Arbres binaires de recherche (ABR)
Binary Search Trees (BST)

I. Arbres binaires
   1. Structure
   2. Parcours
II. Arbres binaires de recherche
   1. Définition
   2. Opérations sur les ABR
      a) Recherche d'une clé
      b) Insertion d'un nœud
      c) Recherche d'un nœud successeur
      d) Suppression d'un nœud
         i. Cas1
         ii. Cas2
         iii. Cas3

Remerciements : illustrations sur les ABR d'après François Denis, Université d'Aix- Marseille

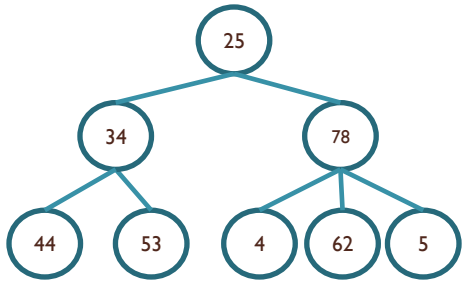
La structure de données

Rappel : *un graphe connexe possédant exactement (n-1) arcs est un arbre.*

Les arbres binaires (AB) forment une structure de données qui se définit récursivement. Un arbre binaire est :

- soit vide,
- soit composé d'une racine portant une étiquette (= clé) et d'une paire d'arbres binaires, appelés fils gauche et fils droit.

L'arbre ci-contre n'est donc pas de type binaire

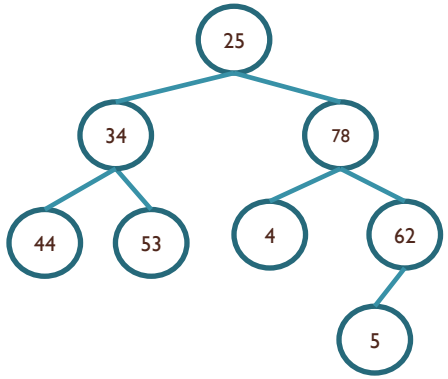


Pour pouvoir manipuler des arbres de recherche, on doit donc disposer au minimum des fonctionnalités suivantes :

– **AB : cree arbre vide ( )** // crée un arbre vide
– **AB : cree arbre (v, fg, fd)** // crée un arbre dont les fils gauche et // droit sont fg et fd et dont la racine //est étiquetée par v
– **booleen : est_vide (AB)** // teste si un arbre est vide ou non

– **AB : fils gauche(AB, a)** et **AB : fils droit(AB, a)** // retournent les fils gauche // et droit d'un arbre (AB) non vide
– **CLE : val(AB, a)** // retourne l'étiquette de la racine // d'un arbre non vide,

On peut (doit ?) disposer aussi de fonctions permettant l'insertion et la délétion de nœuds.



Arbre binaire

Parcours d'un arbre binaire.

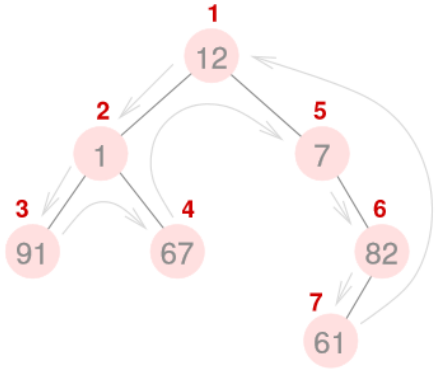
Le parcours le plus simple à programmer est le parcours en profondeur d'abord (DFS). Son principe est simple : pour parcourir un arbre non vide A, on parcourt récursivement :

1. Son sous-arbre gauche,
2. puis son sous-arbre droit,

la racine de l'arbre pouvant être traitée au **début**, **entre** les deux parcours ou à la **fin**.

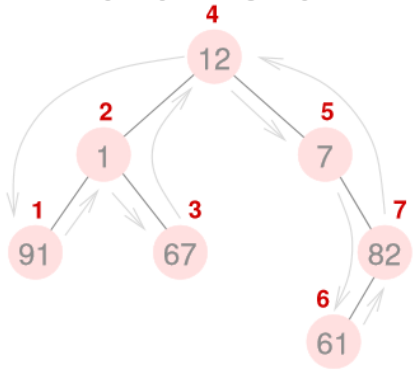
Dans le premier cas, les nœuds sont traités dans un **ordre préfixé** , dans le second cas, dans un **ordre infixé** et dans le troisième cas, selon un ordre **postfixé**.

```
Affichage_préfixe(AB a)
   si NON est_vide(a)
      Afficher val(a)
      Affichage_préfixe(fils_gauche(a))
      Affichage_préfixe(fils_droit(a))
```



