

NFP136- Cours 7

ARBRES DE RECHERCHE

PLAN

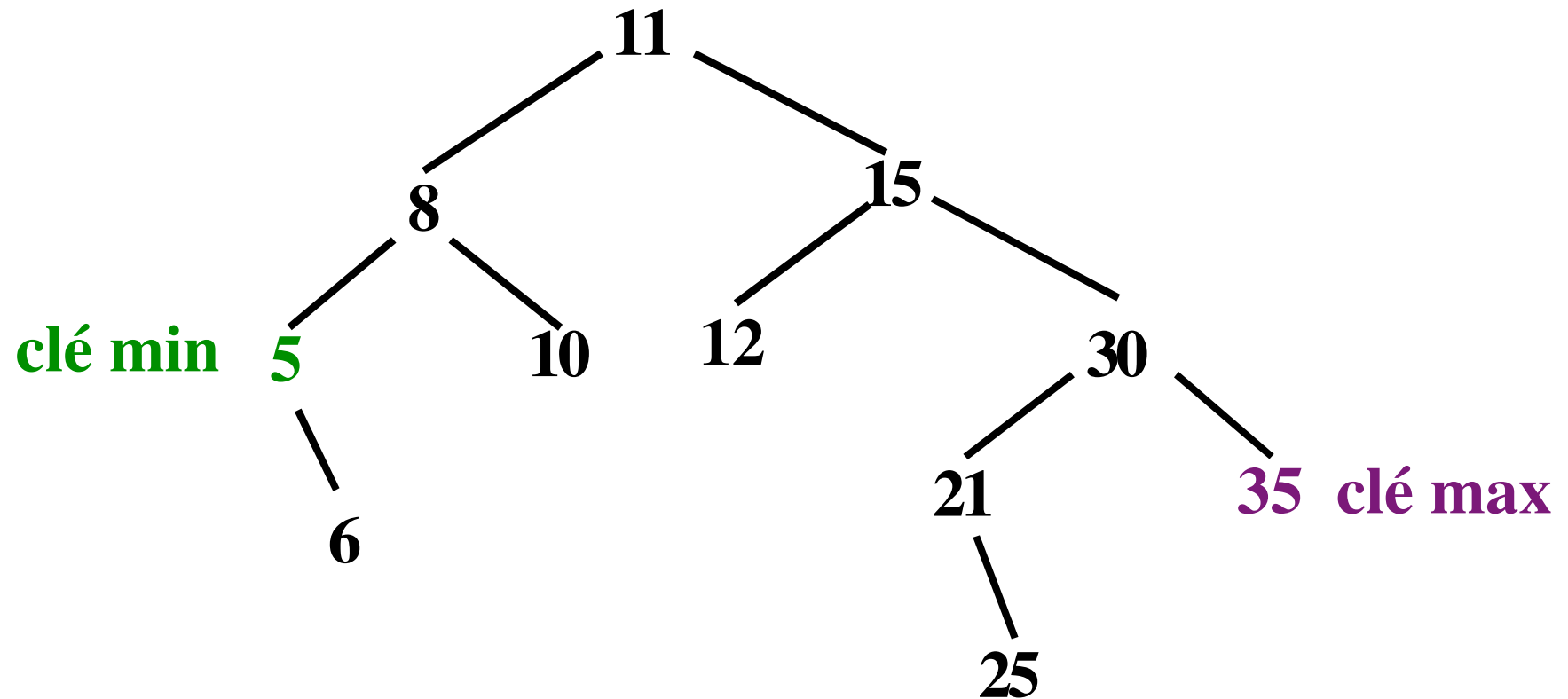
- **Arbres binaires de recherche**
- **Arbre h-équilibrés**
- **Arbres balancés (b-arbres)**
- **Fichiers séquentiels indexés**

7.1 ARBRES BINAIRES DE RECHERCHE

Définition

Arbre binaire tel qu'en tout nœud la clé du nœud est supérieure à celle de tous ses descendants de gauche et inférieure à celle de tous ses descendants de droite

EXEMPLE



RAPPEL: Voir cours n° 3 sur les arbres

ABR: **structures de données pour grands
volumes variables d'informations
structures dynamiques**

Arbre binaire a

a.g = sous-arbre de gauche de racine(a)

a.d = sous-arbre de droite de racine(a)

