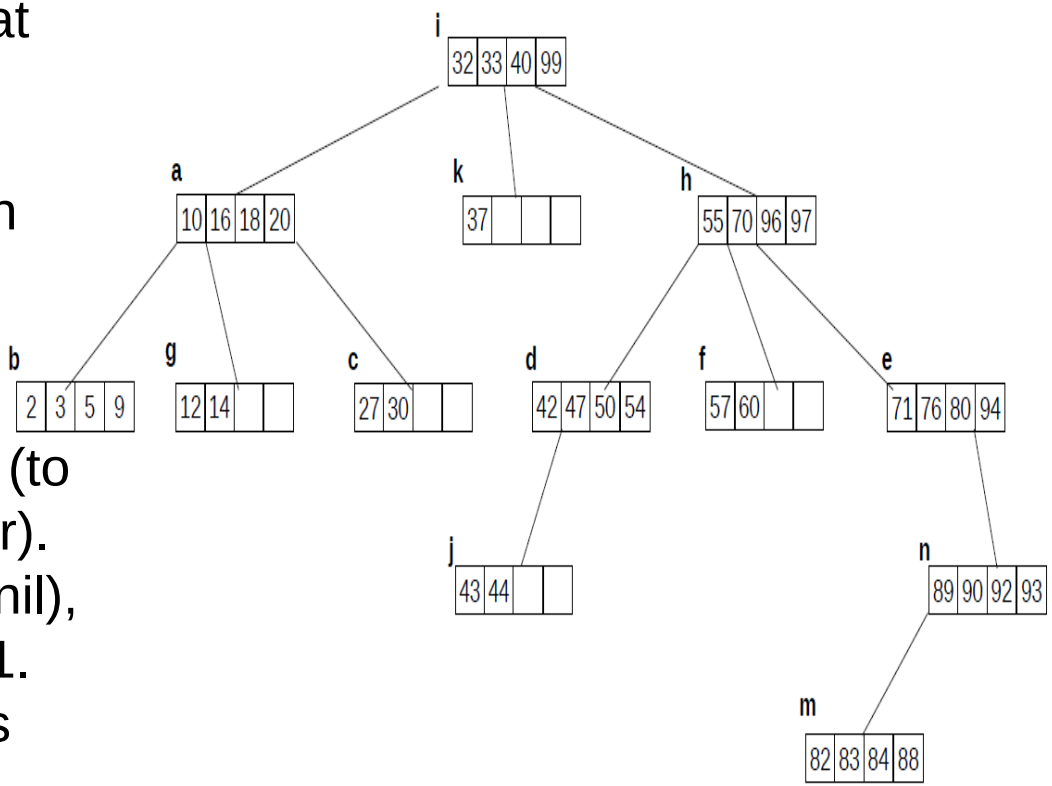
2- As a data file organized in a tree
Type Tval = Tenreg



Search Mechanism

The search for a value C starts at the root node P and continues along a branch:

- If C exists in P, then the search ends successfully.
- If C does not exist in P, then:
  - Let k be the position in P where C should be located (to maintain the values in order).
  - If Filsk is different from -1 (nil), then P ← Filsk; go to step 1.
  - Otherwise, the search ends with failure.



Example:

Rech(47) → traversal of the branch: i, h, d (stops successfully: P = d and k = 2)

Rech(15) → traversal of the branch: i, a, g (stops with failure: P = g and k = 3)

