Arbres en Mémoire Secondaire

Les blocs d'un fichier peuvent être organisés sous d'un **arbre de blocs**
C'est utile pour représenter des **arbres de recherche m-aires** en Mémoire Secondaire (**MS**)

Définitions

Un arbre de recherche m-aire **d'ordre N** est un arbre où chaque nœud peut contenir, au maximum : **N-1 valeurs ordonnées** (val_1,val_2,...val_{N-1}) et **N fils** ($Fils_1$,$Fils_2$,… $Fils_N$)

Pour un nœud donné, le ***degré*** représente le nombre de fils courant.

Pour un nœud donné, le nombre de valeurs courantes est toujours = ***degré - 1***

Au minimum : ***degré = 2*** et au maximum : ***degré = N***

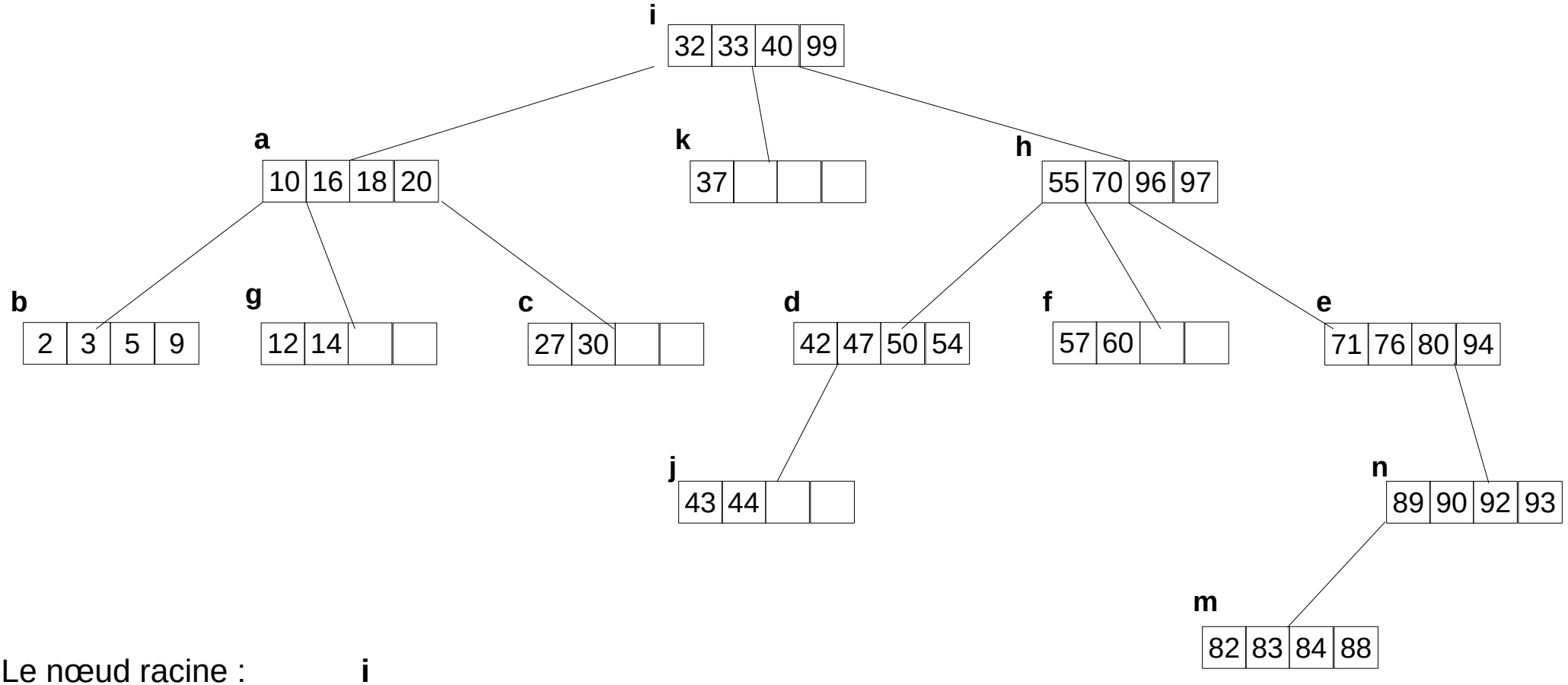
Propriétés

a) Toutes les valeurs dans le sous-arbre $Fils_1$ sont **<** val_1

b) Toutes les valeurs dans le sous-arbre $Fils_j$ sont **>** val_{j-1} et **<** val_j (avec j dans [2 , *degré-1*])

c) Toutes les valeurs dans le sous-arbre $Fils_{degré}$ sont **>** $val_{degré-1}$

Exemple d'arbre de recherche m-aire d'ordre 5



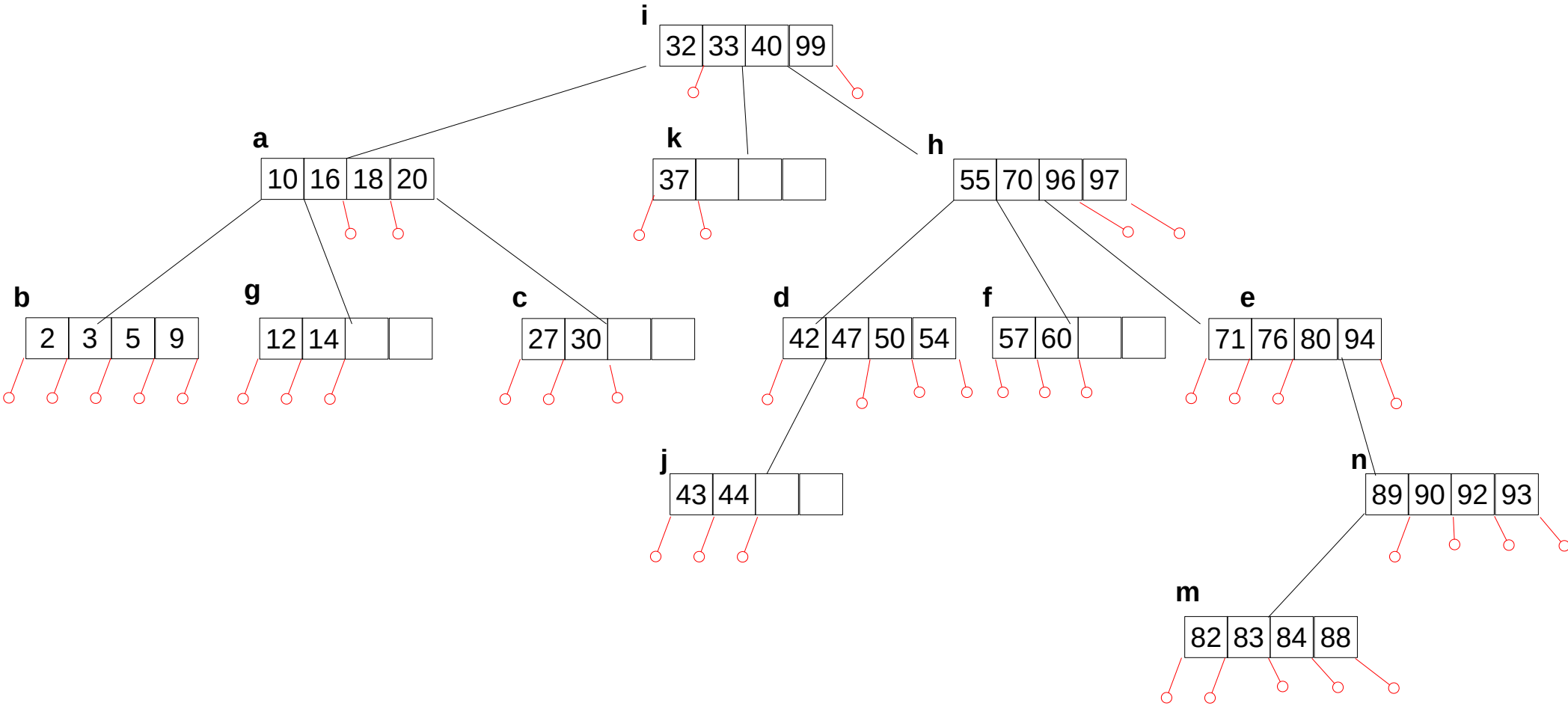
Le nœud racine : **i**
Les nœuds internes : **i , a , h , d , e , n**
Les nœuds feuilles : **b , g , c , k , j , f , m**

La **profondeur** (ou **hauteur**) de l'arbre = **4** (le niveau de la feuille la plus éloignée)

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

Propriété ***Top-Down***: Tous les nœuds internes sont pleins à 100 % (optionnellement vérifiée)

