CHAP 3 - 2 :

ARBRE BINAIRE DE RECHERCHE

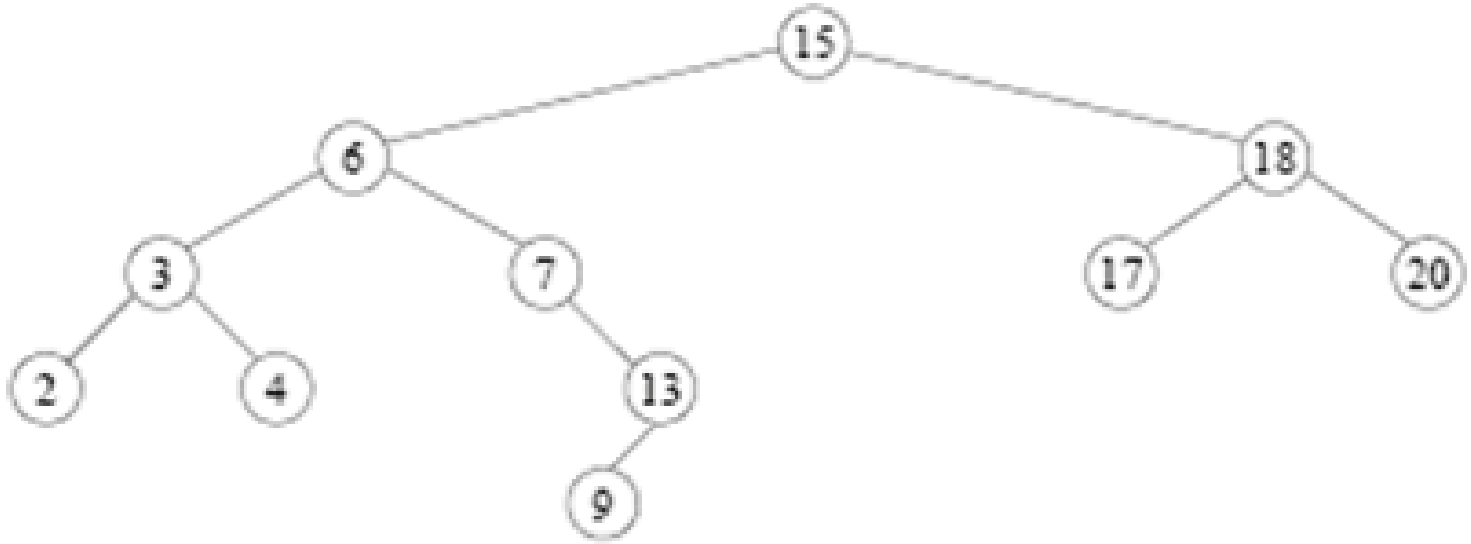
Université Sétif 1
Faculté des sciences
Département d'informatique

Algorithmique et Structures de Données

2017-2018 / Dr. L.Douidi

Définition

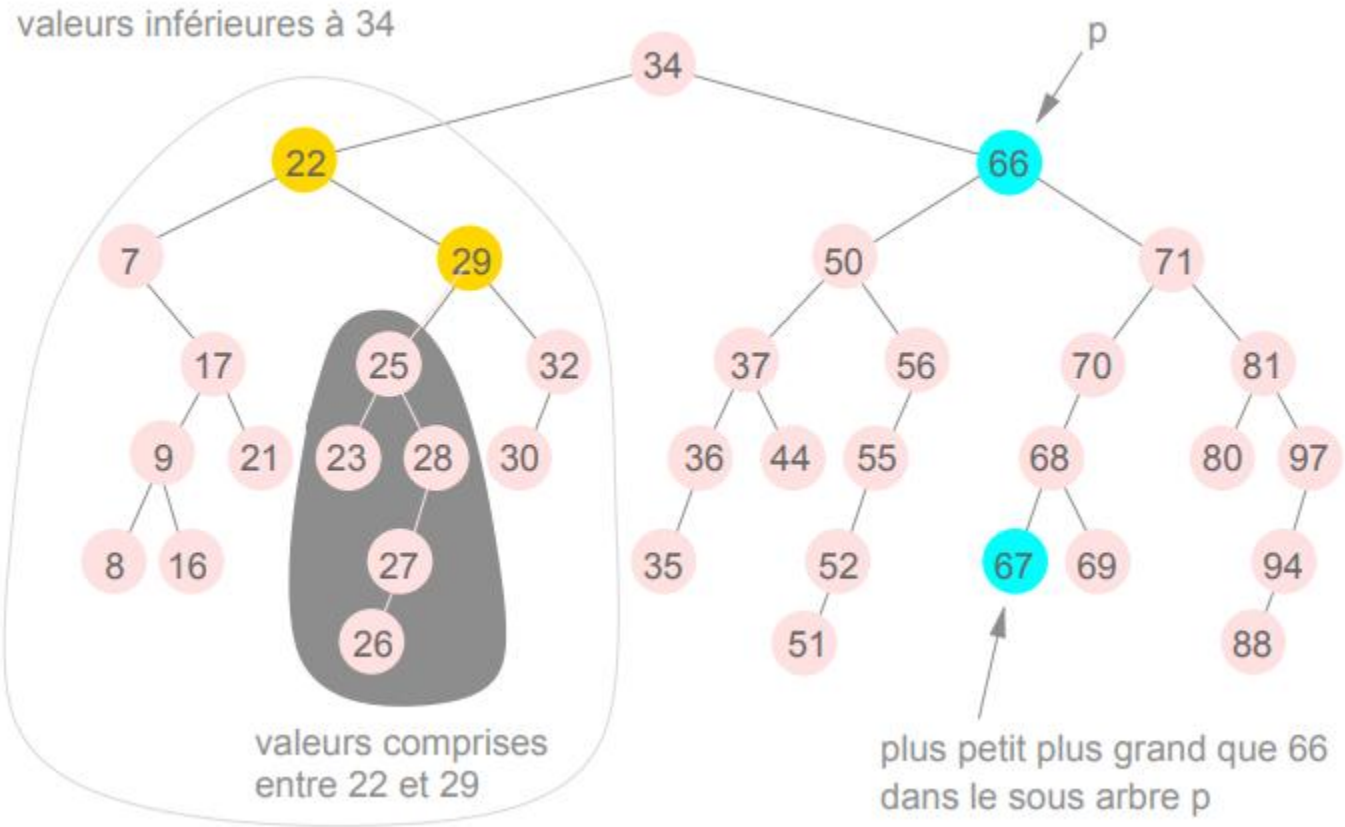
- Un arbre binaire de recherche (ou ABR) est une structure de donnée qui permet de représenter un ensemble de valeurs si l'on dispose d'une relation d'ordre sur ces valeurs.
- Les arbres binaires de recherche sont utilisés pour accélérer la recherche dans les arbres n-aires.
- Un arbre binaire de recherche est un arbre binaire vérifiant la propriété suivante :
- soient *x et y deux nœuds de l'arbre,*
  - *si y est un nœud du sous-arbre gauche de x, alors clé(y) ≤ clé(x),*
  - *si y est un nœud du sous-arbre droit de x, alors clé(y) ≥ clé(x).*