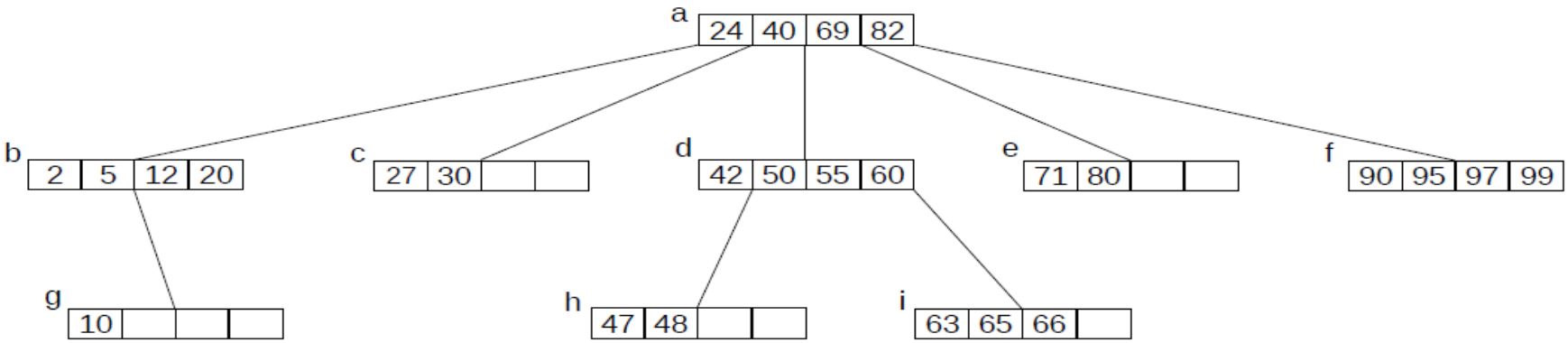
Insertion Mechanism

To insert a new value V into an m-ary search tree, proceed as follows:
1. **Search for V** to verify that it does not already exist and to locate the last visited node P. The search also returns the index k where the value V should be placed to maintain the order of values in P.
2. **IF** P is not full:
   Insert V into P, shifting values as needed to keep the array of values ordered.
ELSE:
   Allocate a new block Q containing a single value V and two child pointers set to -1.
   Connect Q as the k-th child of P (which must currently be -1).
END IF.

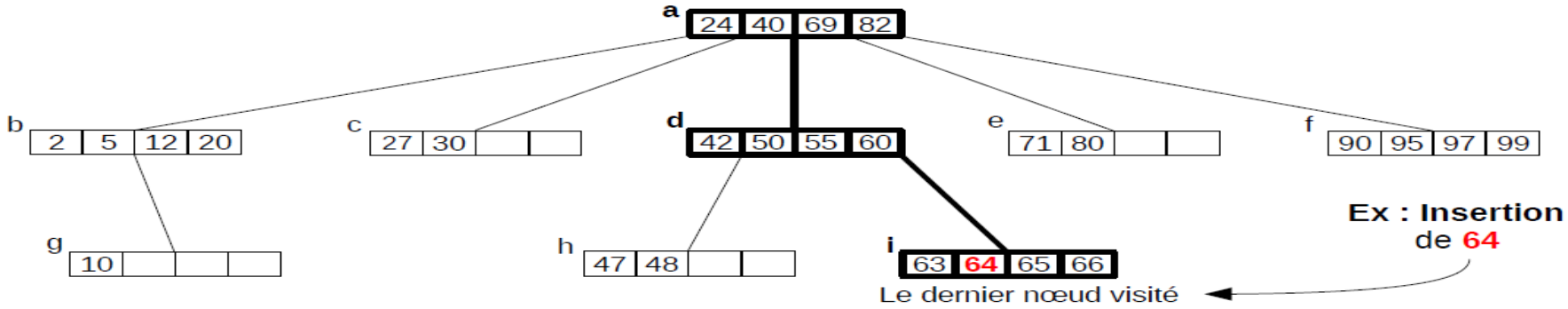
Insertion Mechanism

Example: inserting the value 64

If the last node visited during the search is not full ...



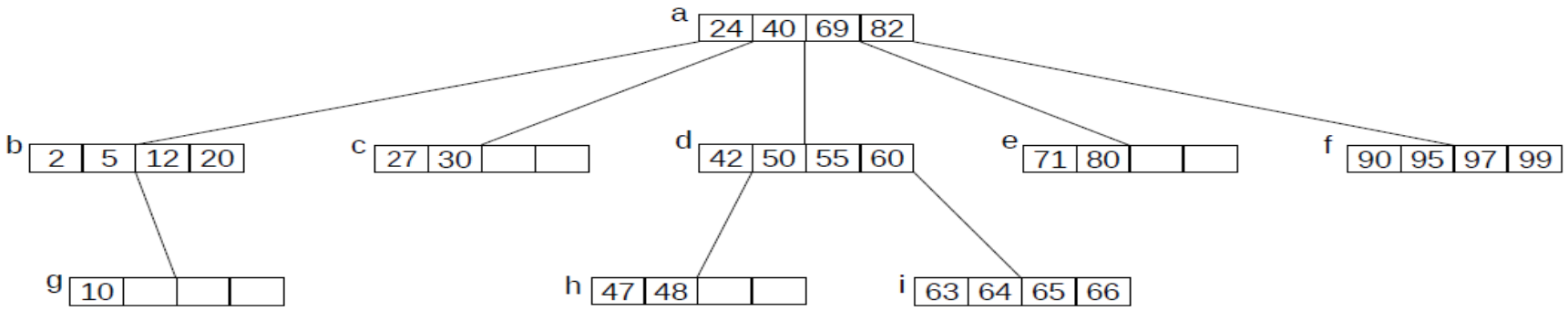
Then insert the new value into the last visited node (with internal shifts within the