Clés **une donnée → une clé**
 {clés} ensemble totalement ordonné

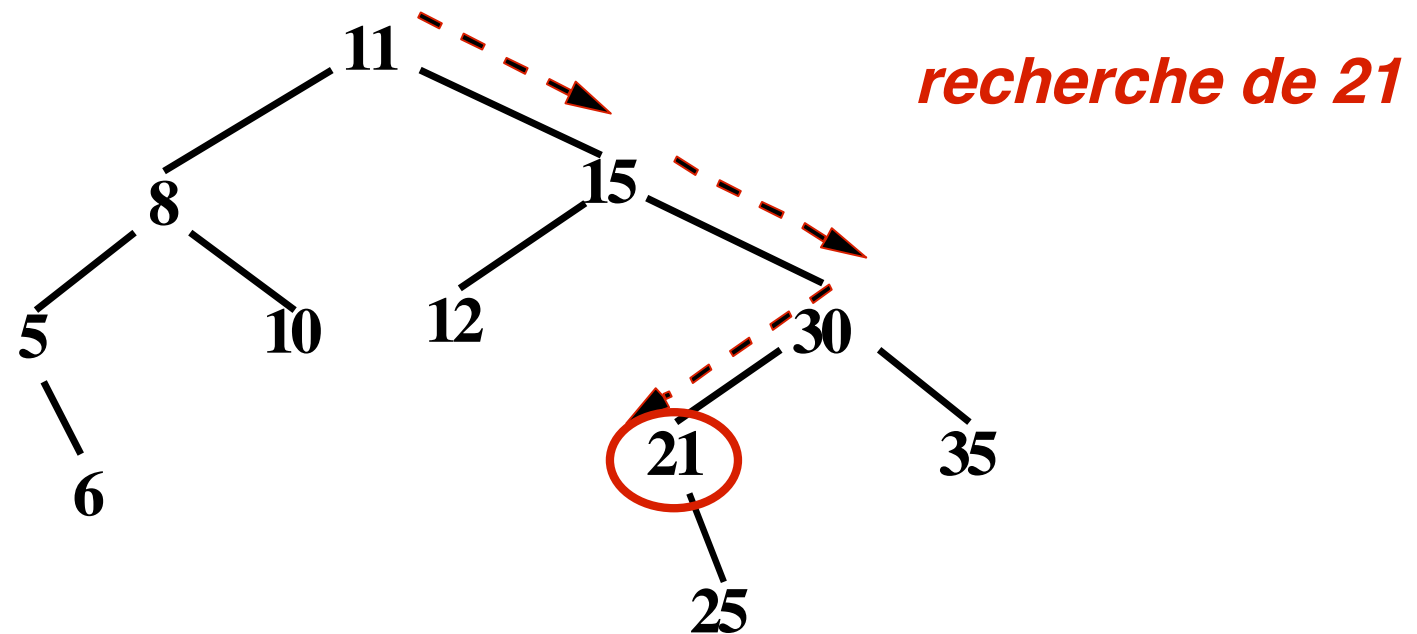
```
class ABR {  
    private int cle;  
    private String element;  
    private ABR gauche;  
    private ABR droit;  
  
    public ABR(int a, String s, ABR g, ABR d) {  
        cle = a;  
        element = s;  
        filsG= g;  
        filsD=d;  
    }  
}
```

//les méthodes

Recherche :

**descente de l'arborescence avec
comparaison en chaque nœud**

EXEMPLE



principe de la recherche de l'élément e de clé c dans l'arbre a

précondition: $c \in a$

recherche(c,a) renvoie String

si $c == a.cle$

alors retourner (a.element)

//élément de la racine de a

sinon

si $c < a.cle$

alors retourner (recherche(c,a.gauche))

sinon retourner (recherche(c,a.droit))