ABR

- Soit E un ensemble muni d'une relation d'ordre
- Un arbre binaire **a** étiqueté avec des éléments de E est un arbre binaire de recherche si pour tout nœud p = h(x,G,D)
  - $\forall\ q \in G$, val (q) $\leq$ x,
  - $\forall\ q \in D$, val (q) $\geq$ x.

Arbres binaires de recherche



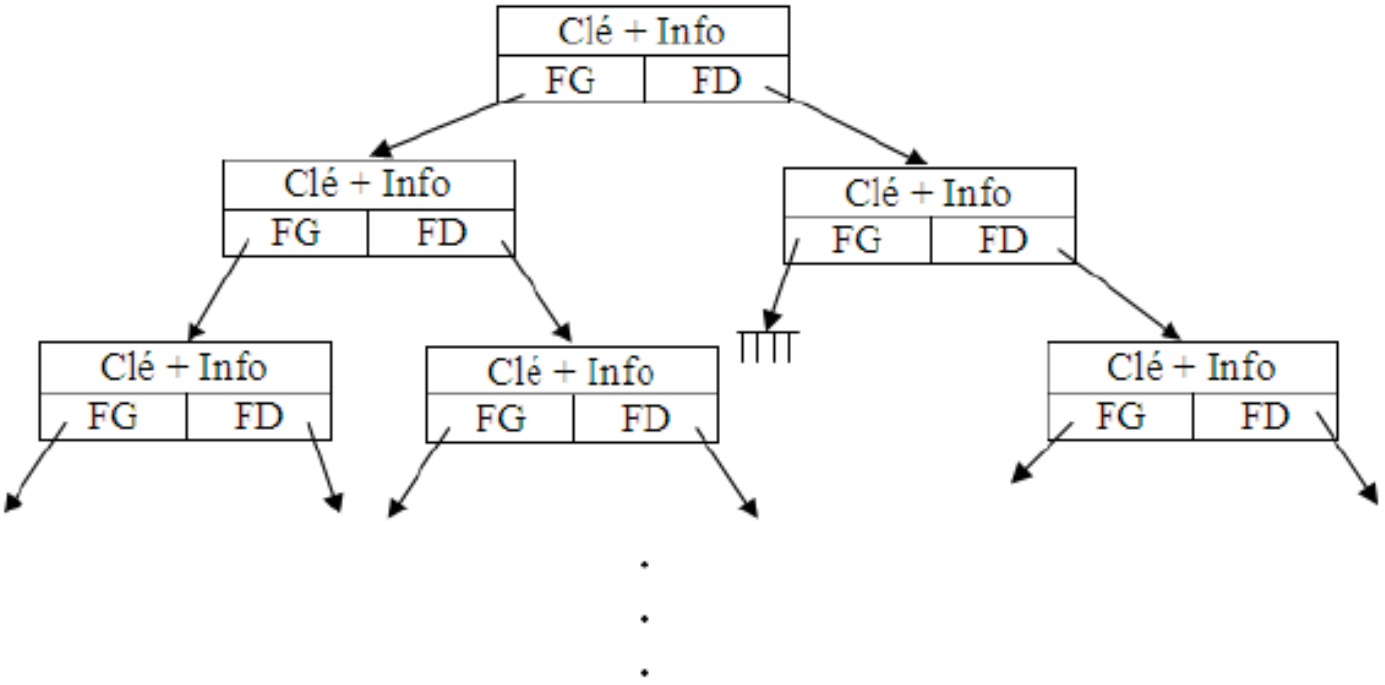
ABR: implémentation

- Les arbres de recherche binaires sont implémentés de la même manière que celles n-aires (statique ou dynamique)
- Représentation Statique

Num	Information	Fils gauche	Fils droit
1	15	2	3
2	6	4	5
3	18	6	7
4	3	8	9
5	7	0	10
6	17	0	0
7	20	0	0
8	2	0	0
9	4	0	0
10	13	11	0
11	9	0	0

ABR: implémentation

Représentation dynamique:




```
Type TNoeud = Structure
   Clé : entier ;
   Info : typeqq ;
   FG,FD : Pointeur(TNoeud) ;
Fin ;
Var Racine : Pointeur(TNoeud) ;
```

Déclaration

```
typedef struct noeud *ARBRE;
struct noeud {
   VALEUR val;
   ARBRE fg, fd, pere;
};
```