Exemple d'arbre de recherche m-aire d'ordre 5



bloc **a**

val	10 | 16 | 18 | 20	5
Fils	b | g | -1 | -1 | c	deg

bloc **b**

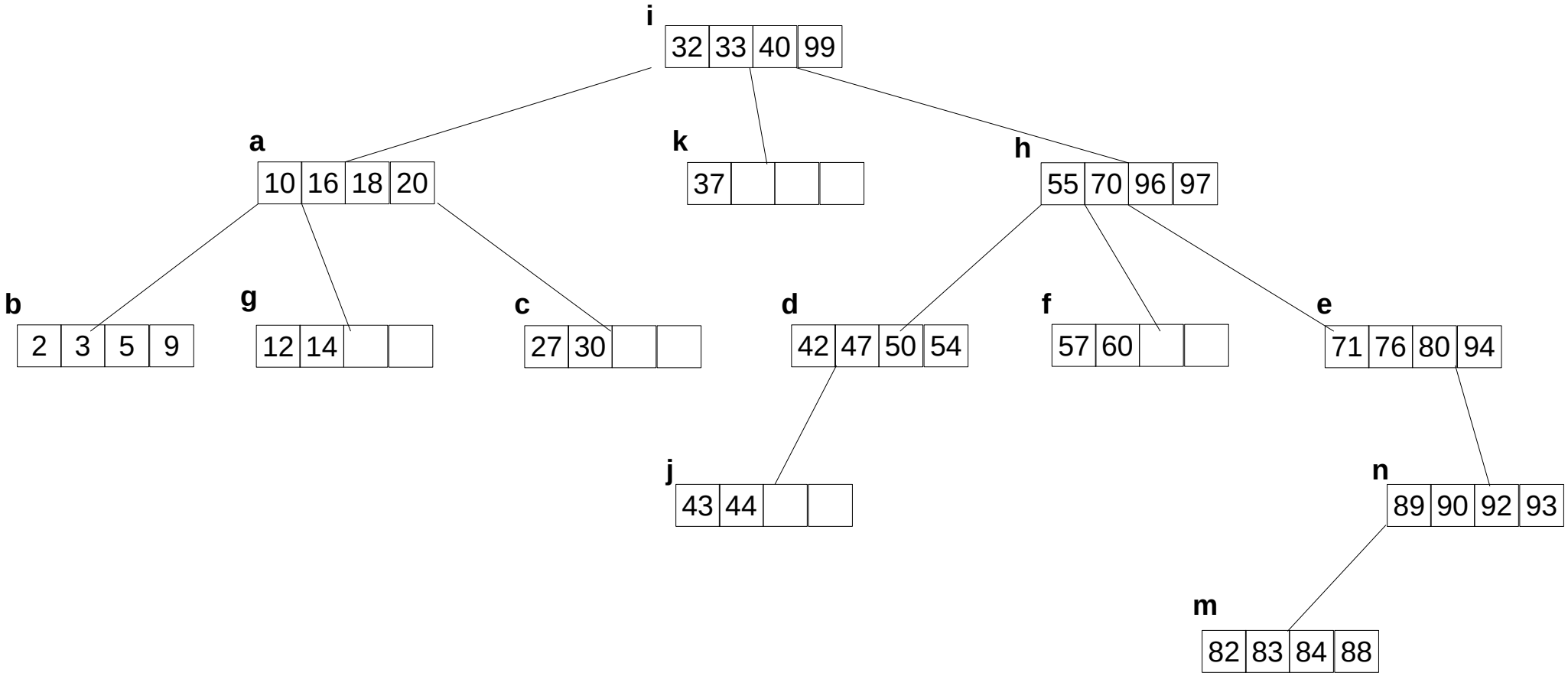
val	2 | 3 | 5 | 9	5
Fils	-1 | -1 | -1 | -1 | -1	deg

...

bloc **k**

val	37 | | |	2
Fils	-1 | -1 | | |	deg

Exemple d'arbre de recherche m-aire d'ordre 5



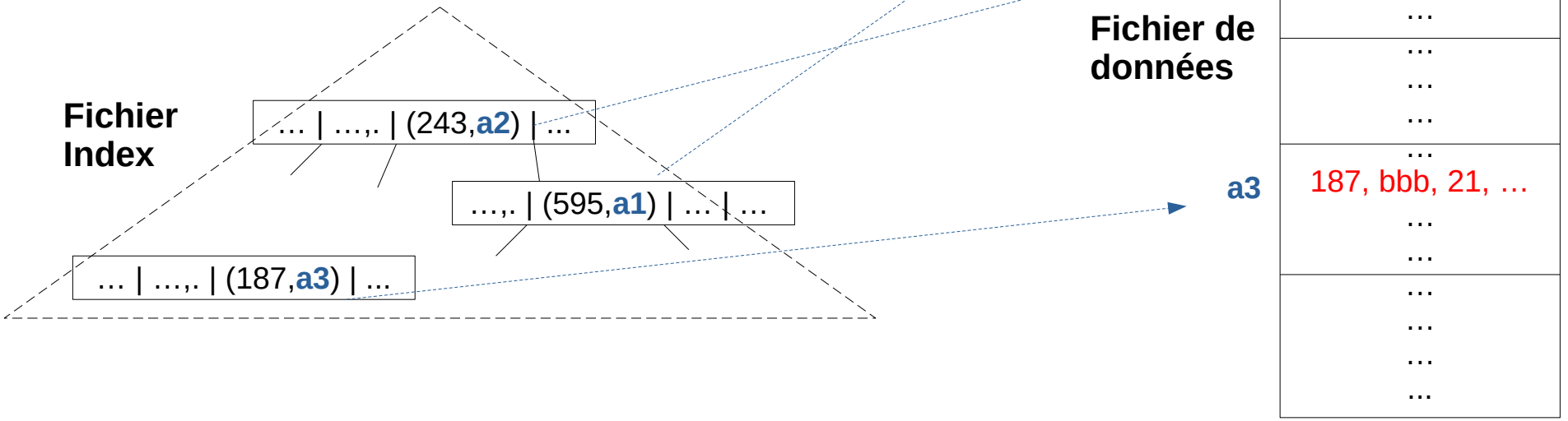
a	b	c	d	e	f	g	h	i	j	k	m	n
10,16,18,20 ...	2, 3, 5, 9 ...	27,30, , ...	42,47,50,54 ...	71,76,80,94 ...	57,60, , ...	12,14, , ...	55,70,96,97 ...	32,33,40,99 ...	43,44, , ...	37, , , ...	82,83,84,88 ...	89,90,92,93 ...

Fichier organisé en arbres m-aire de recherche : les blocs sont chaînés selon une structure d'arbre m-aire
→ Le *numéro du bloc Racine* peut être gardé comme *caractéristique* du fichier (dans le *Bloc d'entête*)

Utilisations des arbres de recherche m-aire

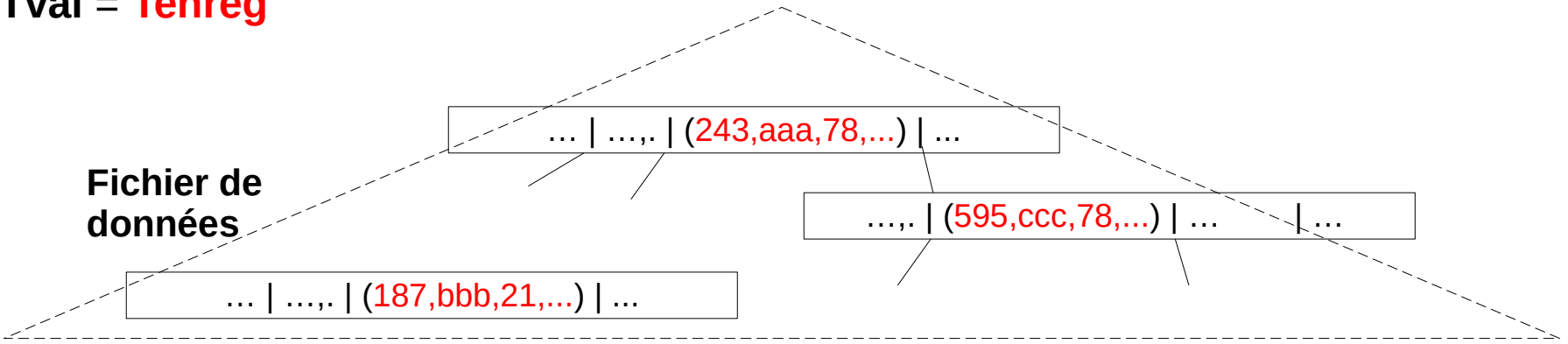
1- **Comme index en MS** vers un fichier de données

Type **Tval** = struct (**clé** , **adr**)

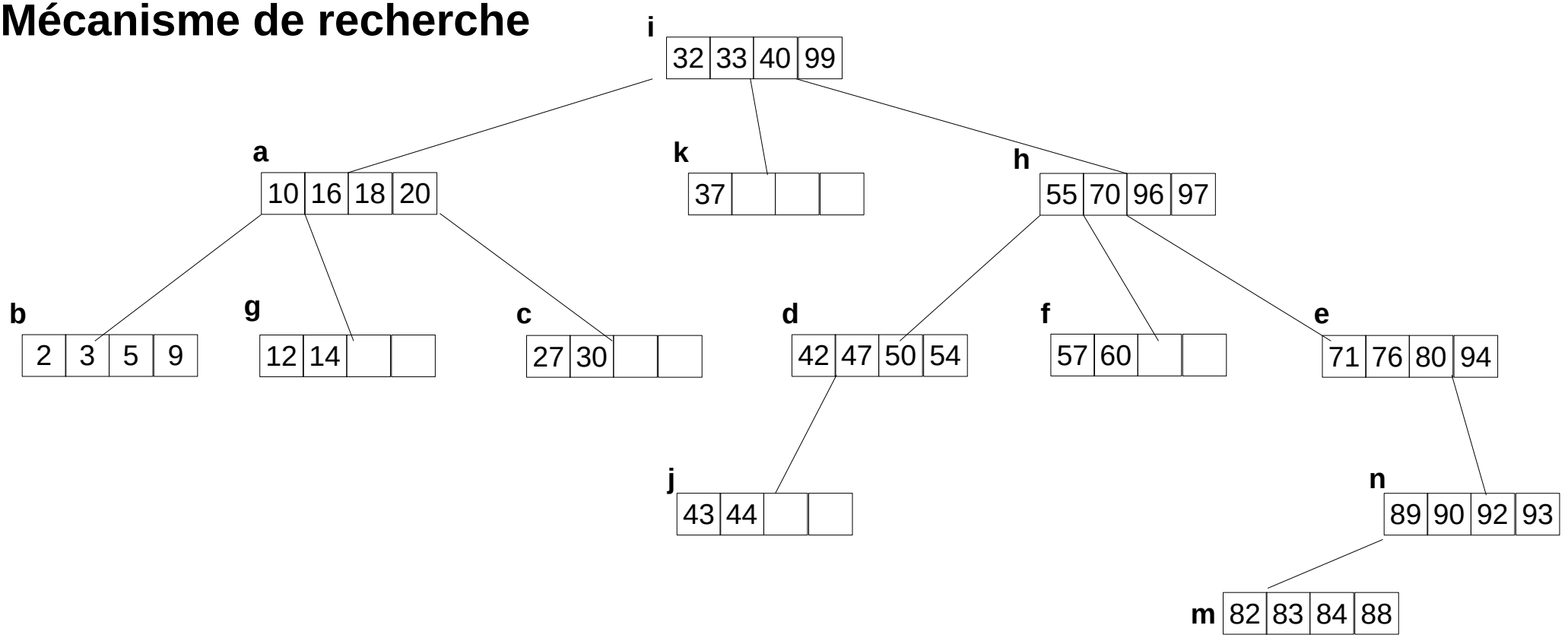


2- **Comme fichier de données** organisé en arbre

Type **Tval** = **Tenreg**



Mécanisme de recherche



La recherche d'une valeur ***C*** commence dans le nœud racine ***P*** et se poursuit le long d'une branche :