finsi

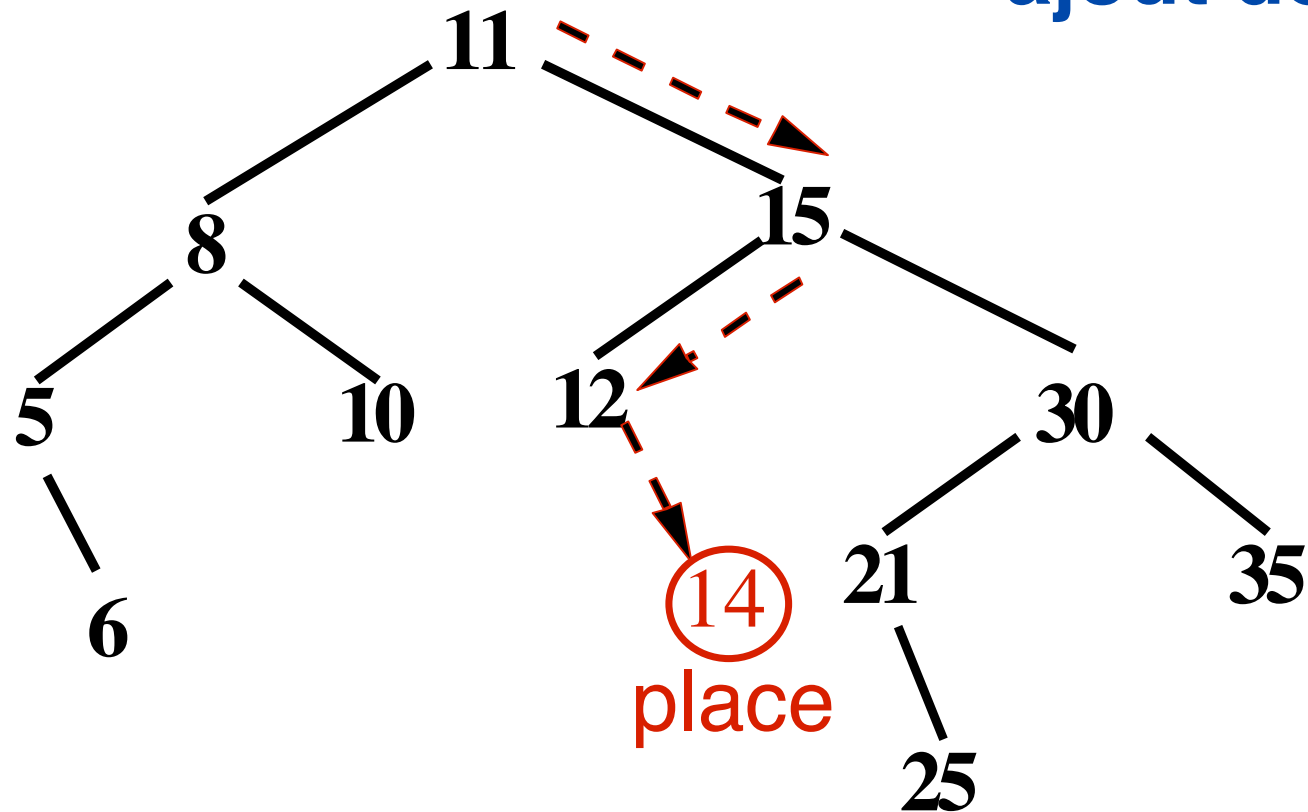
finsi

complexité: $O(h(a))$

ADJONCTION D'UN ELEMENT

EXEMPLE

ajout de 14



complexité: $O(h(a))$

adjonction dans a d'un élément e
de clé c

ABR ajouter (ABR a, String e, int c)

b=a;

si b == null alors b=new ABR (c, e, null, null) ;

sinon tant que non place_trouvée faire

si b.cle == c alors "erreur" //élément déjà présent

si b.cle > c

alors //descendre à gauche

si b.gauche == null alors place_trouvée=vrai //fils gauche de b

b.gauche=new ABR (c, e, null, null) ;

sinon b = b.gauche;

finsi;

sinon //descendre à droite

si b.droit == null alors place trouvée=vrai //fils droit

b.droit=new ABR (c, e, null, null) ;

sinon b =b.droit;

finsi;

finsi;

fait;

finsi;

retourner b;