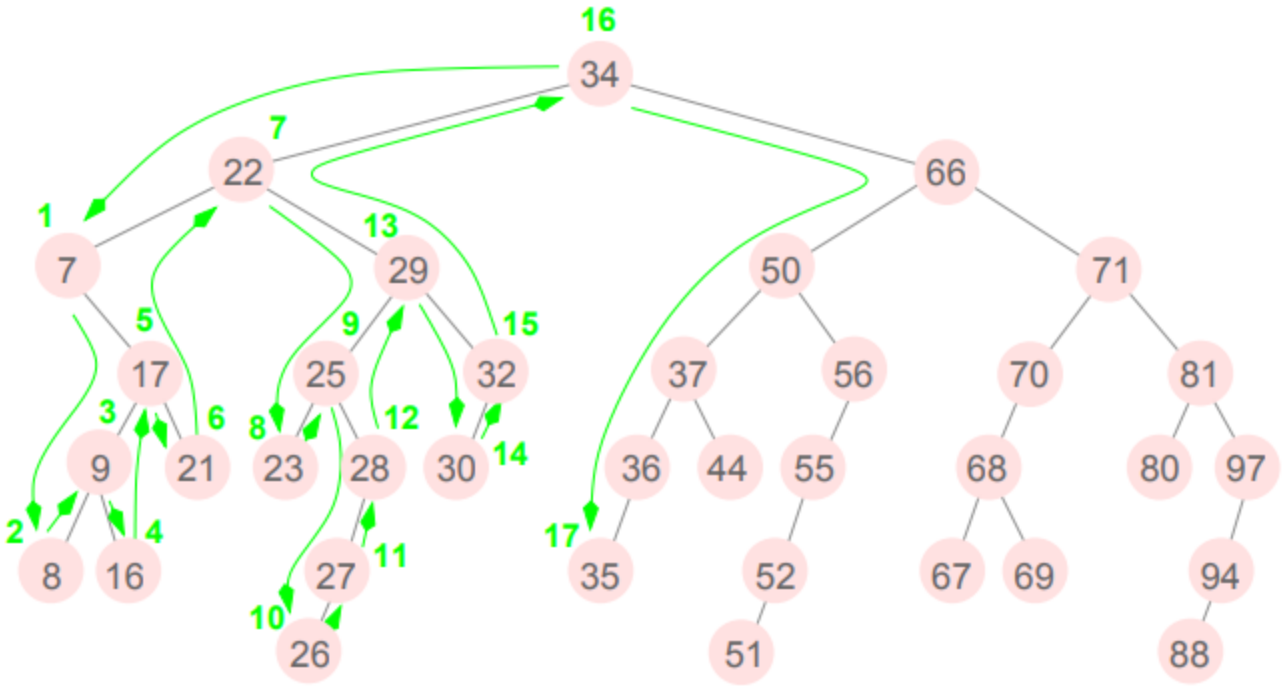
L'accès au père est utile par exemple pour chercher le successeur d'un nœud

Opérations sur ABR

Les opérations caractéristiques sur les arbres binaires de recherche sont **l'insertion**, la **suppression**, et la **recherche** d'une valeur. Ces opérations sont peu couteuses si l'arbre n'est pas trop déséquilibré. En pratique, les valeurs sont des cl´es permettant d'accéder à des enregistrements.

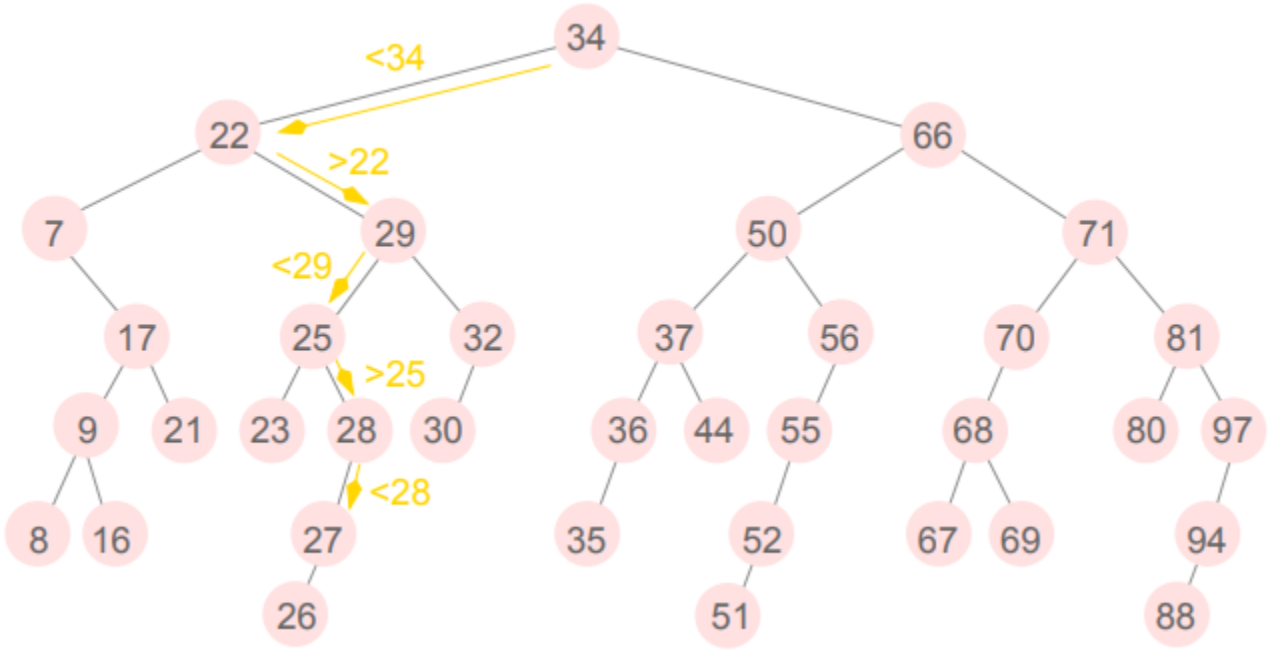
Arbres binaires de recherche : parcours infixé

- Le parcours infixé de l'arbre produit la suite ordonnée des clés
- 7 8 9 16 17 21 22 23 25 26 27 28 29 30 32 34 35 36 37 ...



ABR: recherche d'une clé

- Idée : descendre à gauche ou à droite suivant la valeur de la clé tant que la valeur n'est pas trouvée (et qu'on peut descendre)
- recherche de la valeur 27



ABR: recherche d'une clé

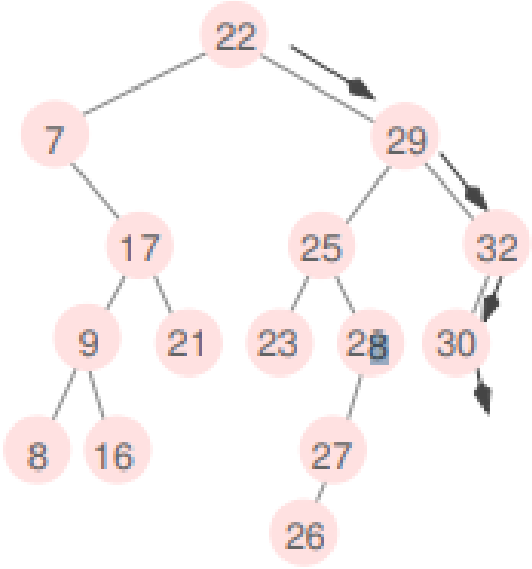
```
Fonction recherche(a, v)    // version itérative
// entrée : a est un ABR, v est une clé.
// sortie :Vrai si v figure dans a et Faux sinon.
début
Tant_que NON est_vide(a) ET v ≠ val(a) faire
    si v < val(a) alors
            a = fils_gauche(a)
    sinon
            a = fils_droit(a)
    finsi
fintq
si est_vide(a) alors
    retourner Faux
sinon
    retournerVrai
finsi
fin
```