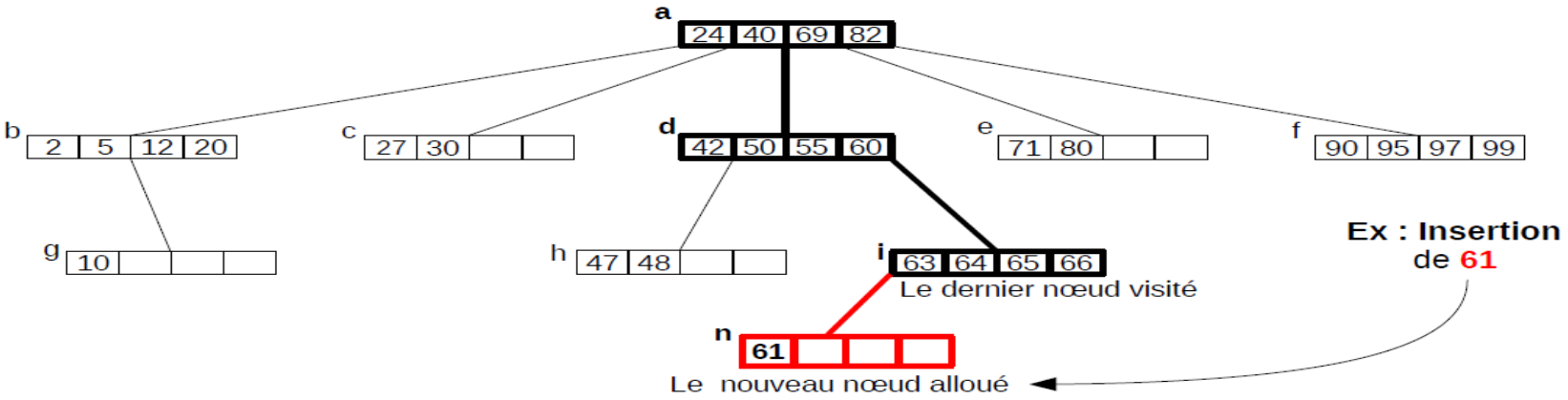
Insertion Mechanism

Example: inserting the value 61

If the last node visited during the search is already 100% full ...



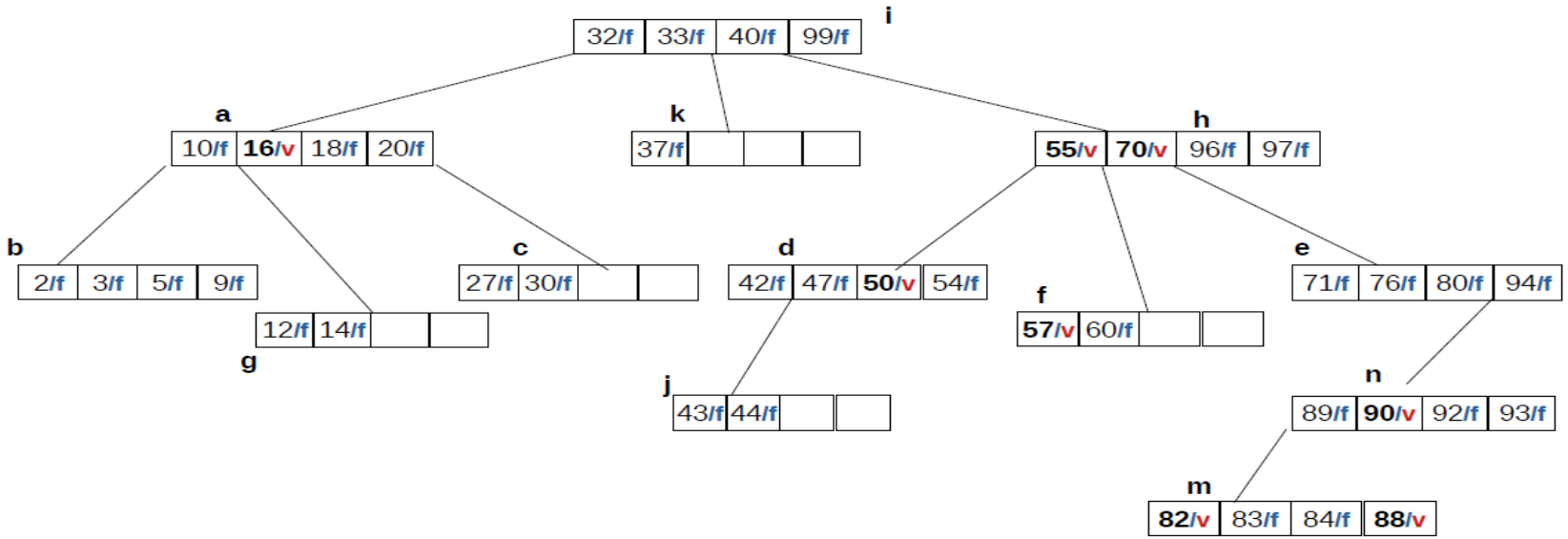
Then allocate a new block containing the new value and connect it to the tree.