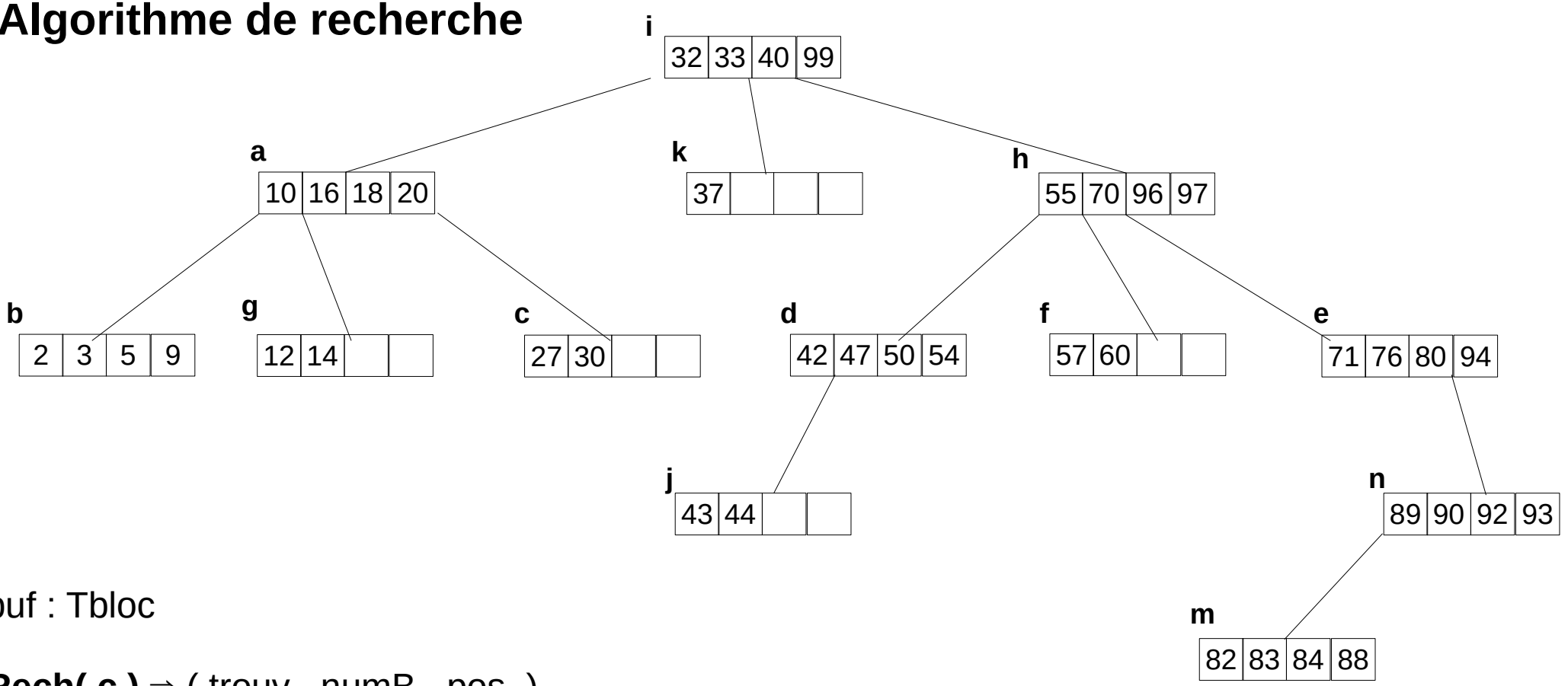
1- Si ***C*** existe dans ***P*** alors la recherche s'arrête avec ***succès***

2- Si ***C*** n'existe pas dans ***P*** alors

- soit ***k*** la position dans ***P*** où devrait se trouver ***C*** (pour que les valeurs restent ordonnées)
- Si $Fils_k$ différent de **-1** (nil) alors $P \leftarrow Fils_k$; aller à 1
- Sinon la recherche s'arrête avec ***échec***.

Ex : Rech(47) → parcours de la branche : **i** , **h** , **d** (arrêt avec succès : ***P*** = ***d*** et ***k*** = ***2***)
Rech(15) → parcours de la branche : **i** , **a** , **g** (arrêt avec échec : ***P*** = ***g*** et ***k*** = ***3***)

Algorithme de recherche



buf : Tbloc

Rech(c) ⇒ (trouv , numB , pos)

Ouvrir(F , … , 'A')

numB ← Entete(F , 1) // le num du bloc Racine

trouv ← faux ; i ← numB

TQ (i <> -1 ET non trouv)

LireDir(F , i , buf) ; numB ← i

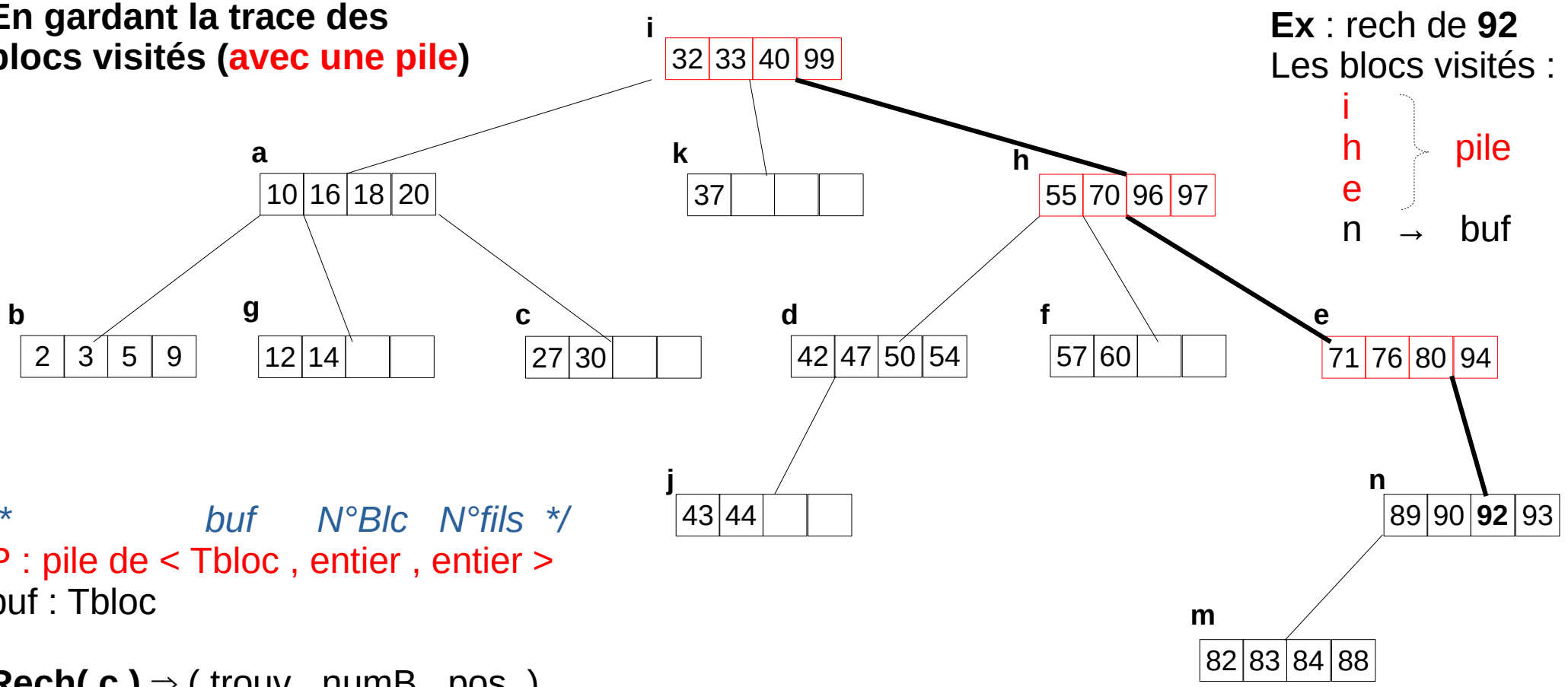
Rechercher c dans buf.val[ ] ⇒ ***Recherche_interne(pos , trouv)***

SI (non trouv) i ← buf.Fils[pos] **FSI**

FTQ

Fermer(F)

En gardant la trace des blocs visités (avec une pile)