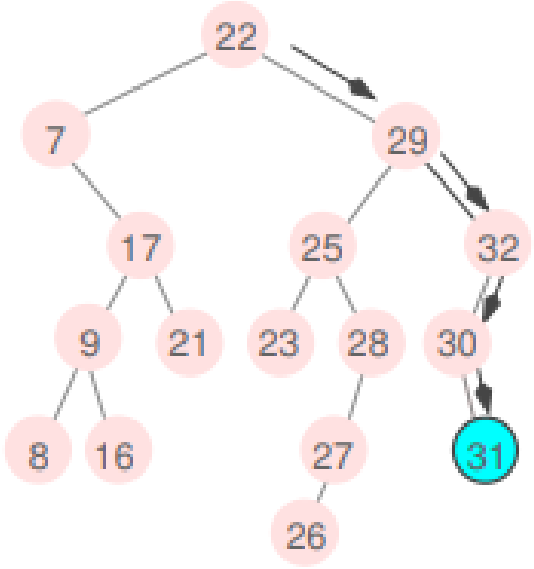
ABR: recherche d'une clé

```
Fonction recherche(a, v)    // version récursive
entrée : a est un ABR, v est une clé.
sortie :Vrai si v figure dans a et Faux sinon.
début
    si est_vide(a) alors
            retourner Faux
    sinon si v == val(a) alors
                    retourner Vrai
            sinon si v < val(a) alors
                        retourner recherche(v,fils_gauche(a))
                    sinon
                        retourner recherche(v,fils_droit(a))
                    finsi
            finsi
    finsi
fin
```

ABR: insertion d'une clé

- Idée : on fait comme pour la recherche
  Un nouveau nœud est créé avec la nouvelle valeur et inséré à l'endroit ou la recherche s'est arrêtée (ici la valeur 31)





ABR: insertion d'une clé

Algorithme **Insertion** // Insertion d'une nouvelle clé dans un ABR

entrée : a est un ABR, v est une clé.

résultat : v est insérée dans a

début

```
si est_vide(a) alors
   a=cree_arbre(v,cree_arbre_vide(),cree_arbre_vide())
sinon   si v < val(a) alors
                Insertion(v,fils_gauche(a))
          sinon    si v > val(a) alors
                        Insertion(v,fils_droit(a))
                    finsi
          finsi
finsi
```