Deletion Mechanism

Logical Deletion:

Adding a logical deletion flag for each value.



In this example, the values... 16, 50, 55, 57, 70, 82, 88 et 90 have been logically deleted.

Deletion Mechanism

Physical Deletion:

1. Search for the node containing the value v ⇒ p
2. IF **p** is a leaf
   - Delete v by shifting values; if p becomes empty, free p and update its parent.

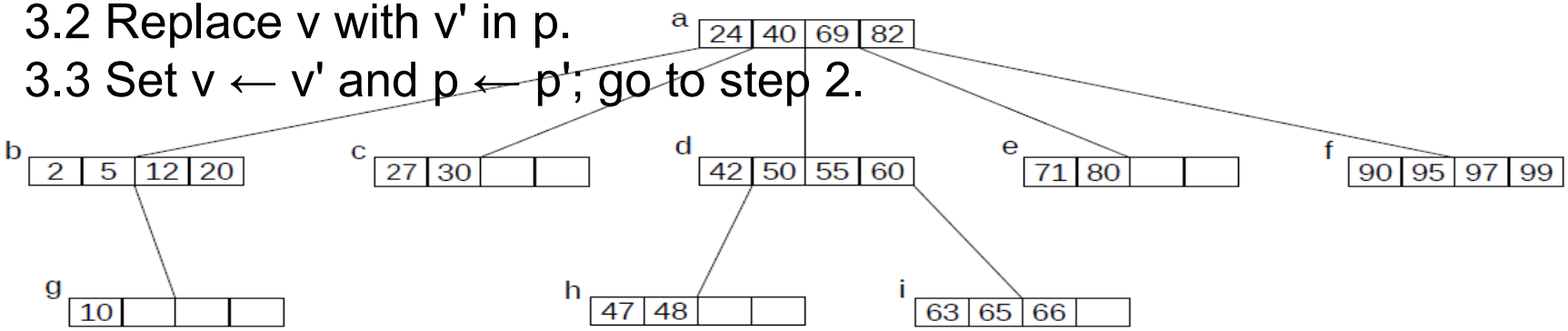
   END IF

   Stop.
3. ELSE // so **p** is an internal node

   3.1 Search for the next or previous in-order value ⇒ value: v' and node: p'.

   3.2 Replace v with v' in p.

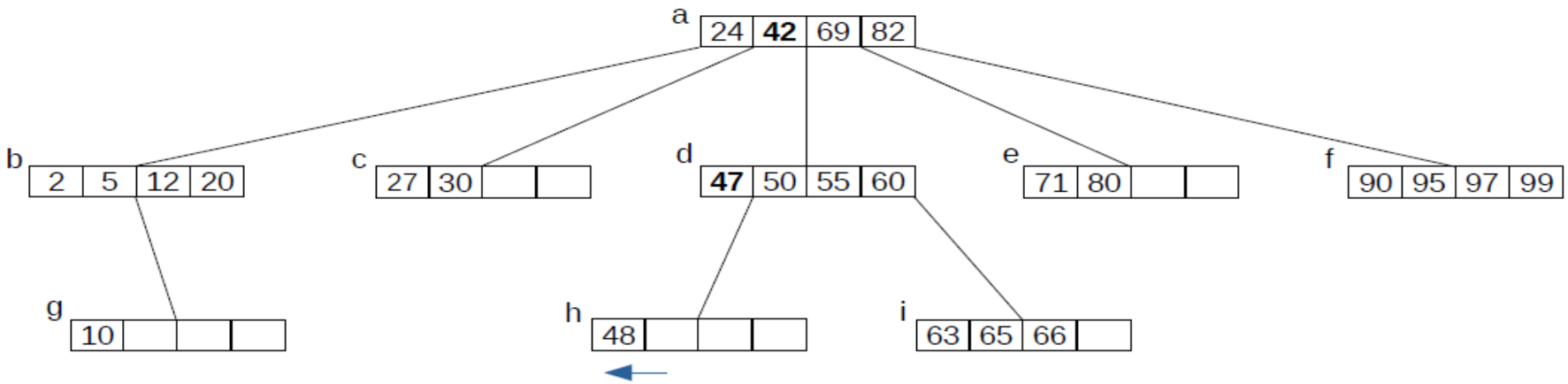
   3.3 Set v ← v' and p ← p'; go to step 2.