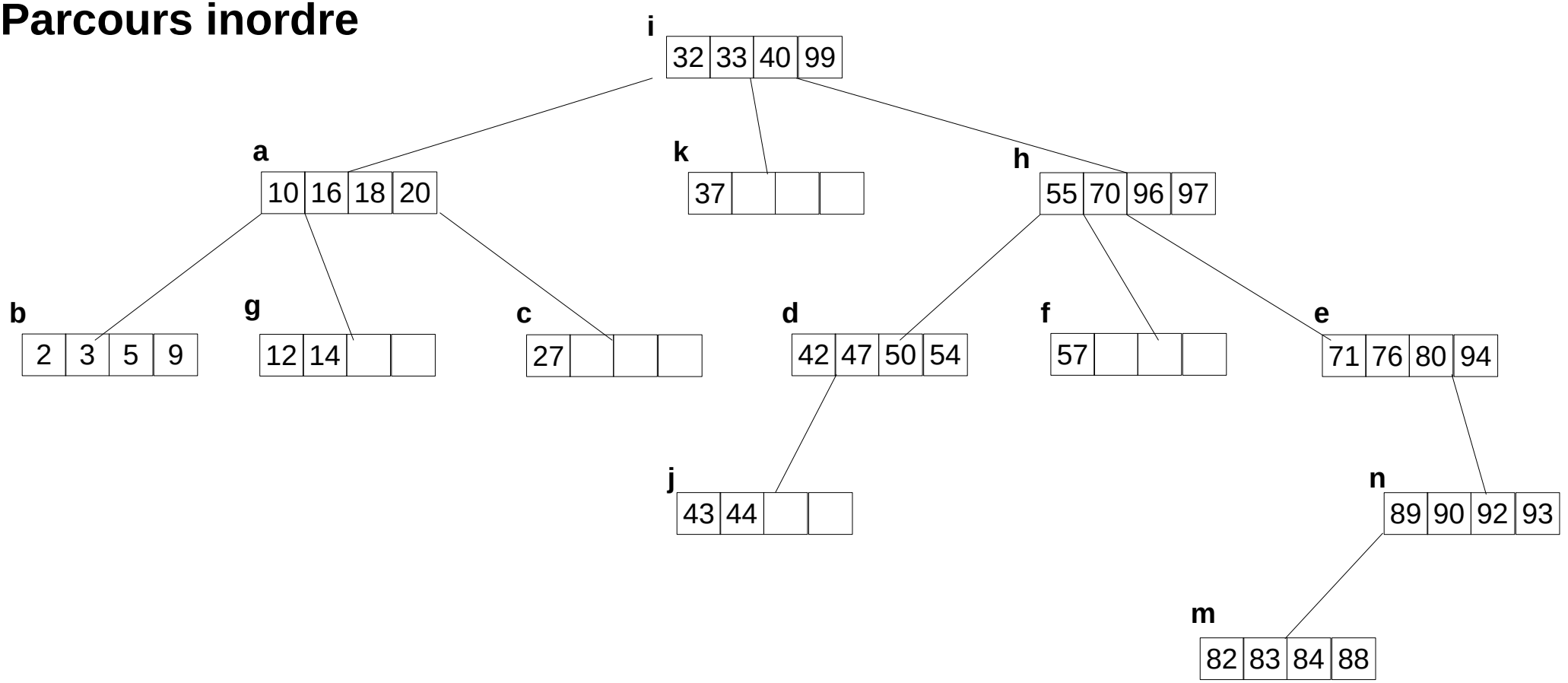


Ex : rech de **92**
Les blocs visités :
i
h
e
pile
n → buf

```
/*                buf     N°Blc   N°fils  */
P : pile de < Tbloc , entier , entier >
buf : Tbloc

Rech( c ) ⇒ ( trouv , numB , pos  )
    Ouvrir( F , … , 'A' )
    CreerPile( P )
    numB ← Entete( F , 1 )     // le num du bloc Racine
    trouv ← faux
    TQ ( numB <> -1 ET non trouv )
        LireDir( F , numB , buf )
        Rechercher c dans buf.val[ ] ⇒  ( pos , trouv )
        SI (non trouv )   Empiler( P , < buf , numB, pos > ) ;   numB ← buf.Fils[ pos ]  FSI
    FTQ
    Fermer( F )
```

Parcours inordre



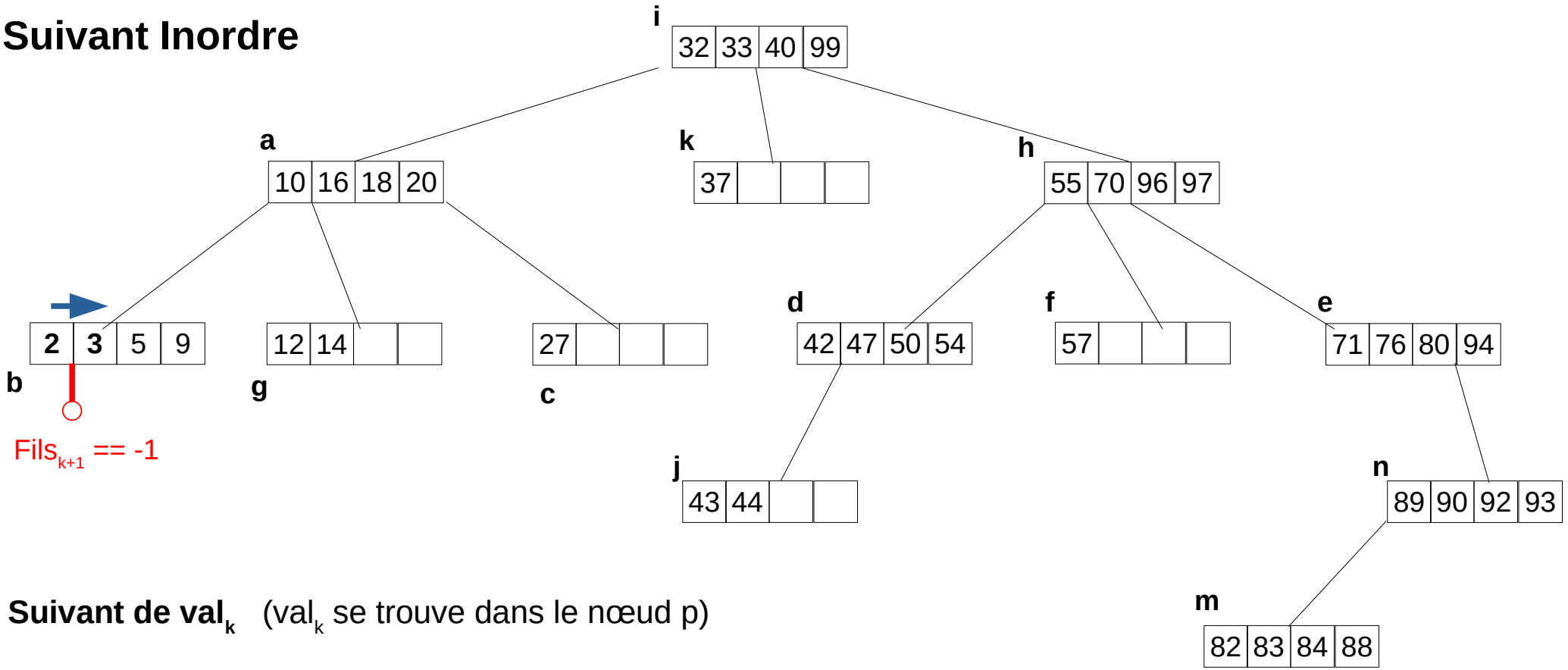

```
Inordre ( r:entier )
var buf:Tbloc   // var locales
    i:entier
SI ( r <> -1 )
        LireDir( F , i , buf )
        POUR (i = 1 , buf.deg – 1 )
            Inordre( buf.fils[ i ] )
            traiter la valeur buf.val[ i ] ...
        FP
        Inordre( buf.fils[ buf.deg ] )
FSI
```

Les valeurs sont visitées (traitées) en ordre croissant :
2, 3, 5, 9, 10, 12, 14, 16, 18, 20, 27, 32, 33, 37, 40, 42, 43, 44, 47, 50, 54, 55, 57, 70, 71, 76, 80, 82, 83, 84, 88, 89, 90, 92, 93, 94, 96, 97, 99

La plus petite valeur de l'arbre (2) se trouve
⇒ à la 1ere position du nœud le plus à gauche.

Comment localiser le suivant inordre d'une valeur ?

Suivant Inordre



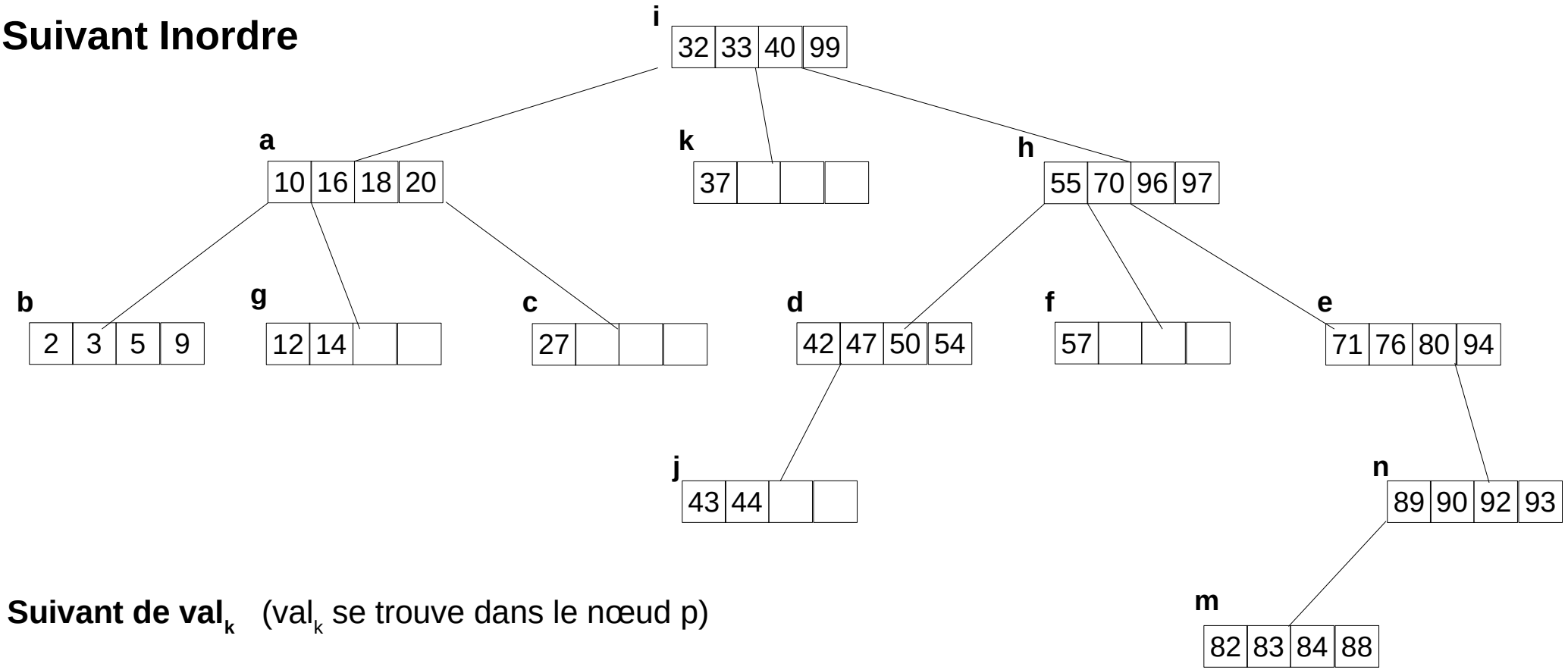
Suivant de val$_k$ (val$_k$ se trouve dans le nœud p)

Ex : *le suivant de **2** ⇒ **3**,* *(car le **fils** à droite de **2** est **-1**)*