suppression d'un nœud N

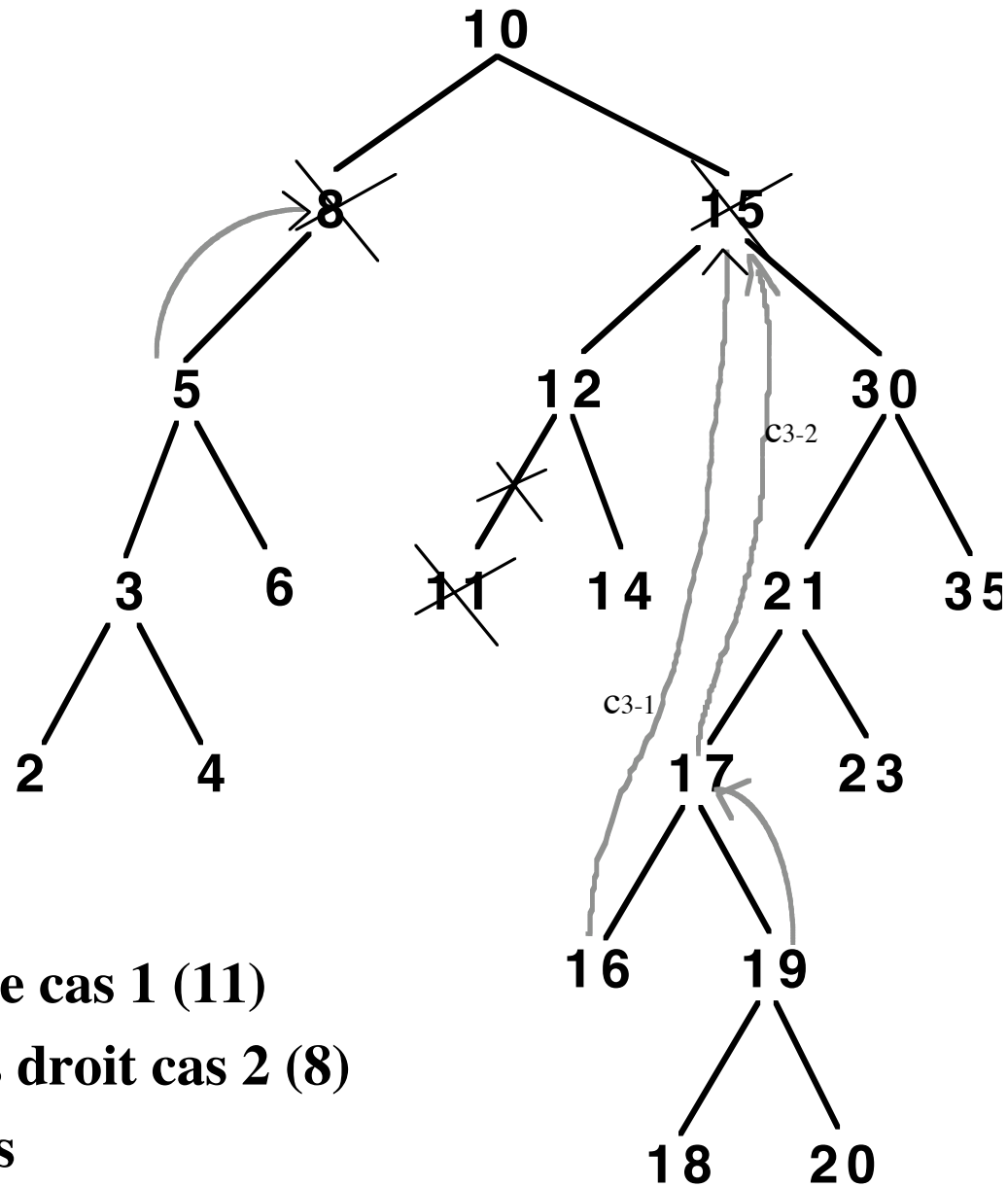
cas 1: N est une feuille

cas 2: N a un seul fils F

cas 3: N a 2 fils

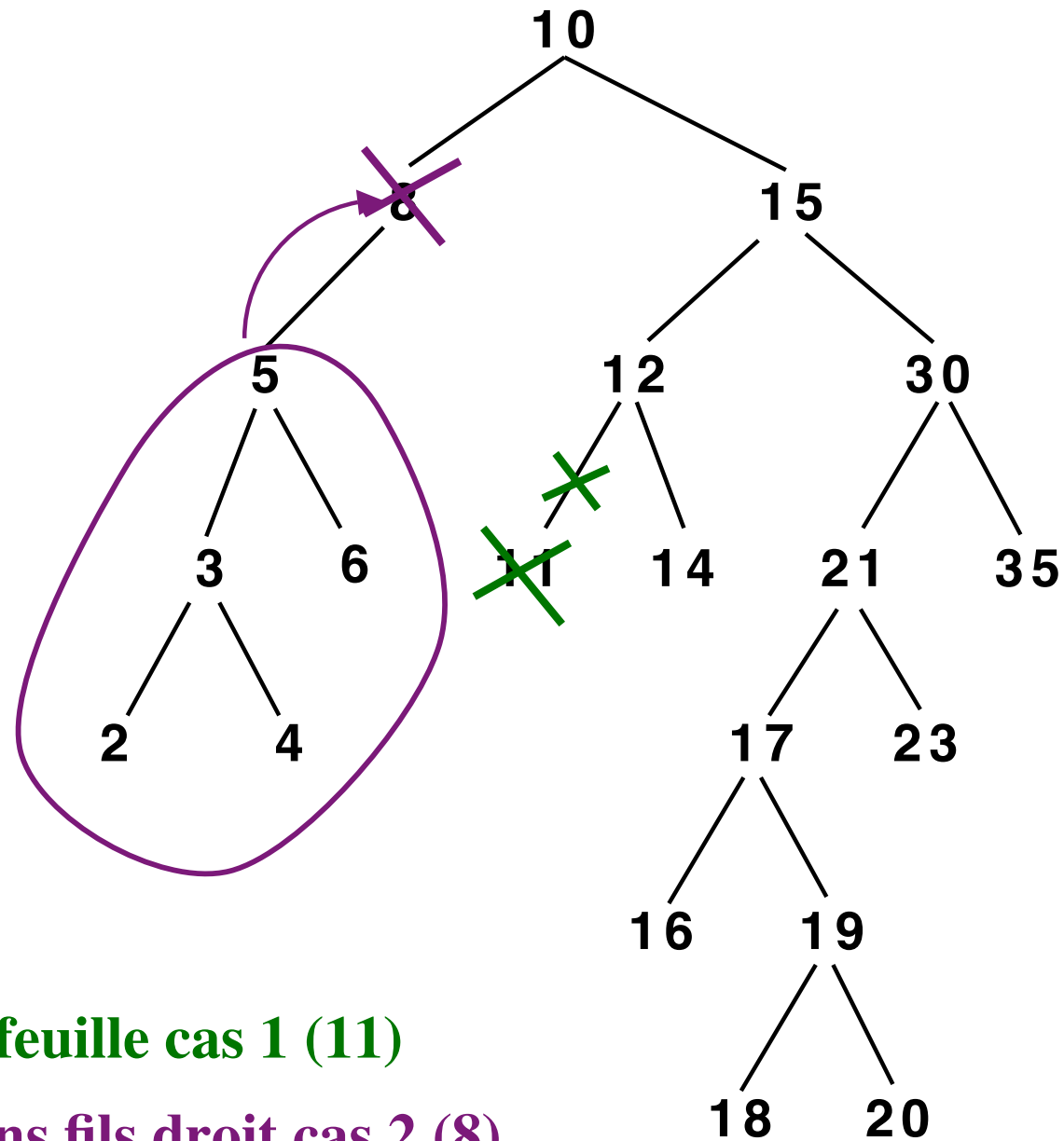
(2 cas cas 3-1 cas 3-2)