fin

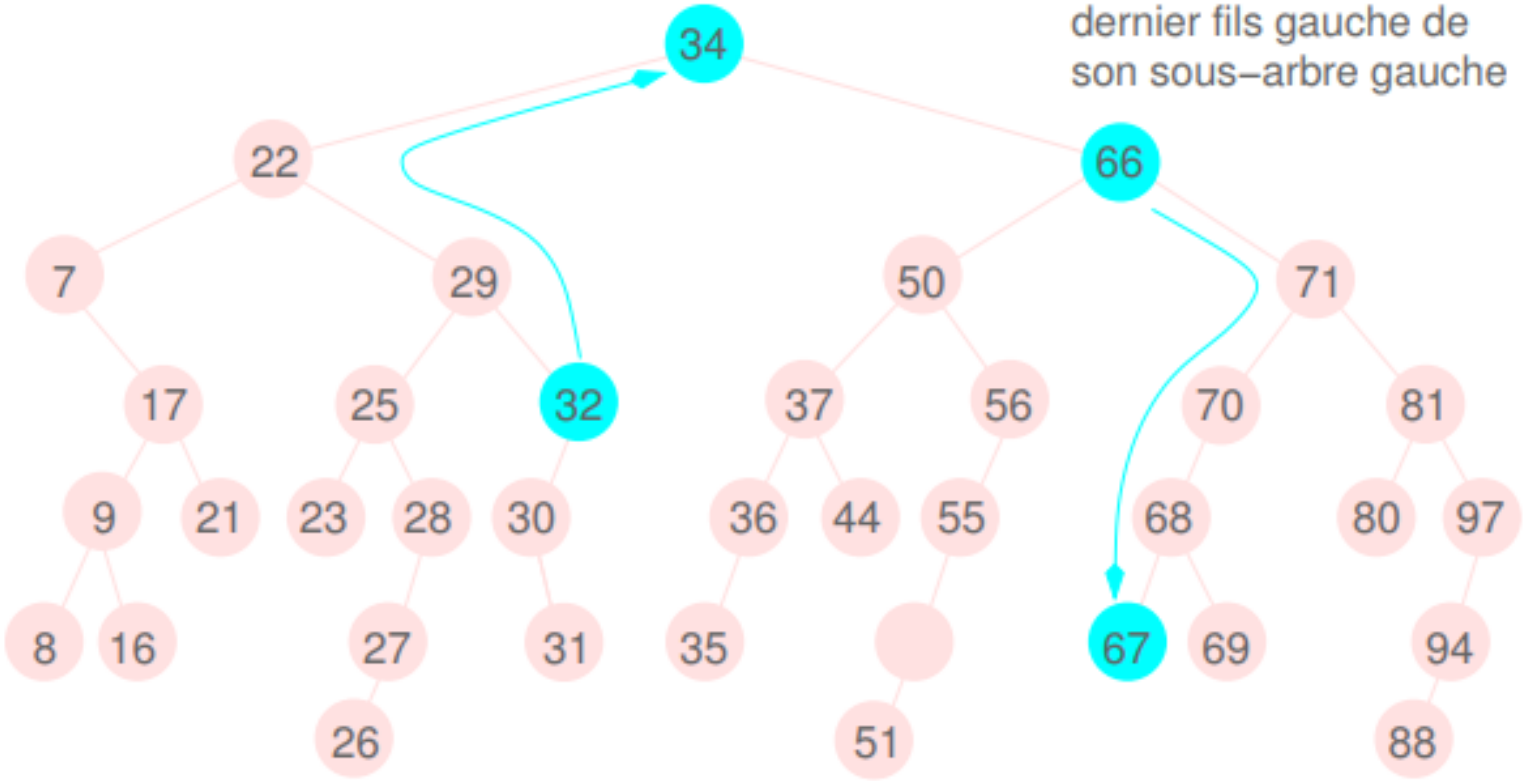
Recherche du successeur d'un nœud

Etant donné un nœud ***p*** d'un arbre A, le successeur de *p* si il existe, est le nœud de A qui porte comme valeur la plus petite des valeurs qui figurent dans A et qui sont plus grandes que la valeur de *p*. Si *p* possède un fils droit, son successeur est le nœud le plus à gauche dans son sous-arbre droit (on y accède en descendant sur le fils gauche autant que possible). Si *p* n'a pas de fils droit alors sont successeur est le premier de ses ascendants tel que *p* apparait dans son sous-arbre gauche. Si cet ascendant n'existe pas c'est que *p* portait la valeur la plus grande dans l'arbre.

Recherche du successeur d'un nœud

32 n'a pas de fils droit :
son successeur est 34, premier ascendant de 32
tel que 32 figure dans son sous–arbre gauche

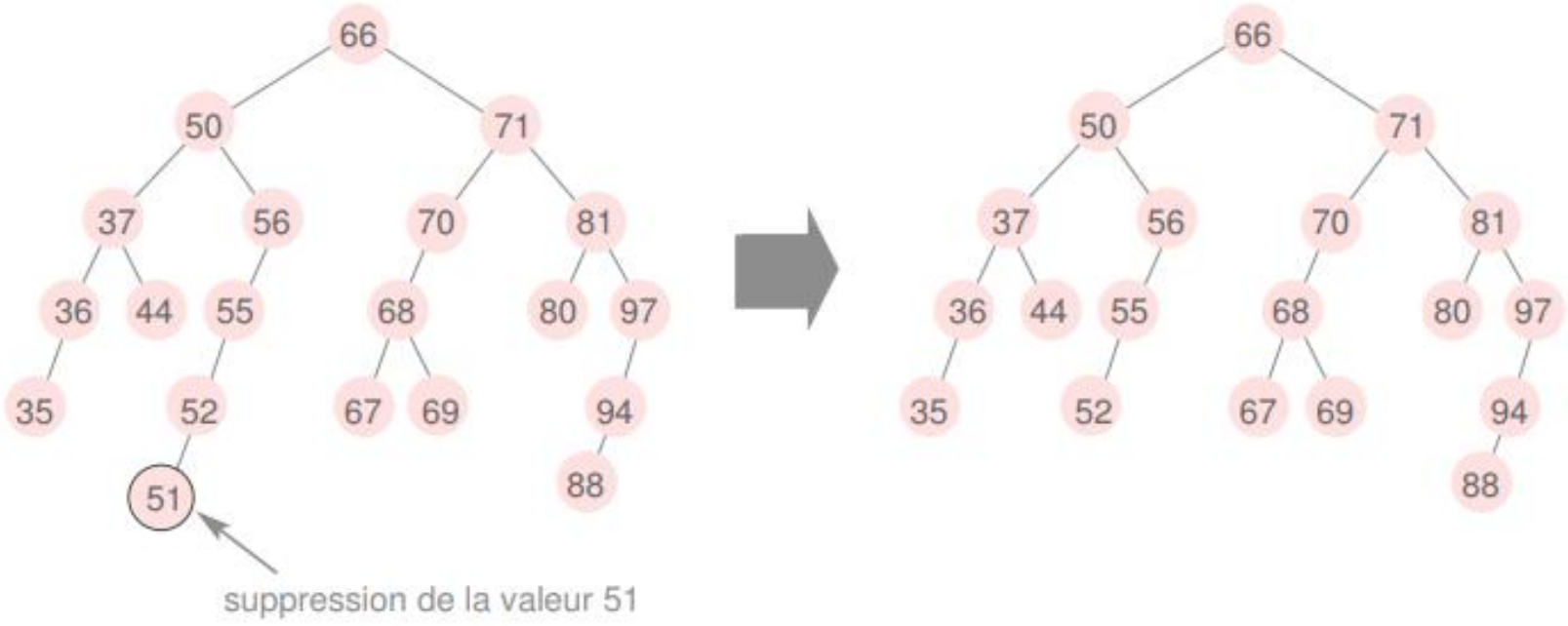
66 a un fils droit :
son successeur est 67
dernier fils gauche de
son sous–arbre gauche



Suppression d'un nœud

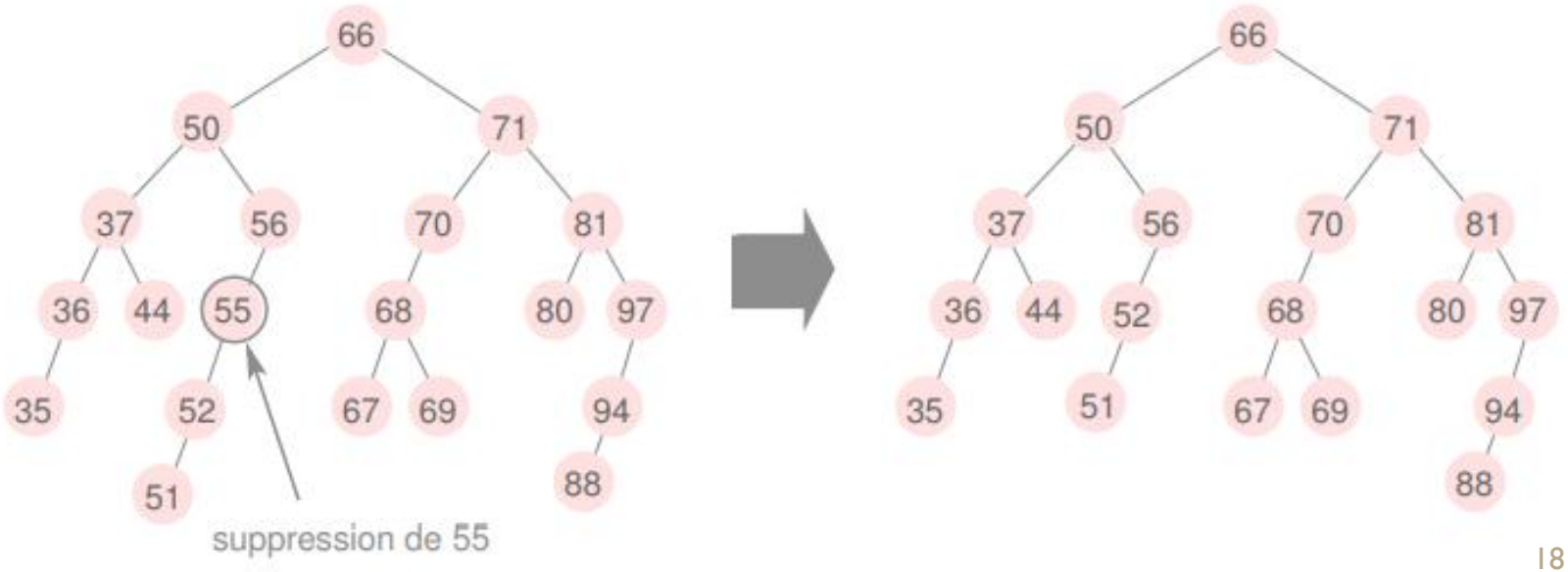
L'opération dépend du nombre de fils du nœud à supprimer.