*(c-a-d le **fils** à droite de **val$_k$** est **-1**)*

*(c-a-d **Fils$_{k+1}$** est **-1**)*

Suivant Inordre

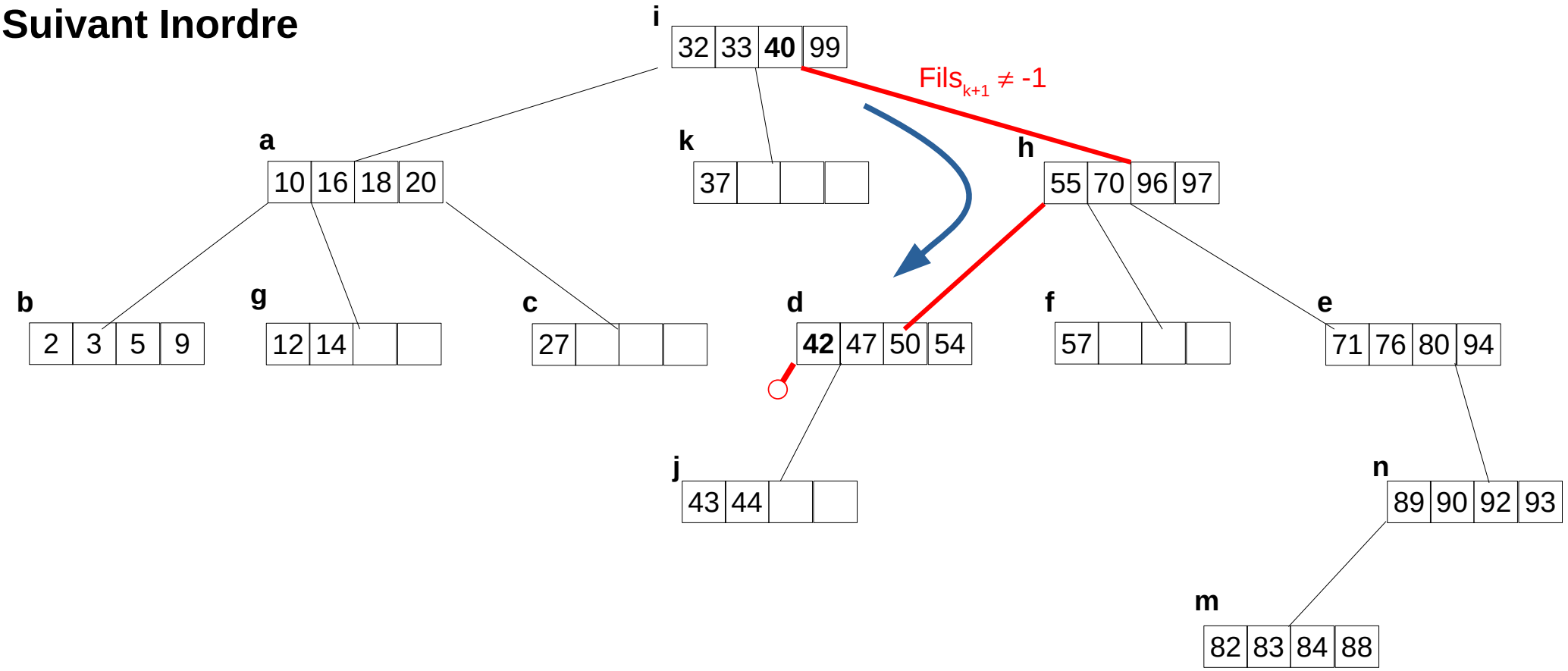


Suivant de val_k (val_k se trouve dans le nœud p)

Si (k < degré-1 && $Fils_{k+1}$ == -1) Le suivant ← val_{k+1} (dans le même nœud p)

Ex : *le suivant de 2 ⇒ 3, le suivant de 3 ⇒ 5, le suivant de 5 ⇒ 9, le suivant de 12 ⇒ 14, le suivant de 16 ⇒ 18, le suivant de 18 ⇒ 20, le suivant de 32 ⇒ 33, …. etc*

Suivant Inordre

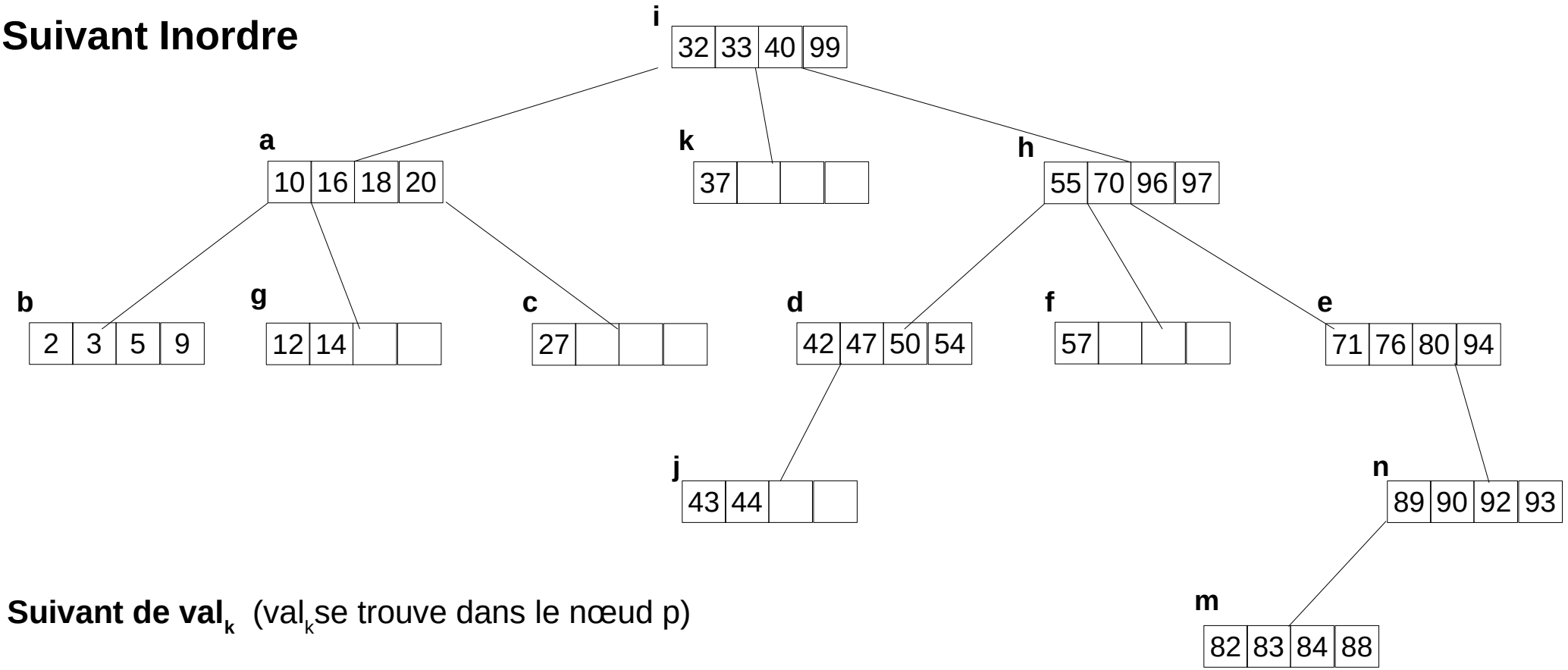


Ex :

*le suivant de 40 ⇒ 42 (car le **fils** à droite de 40 est **différent de -1**)*

*(c-a-d **Fils$_{k+1}$** est différent de **-1**)*

Suivant Inordre



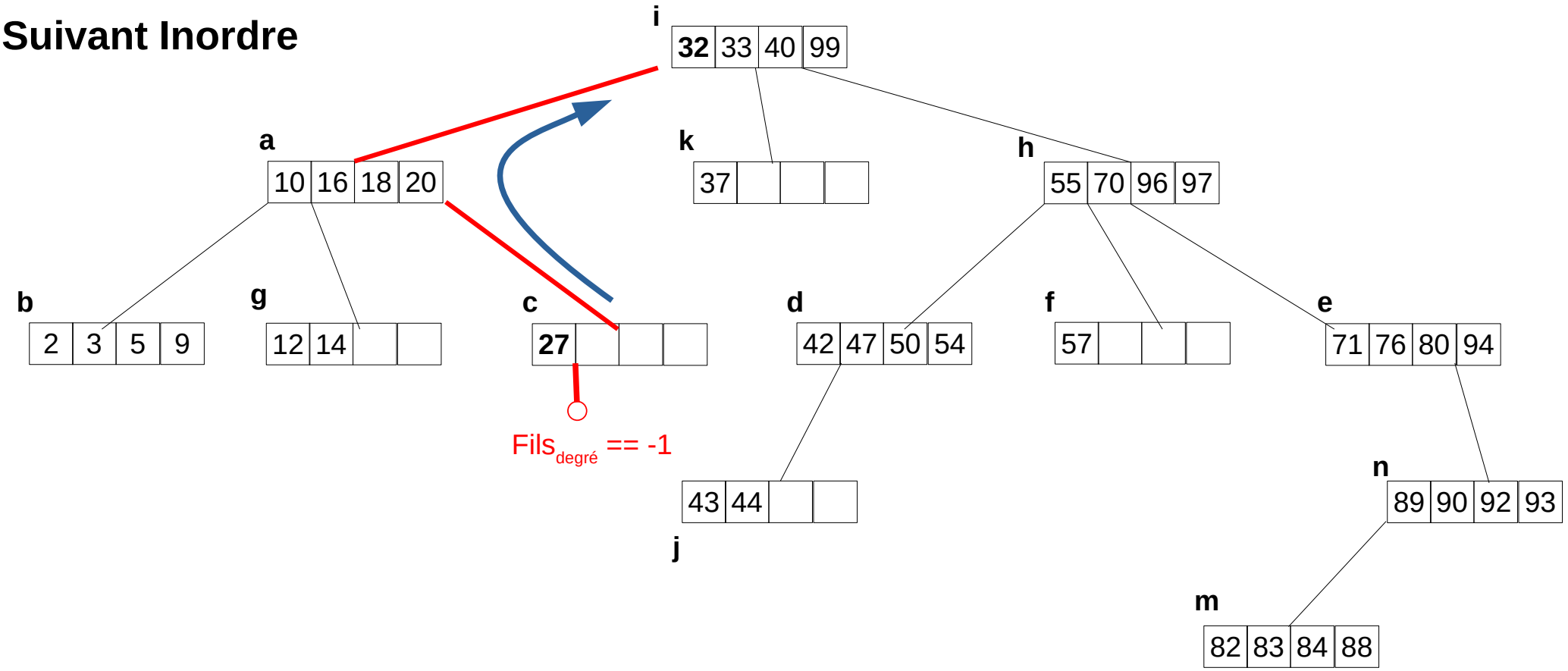
Suivant de val_k (val_k se trouve dans le nœud p)

Si (k < degré-1 && $Fils_{k+1}$ == -1) Le suivant ← val_{k+1} (dans le même nœud p)