EXEMPLE



suppression d'une feuille cas 1 (11)
sup. d'un nœud sans fils droit cas 2 (8)
sup. de nœuds avec 2 fils
cas 3-1 (15) puis cas 3-2 (16)

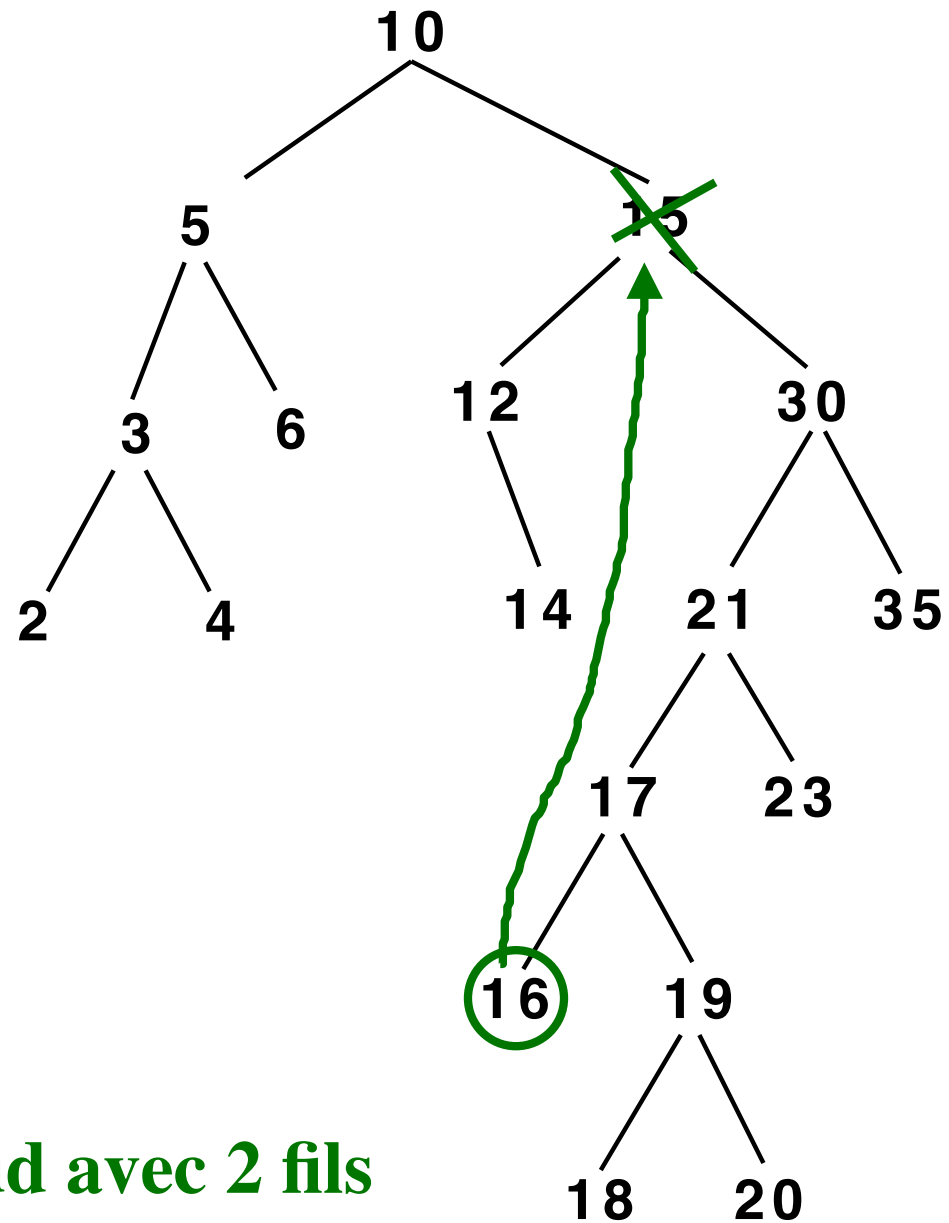
EXEMPLE



suppression d'une feuille cas 1 (11)