Cas 1 : le nœud à supprimer n'a pas de fils, c'est une feuille. Il suffit de décrocher le nœud de l'arbre, c'est-à-dire de l'enlever en modifiant le lien du père, si il existe, vers ce fils. Si le père n'existe pas l'arbre devient l'arbre vide.



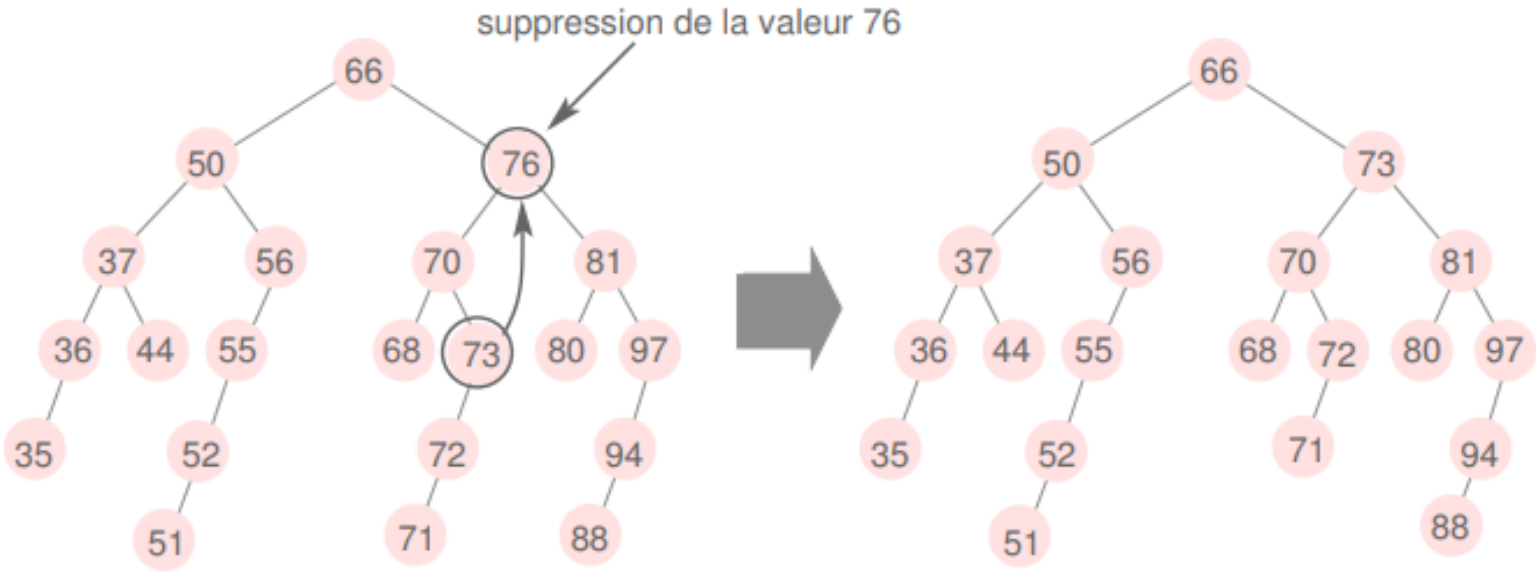
Suppression d'un nœud

Cas 2 : le nœud à supprimer a un fils et un seul. Le nœud est décroché de l'arbre comme dans le cas 1. Il est remplacé par son fils unique dans le nœud père, si ce père existe. Sinon l'arbre est réduit au fils unique du nœud supprimé.