Si ($Fils_{k+1} \neq -1$) Le suivant est la <u>plus petite valeur du sous-arbre</u> de racine $Fils_{k+1}$

Ex : *le suivant de 10 ⇒ 12, le suivant de 20 ⇒ 27, le suivant de 33 ⇒ 37, le suivant de 40 ⇒ 42, le suivant de 42 ⇒ 43, le suivant de 55 ⇒ 57, le suivant de 70 ⇒ 71 et le suivant de 80 ⇒ 82*

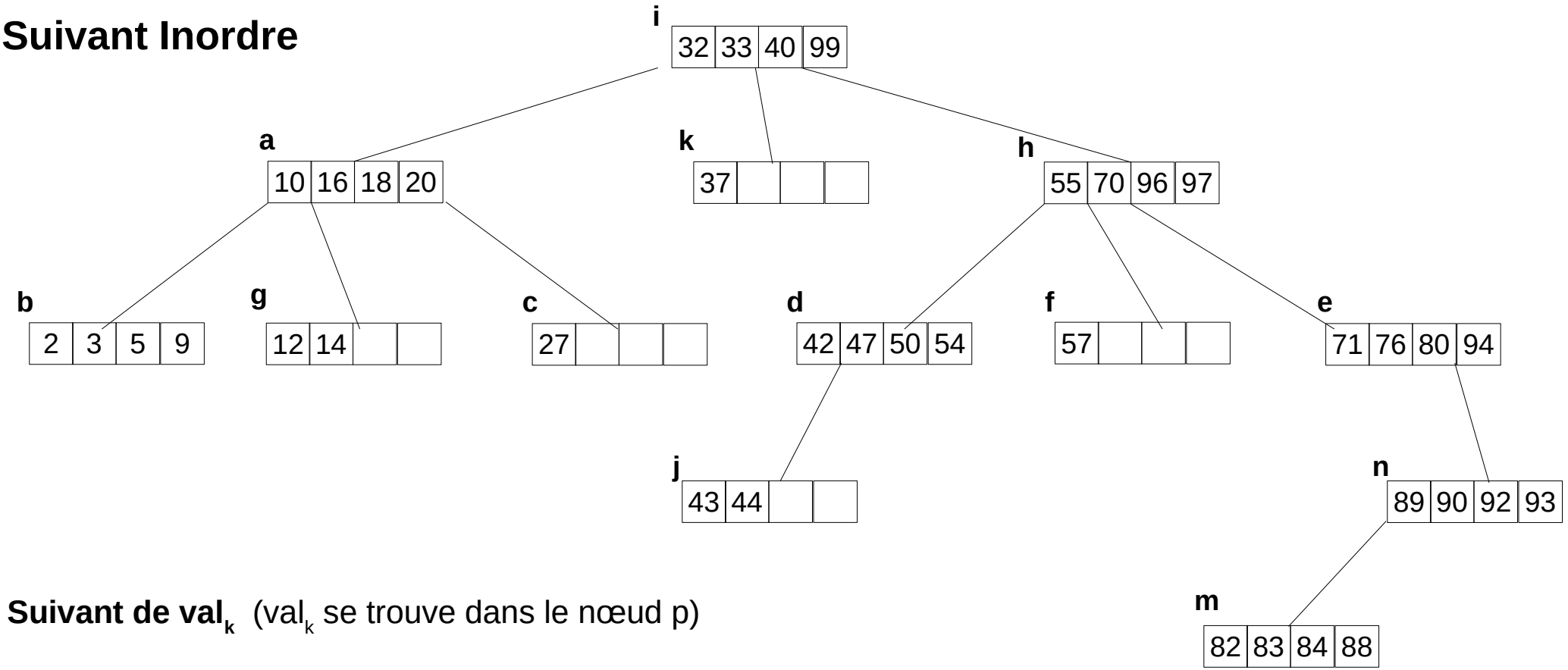
Suivant Inordre



*Ex : le suivant de 27 ⇒ 32 (car 27 est la **dernière valeur du nœud** et son **fils** droit est **-1**)*

*(c-a-d **k == degré-1** et **Fils degré == -1**)*

Suivant Inordre



Suivant de val_k (val_k se trouve dans le nœud p)

Si (k < degré-1 && $Fils_{k+1}$ == -1) Le suivant ← val_{k+1} (dans le même nœud p)

Si ($Fils_{k+1}$ ≠ -1) Le suivant est la plus petite valeur du sous-arbre de racine $Fils_{k+1}$

Si (k == degré-1 && $Fils_{degré}$ == -1) **Le suivant est dans le 1er ascendant duquel on est descendu par un fils autre que le dernier (soit $Fils_j$ avec j < degré). Le suivant est alors la valeur val_j dans cet ascendant**

Ex : le suivant de 9 ⇒ 10, le suivant de 14 ⇒ 16, le suivant de 27 ⇒ 32, le suivant de 37 ⇒ 40, le suivant de 44 ⇒ 47, le suivant de 54 ⇒ 55, le suivant de 57 ⇒ 70, le suivant de 88 ⇒ 89, le suivant de 93 ⇒ 94, le suivant de 94 ⇒ 96, le suivant de 97 ⇒ 99.

Exercice

Donner l'algorithme pour la requête à intervalle :
« trouver toutes les valeurs de l'arbre comprises entre les bornes ***a*** et ***b*** »

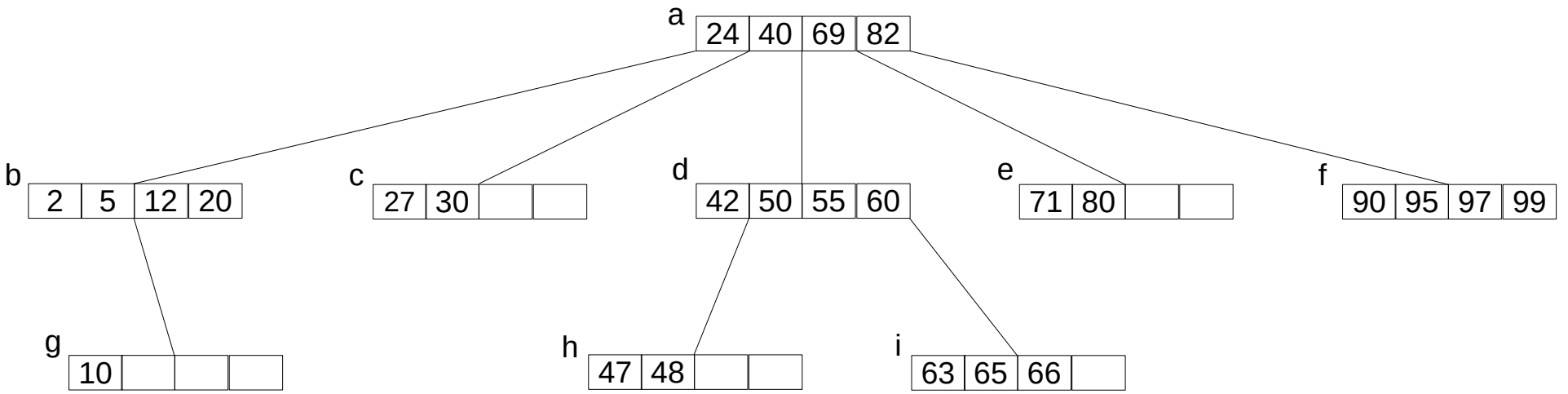
Indications :

- utiliser l'algorithme de recherche avec une *pile*, pour localiser la borne ***a***

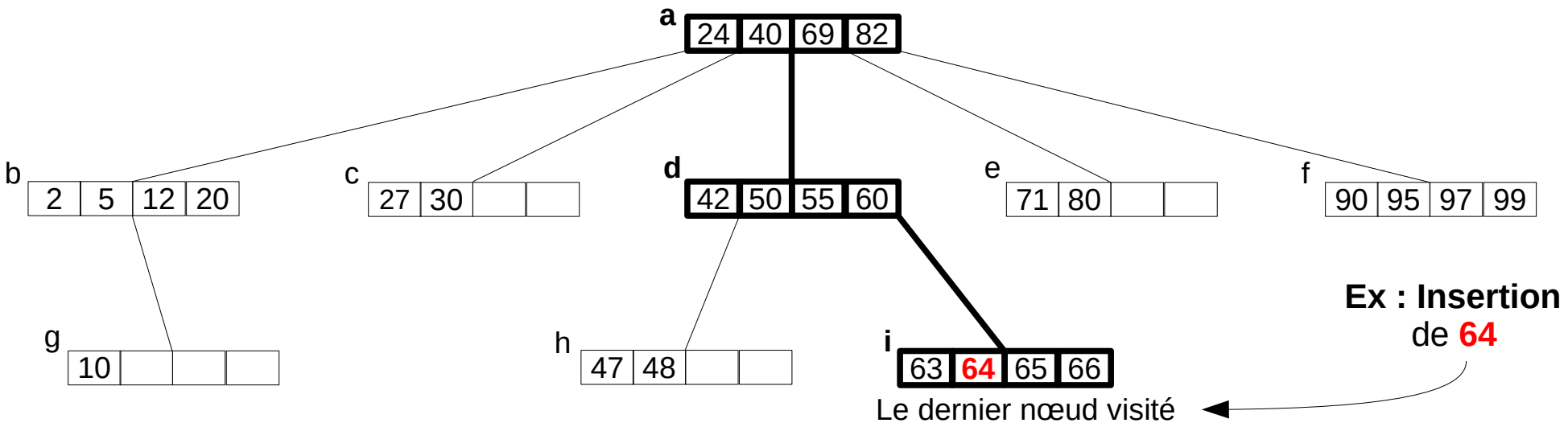
- utiliser la *pile* pour accéder aux ***suivants inordre*** et parcourir ainsi l'intervalle jusqu'à atteindre ou dépasser la borne ***b***

Mécanisme d'insertion

Si le dernier nœud visité par la recherche n'est pas plein ...