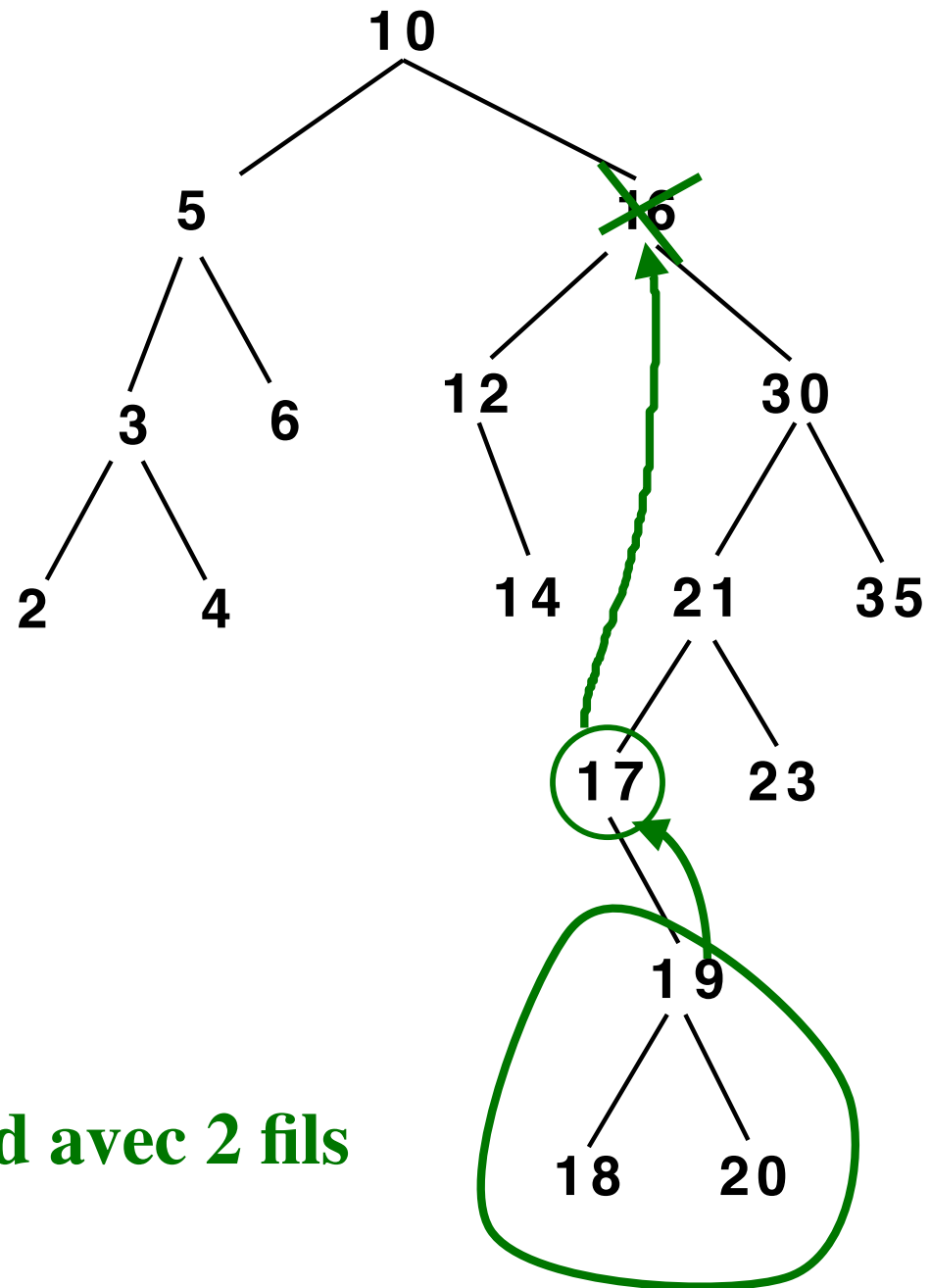
sup. d'un nœud sans fils droit cas 2 (8)