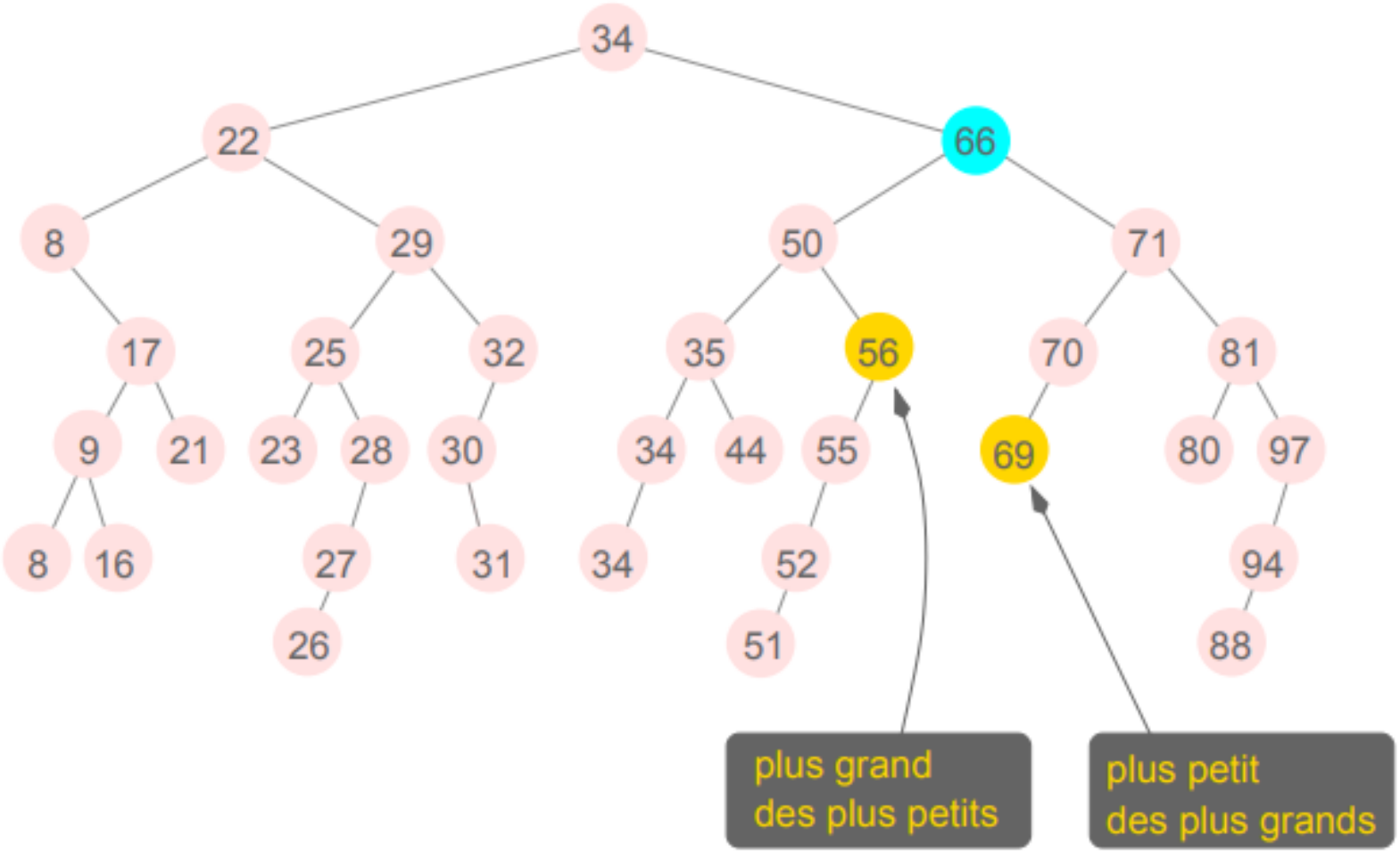
Suppression d'un nœud

Cas 3 : le nœud à supprimer ***p* a deux fils**. Soit *q* le nœud de son sous-arbre gauche qui a la valeur la plus grande (on peut prendre indifféremment le nœud de son sous-arbre droit de valeur la plus petite). Il suffit de recopier la valeur de *q* dans le nœud *p* et de décrocher le nœud *q*. Puisque le nœud *q* a la valeur la plus grande dans le fils gauche, il n'a donc pas de fils droit, et peut être décroché comme on l'a fait dans les cas 1 et 2.