Deletion Mechanism

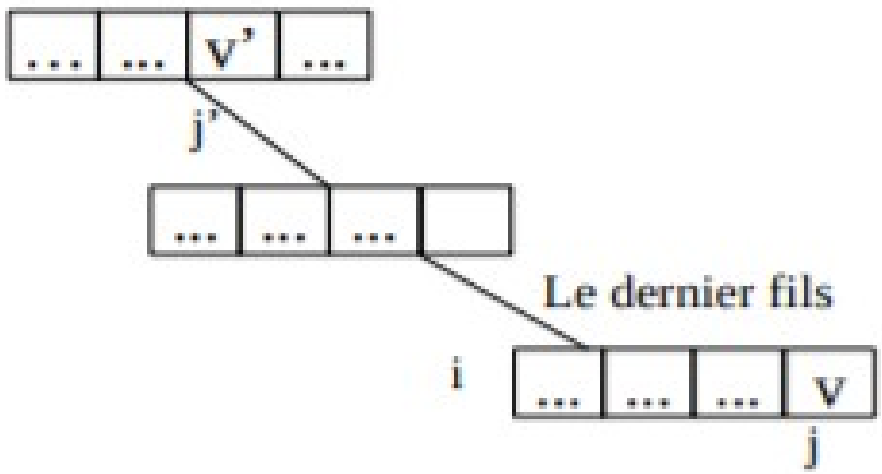
Physical Deletion: Example of deleting the value 40

1. Search for the value 40 (a,2) // a is an internal node
2. In-order successor of 40 is 42 (d,1), and replace 40 with 42 in a // d is also an internal node…
3. In-order successor of 42 is 47 (h,1), and replace 42 with 47 in d // h is a leaf node...
4. Delete 47 by shifting in h.



In-order successor

Case 1:
J=buf.degre-1 and fils.degre=-1



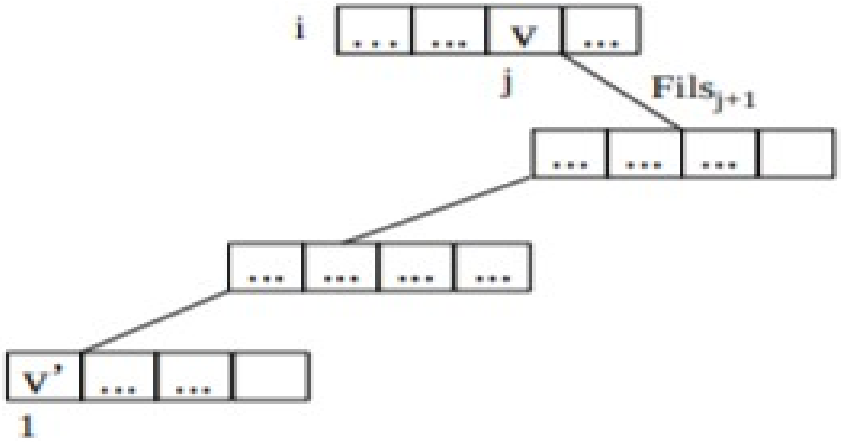
In-order successor

Case 2:

J<buf.degre-1 and $fils_{j+1}$=-1

Cas 3:

$fils_{j+1}$ <> -1



In-order traversal

The **in-order traversal** of an m-ary search tree allows visiting all the values in the tree in ascending order.

```
In-order( r:entier )          // input r: the number of the root block of the tree
var buf : Tbloc ; k:entier
If ( r <> -1 )
   LireDir( F , r , buf )
    for (k = 1 , buf.degre - 1 )
        In-order( buf.Fils[k] )
        visiter( Val[k] )
    endfor

In-order( buf.Fils[ buf.degre ] )
endIf
```