EXEMPLE



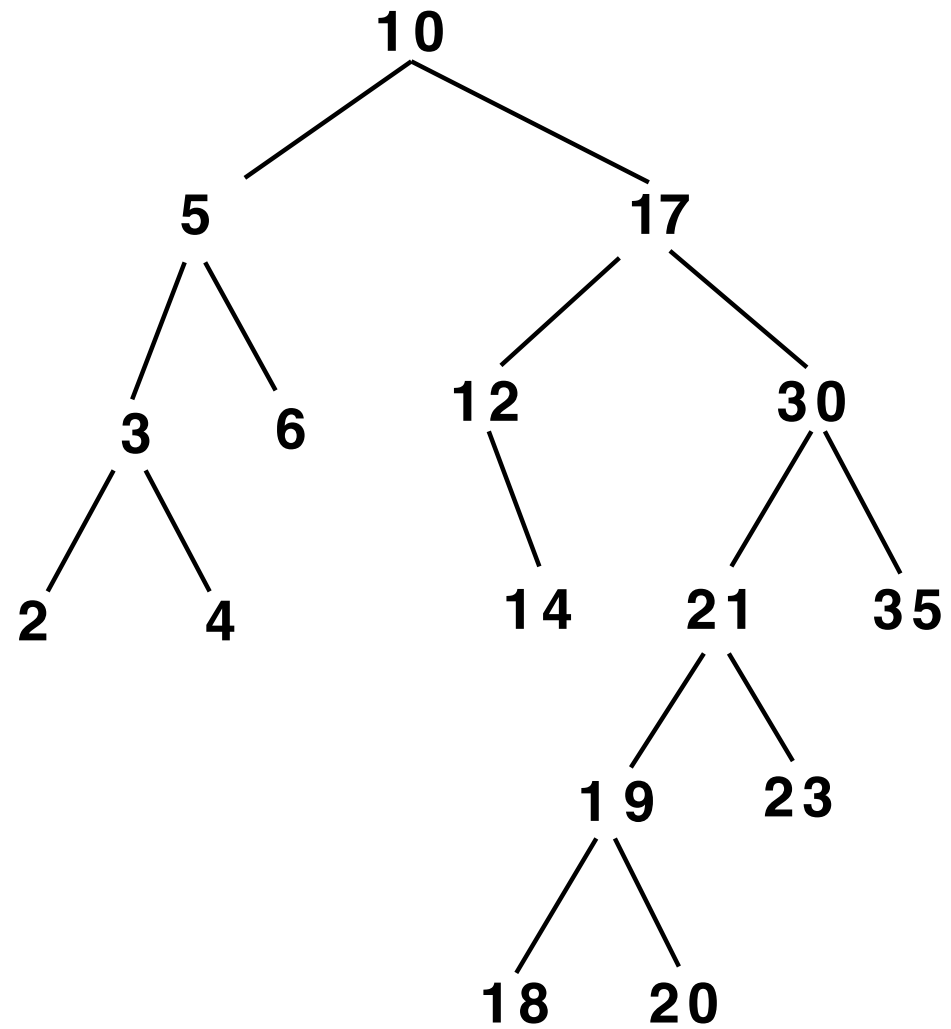
suppression d'un nœud avec 2 fils
cas 3-1 (15)

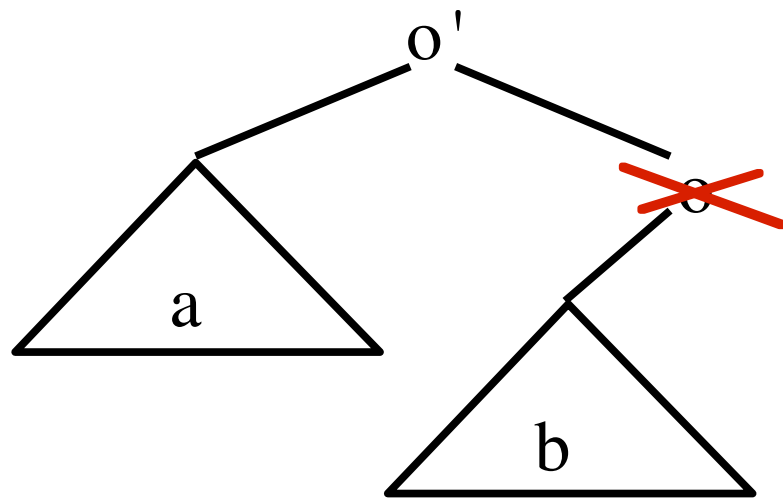
EXEMPLE



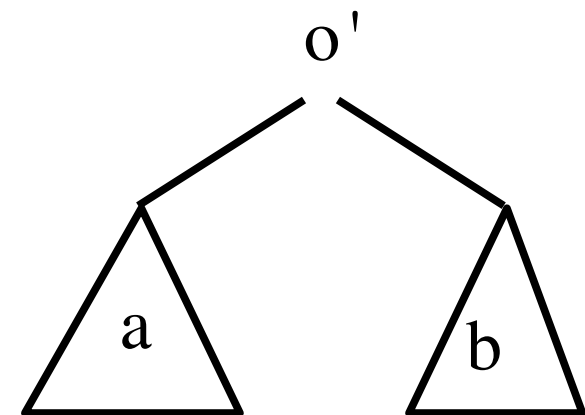
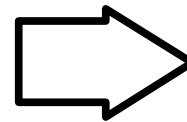
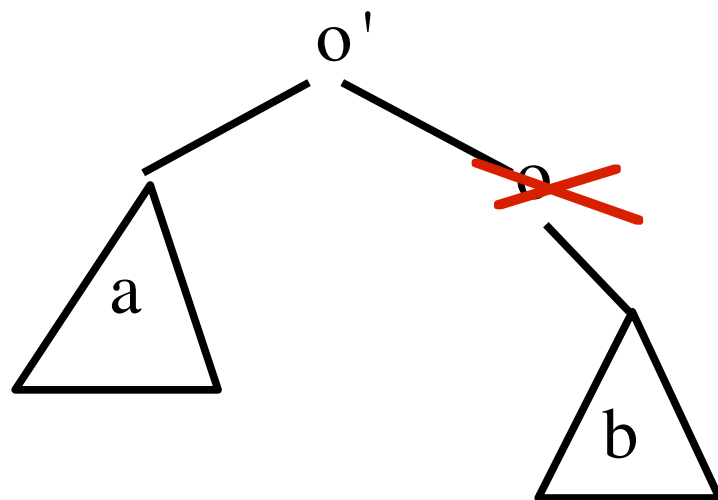
suppression d'un nœud avec 2 fils
cas 3-2 (16)