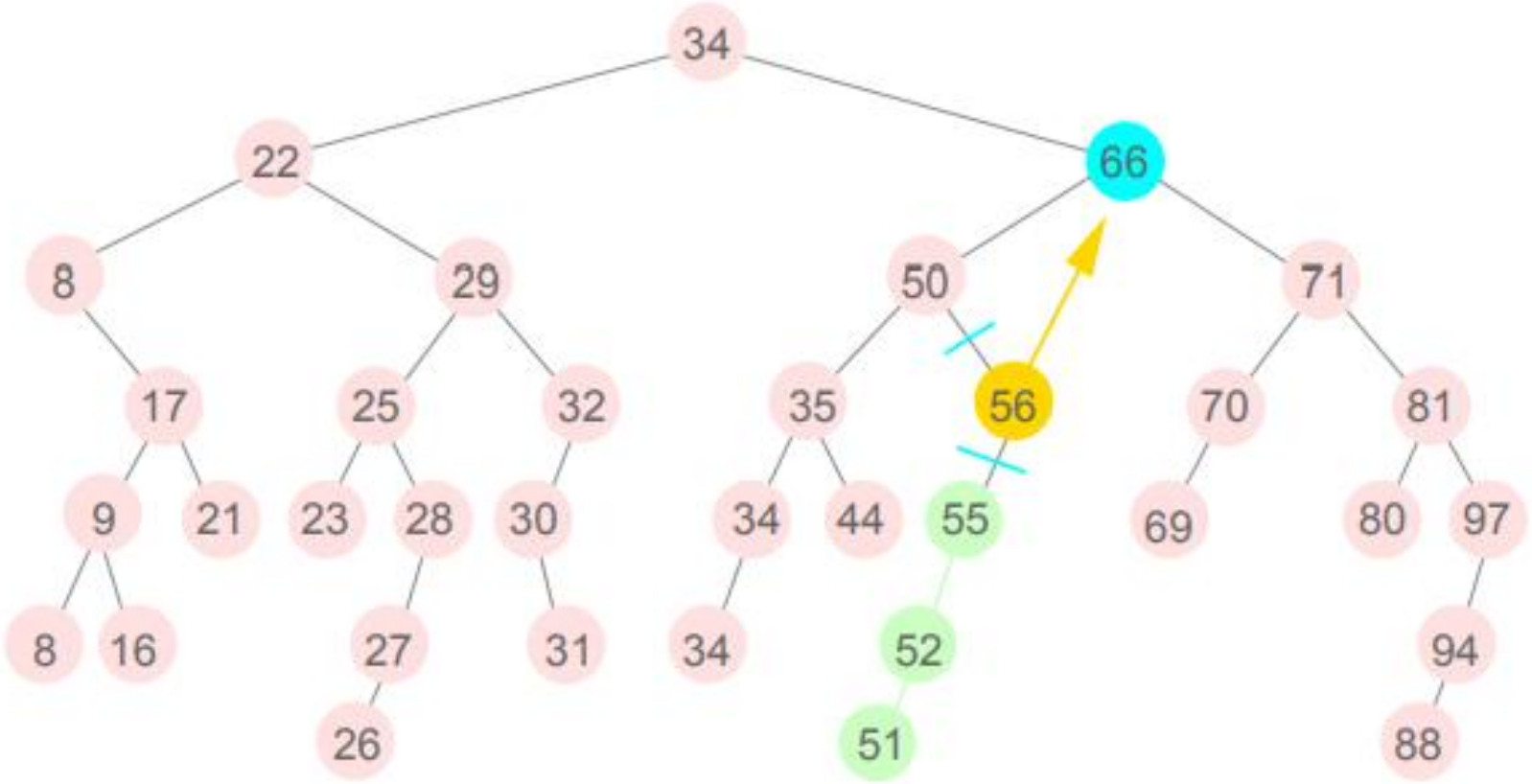
Suppression d'un nœud

Cas 3 : le nœud à supprimer ***p*** **a deux fils**



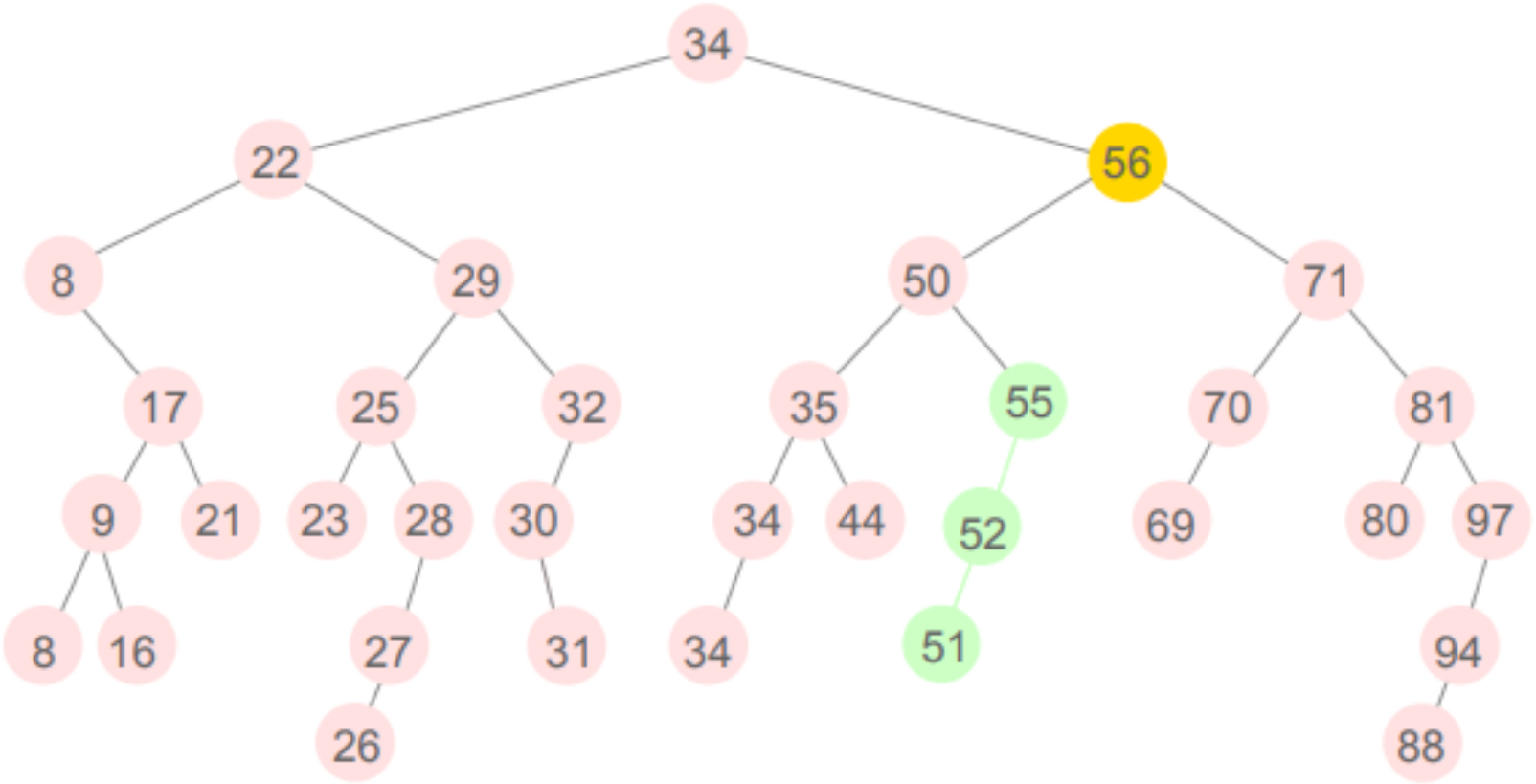
Suppression d'un nœud

Cas 3 : le nœud à supprimer ***p*** **a deux fils**



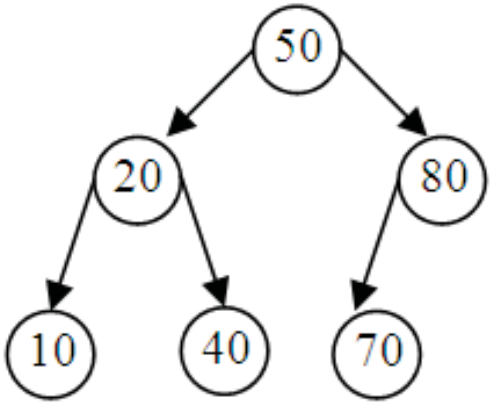
Suppression d'un nœud

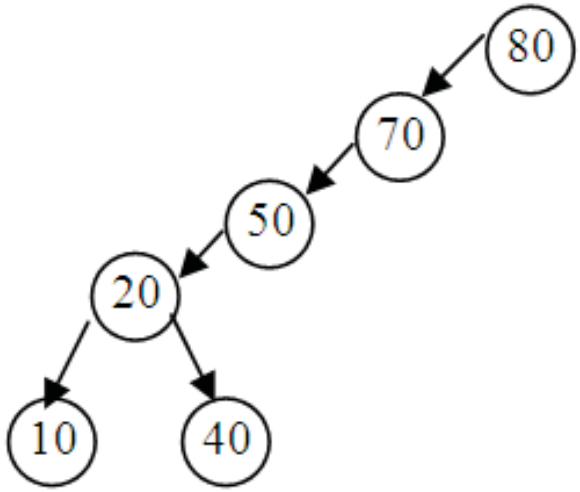
Cas 3 : le nœud à supprimer ***p*** **a deux fils**



ABR: Equilibrage

- Soit les deux ARB suivants :





L'opération d'équilibrage peut être faite à chaque fois qu'on insère un nouveau nœud où à chaque fois que le déséquilibre atteint un certain seuil pour éviter le coût de l'opération d'équilibrage qui nécessite une réorganisation de l'arbre.