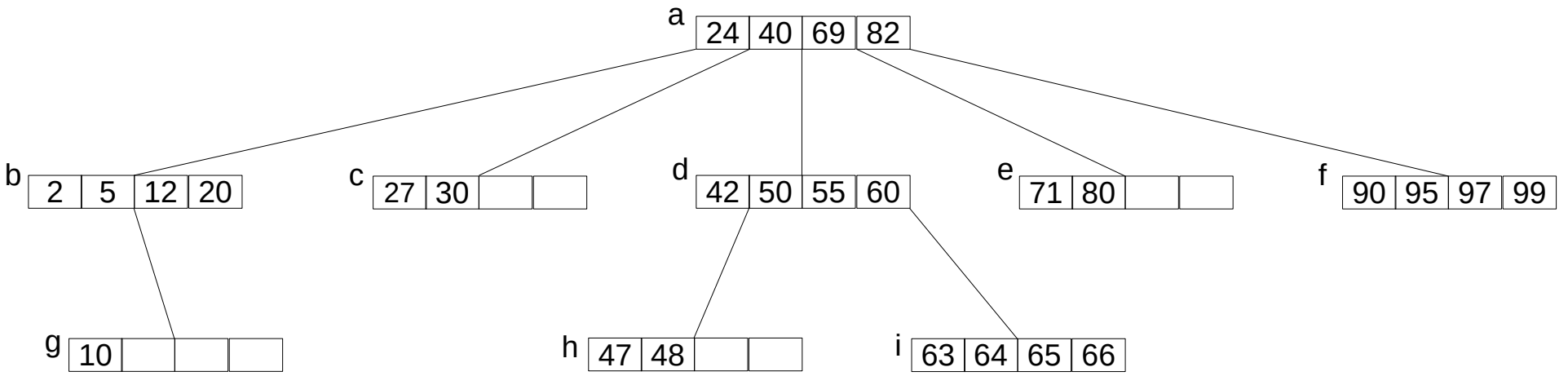


Alors insérer la nouvelle valeur dans le dernier nœud visité (décalages internes au bloc)

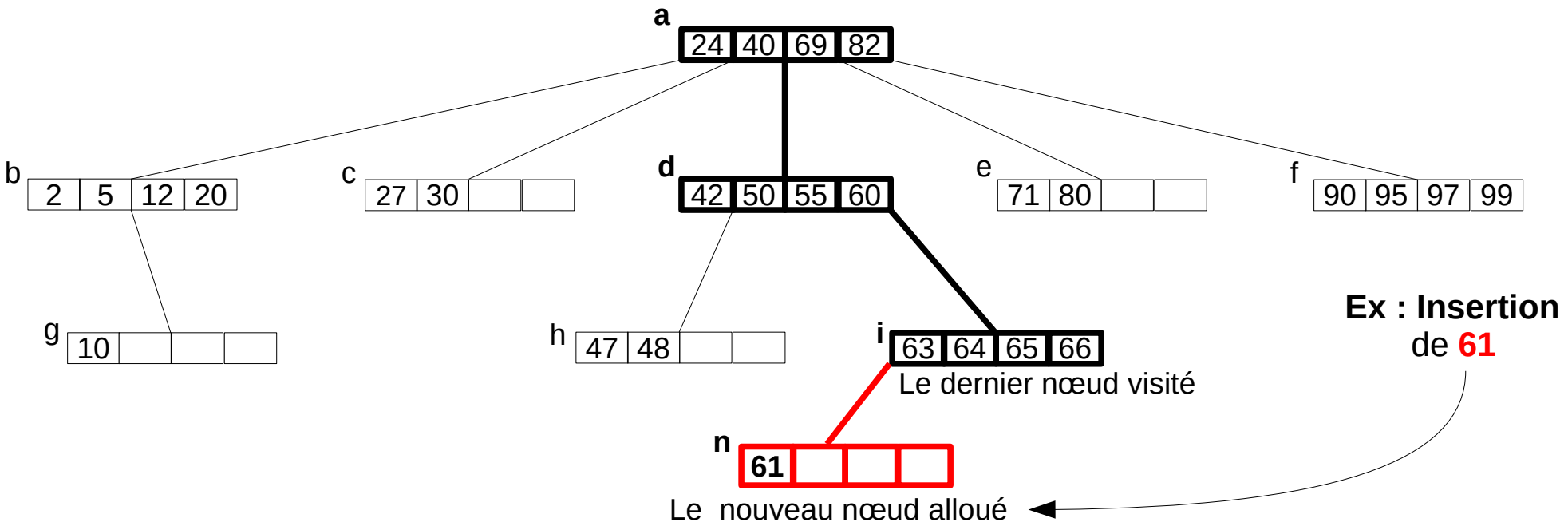


Mécanisme d'insertion

Si le dernier nœud visité par la recherche est déjà à plein 100 % ...



Alors **allouer un nouveau bloc** contenant la nouvelle valeur et le **connecter** avec l'arbre



Mécanisme d'insertion

Pour insérer une nouvelle valeur V dans un arbre de recherche m-aire, il faut :

1- Rechercher V pour vérifier qu'elle n'existe pas et pour localiser le dernier nœud visité P. La recherche retourne aussi l'indice k où devrait se trouver la valeur V pour maintenir l'ordre des valeurs dans P.

2- SI P n'est pas plein,

2.1- Insérer V dans P, par décalages afin de garder le tableau de valeurs ordonné.

SINON

2.2- Allouer un nouveau bloc Q, contenant une seule valeur V et 2 fils à -1

2.3- Connecter Q comme $fils_k$ de P (ce dernier $fils_k$ était forcément à -1)

FSI

Remarques

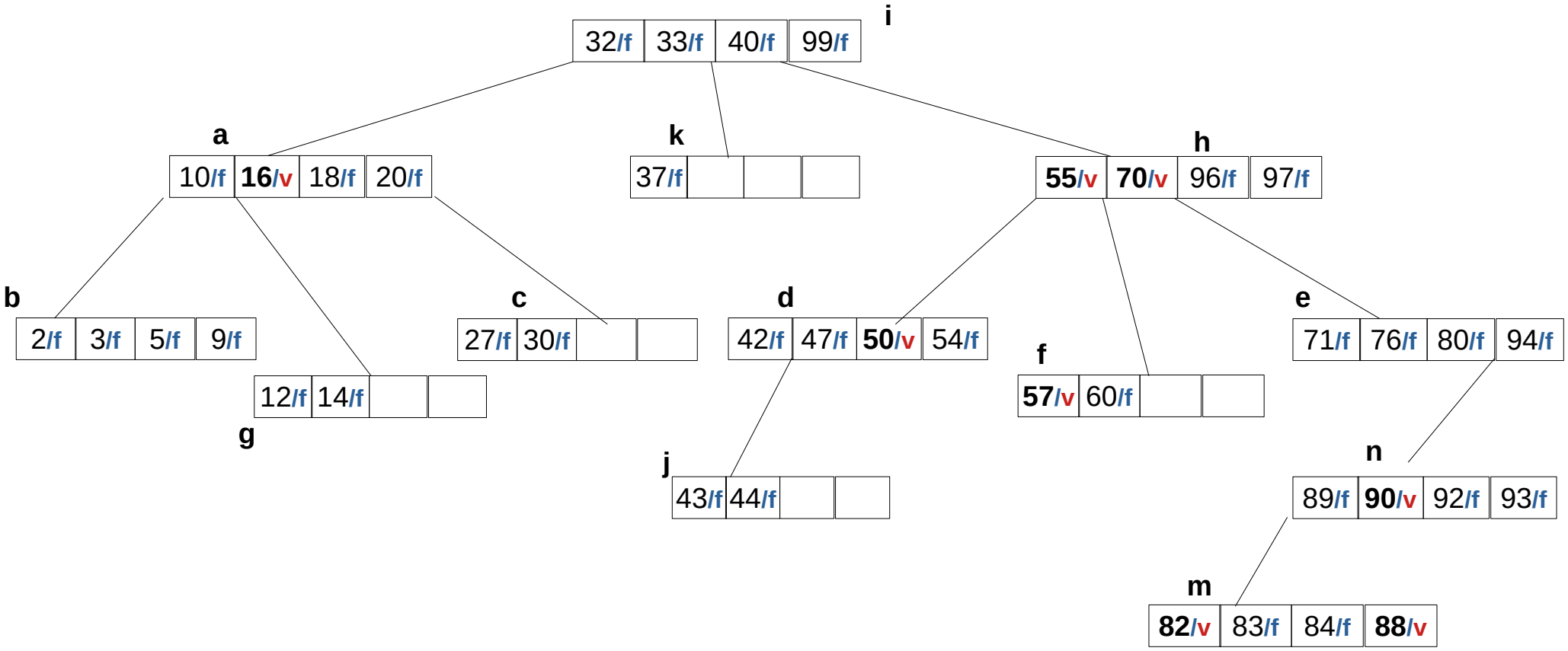
- Maintient la propriété ***Top-Down***

- Peut engendrer un fort déséquilibre de l'arbre
  → Nécessite des réorganisations périodiques

Mécanisme de suppression

- **Suppression Logique** :

Ajout d'un *indicateur d'effacement logique* au niveau de chaque valeur



Dans cet exemple, les valeurs 16, 50, 55, 57, 70, 82, 88 et 90 ont été supprimée

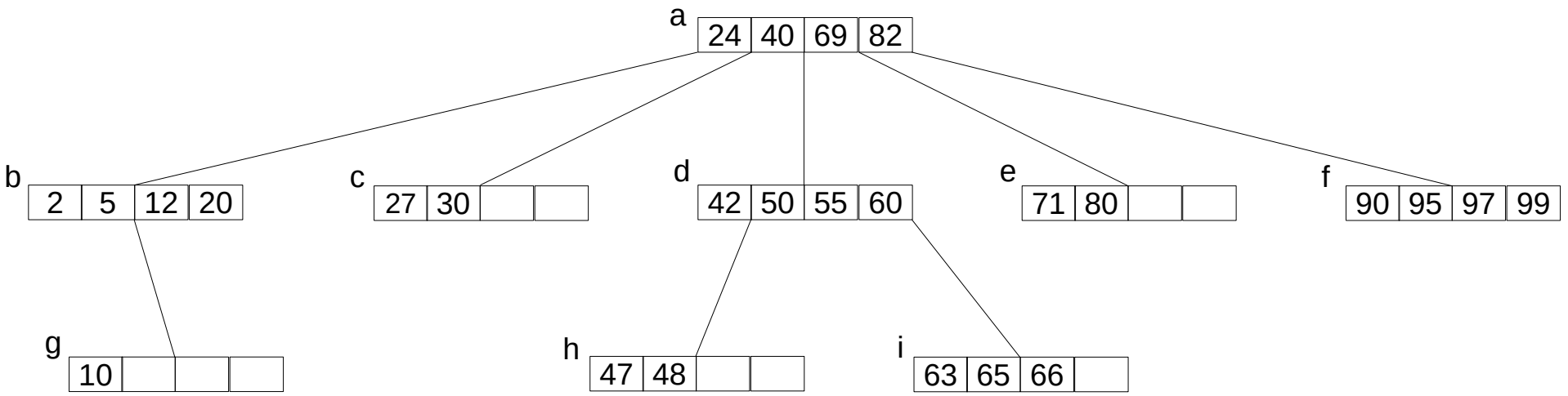
Mécanisme de suppression

- **Suppression Physique** d'une valeur v :