EXAMPLE

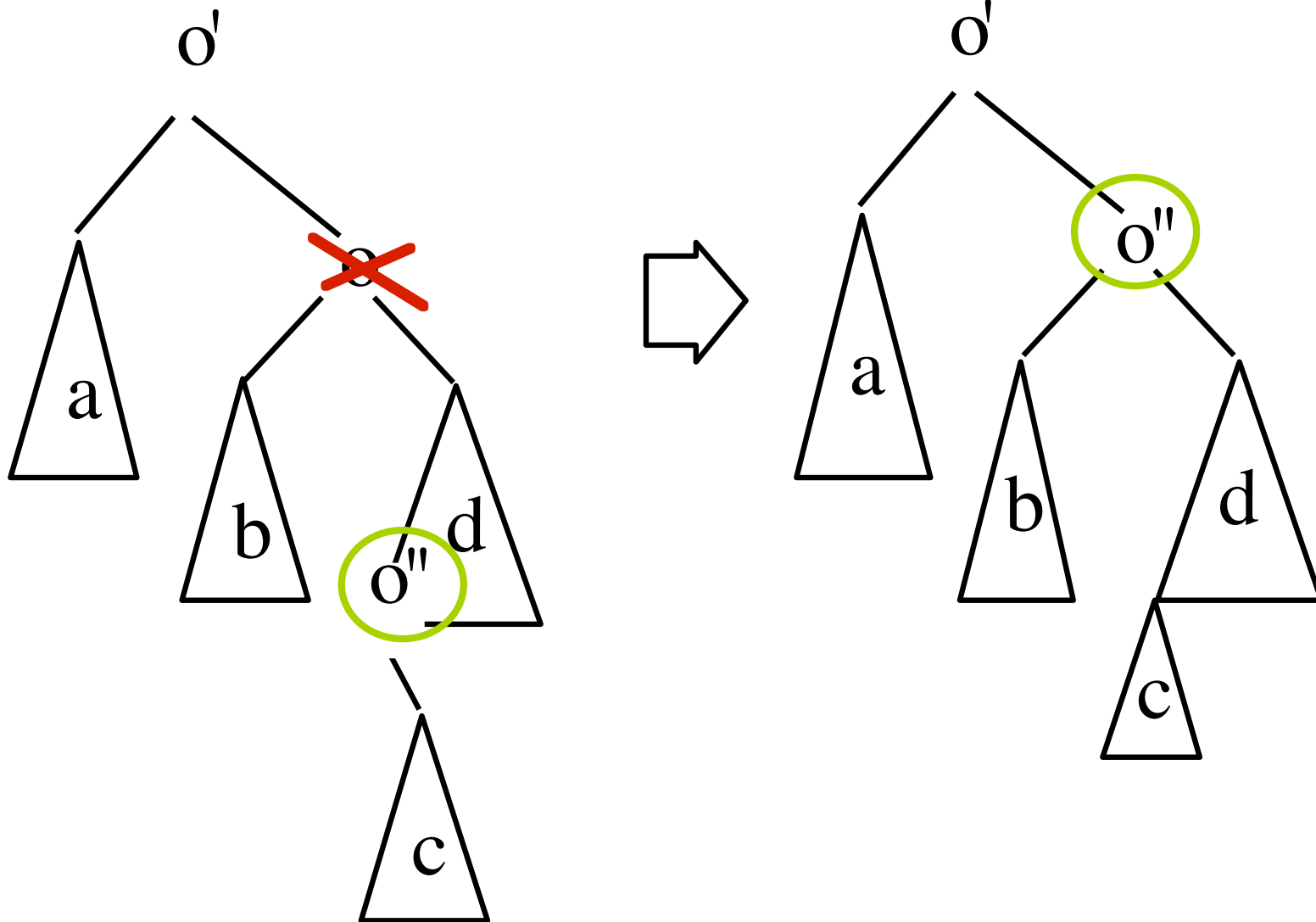




cas 2



Cas 3.2



suppression d'un nœud N

cas 1: N est une feuille

→ suppression simple

cas 2: N a un seul fils F

→ F prend la place de N

(remontée du sous-arbre)

cas 3: N a 2 fils

→ est remplacé par le nœud de plus petite clé de son sous-arbre de droite

cas 3-1: c'est une feuille

cas 3-2: ce noeud n'a qu'un fils droit qui prend sa place