1. Rechercher le nœud contenant la valeur v ⇒ p

2. SI p est une feuille
   suppression de v par décalages ; SI p devient vide, libérer p et m-a-j son père FSI
   Stop

3. SINON // donc p est un nœud interne
   3.1 rechercher le suivant ou le précédent inordre ⇒ valeur : v' et nœud : p'
   3.2 remplacer v par v' dans p
   3.3 v ← v' ; p ← p' ; Aller à 2



- **Suppression Physique** d'une valeur v :

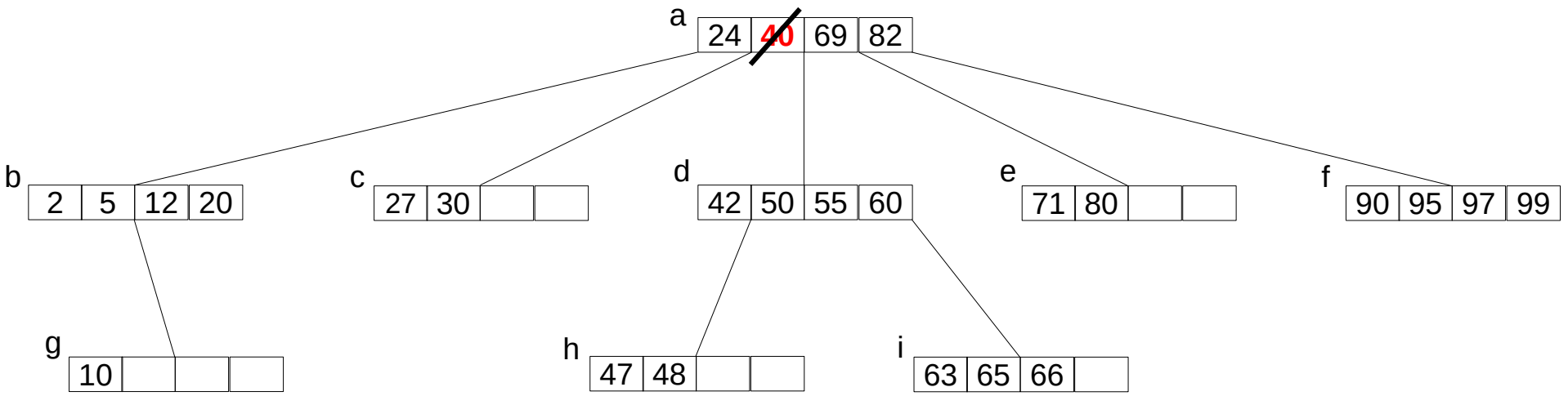
1. Rechercher le nœud contenant la valeur v ⇒ p

2. SI p est une feuille
   suppression de v par décalages ; SI p devient vide, libérer p et m-a-j son père FSI
   Stop

3. SINON // donc p est un nœud interne
   3.1 rechercher le suivant ou le précédent inordre ⇒ valeur : v' et nœud : p'
   3.2 remplacer v par v' dans p
   3.3 v ← v' ; p ← p' ; Aller à 2

Ex : sup (40)

1 rech de 40 ⇒ **(a,2)** // un nœud interne ...



- **Suppression Physique** d'une valeur v :

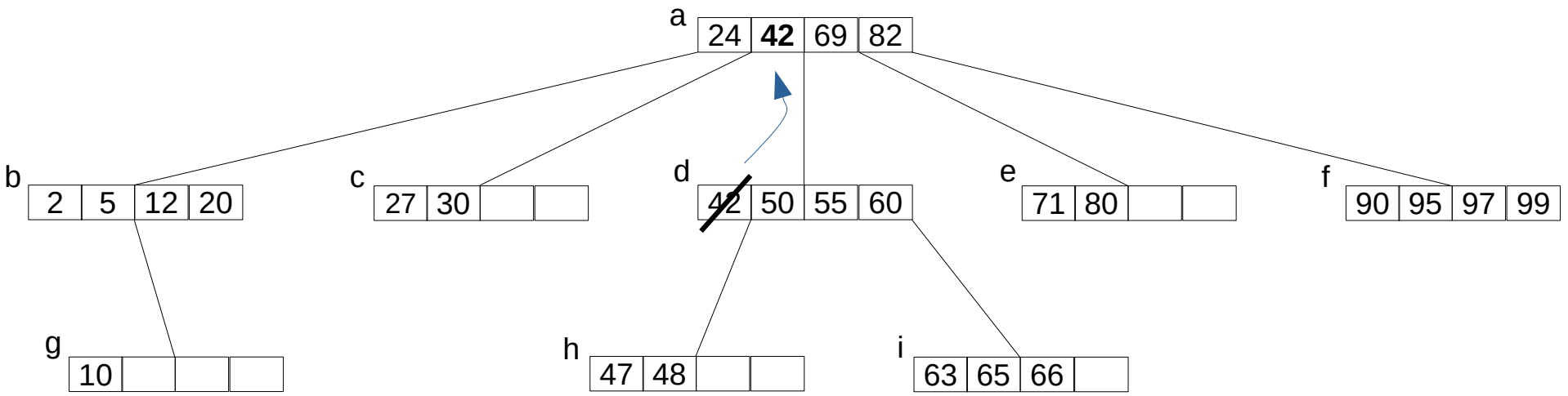
1. Rechercher le nœud contenant la valeur v ⇒ p

2. SI p est une feuille
   suppression de v par décalages ; SI p devient vide, libérer p et m-a-j son père FSI
   Stop

3. SINON // donc p est un nœud interne
   3.1 rechercher le suivant ou le précédent inordre ⇒ valeur : v' et nœud : p'
   3.2 remplacer v par v' dans p
   3.3 v ← v' ; p ← p' ; Aller à 2

Ex : sup (40)

1 rech de 40 ⇒ **(a,2)** // un nœud interne …
3 suiv. Inordre de 40 = 42 **(d,1)** et **remplacement** de 40 par 42 dans **a**
// d est aussi un nœud interne ...



- **Suppression Physique** d'une valeur v :

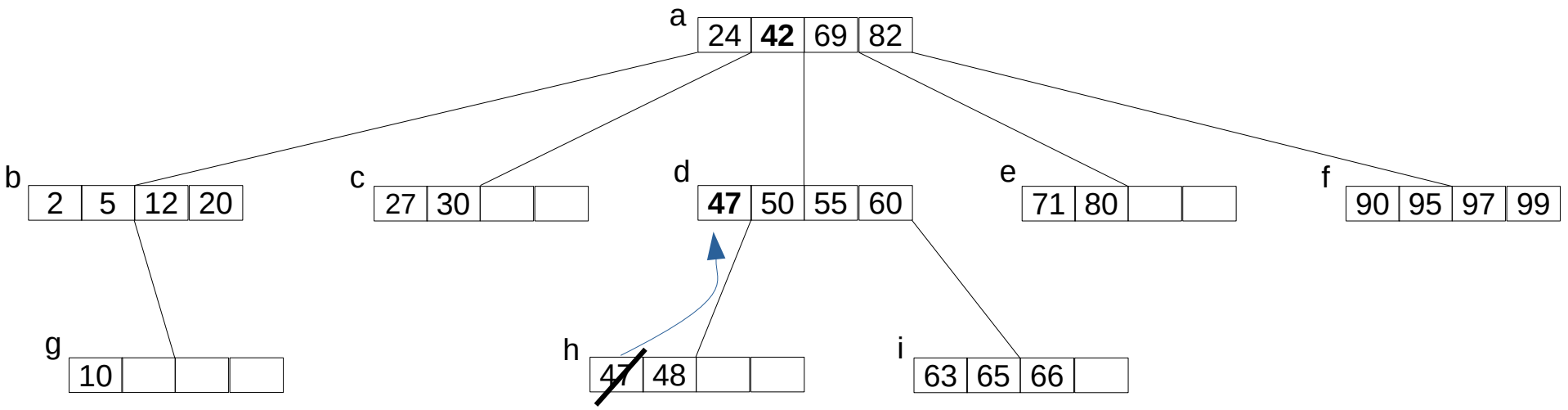
1. Rechercher le nœud contenant la valeur v ⇒ p

2. SI p est une feuille
   suppression de v par décalages ; SI p devient vide, libérer p et m-a-j son père FSI
   Stop

3. SINON // donc p est un nœud interne
   3.1 rechercher le suivant ou le précédent inordre ⇒ valeur : v' et nœud : p'
   3.2 remplacer v par v' dans p
   3.3 v ← v' ; p ← p' ; Aller à 2

Ex : sup (40)

1 rech de 40 ⇒ **(a,2)** // un nœud interne …
3 suiv. Inordre de 40 = 42 **(d,1)** et remplacement de 40 par 42 dans **a**
// d est aussi un nœud interne …
3 suiv. Inordre de 42 = 47 **(h,1)** et **remplacement** de 42 par 47 dans **d**
// h est un nœud feuille …



- **Suppression Physique** d'une valeur v :

1. Rechercher le nœud contenant la valeur v ⇒ p

2. SI p est une feuille
   suppression de v par décalages ; SI p devient vide, libérer p et m-a-j son père FSI
   Stop

3. SINON // donc p est un nœud interne
   3.1 rechercher le suivant ou le précédent inordre ⇒ valeur : v' et nœud : p'
   3.2 remplacer v par v' dans p
   3.3 v ← v' ; p ← p' ; Aller à 2

Ex : sup (40)

1 rech de 40 ⇒ **(a,2)** // un nœud interne …
3 suiv. Inordre de 40 = 42 **(d,1)** et remplacement de 40 par 42 dans a
// d est aussi un nœud interne …
3 suiv. Inordre de 42 = 47 **(h,1)** et remplacement de 42 par 47 dans d
// h est un nœud feuille …
2 suppression de 47 par **décalages** dans **h**.