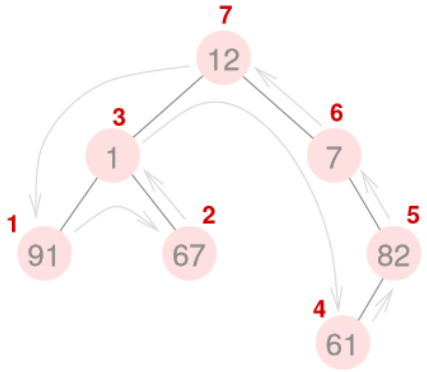
12 1 91 67 7 82 61

```
Affichage_infixe(AB a)
   si NON est_vide(a)
      Affichage_infixe(fils_gauche(a))
      Afficher val(a)
      Affichage_infixe(fils_droit(a))
```



91 1 67 12 7 61 82

```
Affichage_postfixe(AB a)
   si NON est_vide(a)
      Affichage_postfixe(fils_gauche(a))
      Affichage_postfixe(fils_droit(a))
      Afficher val(a)
```



91 67 1 61 82 7 12

II. Arbre Binaire de Recherche

1. Définition

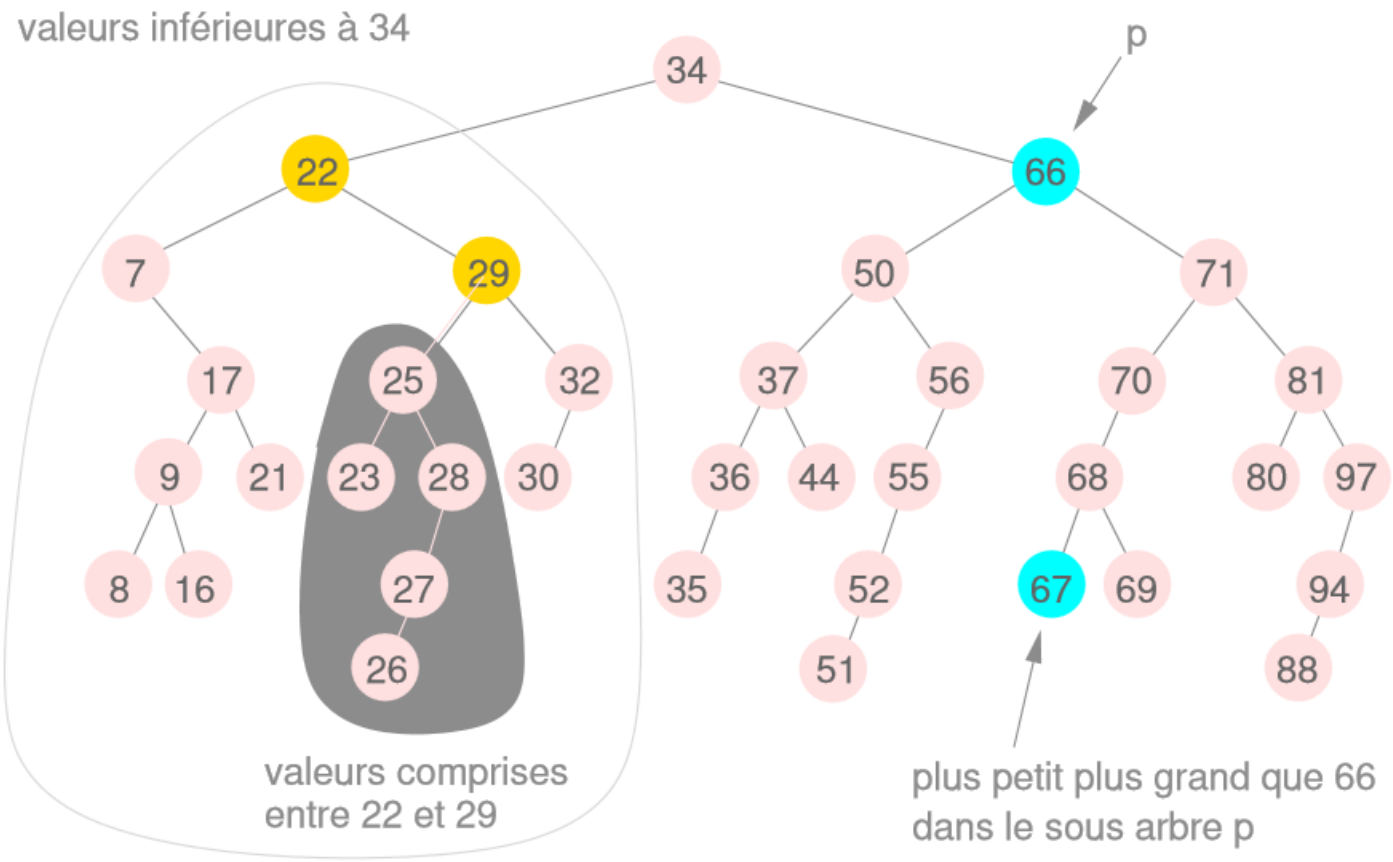
Un arbre binaire de recherche (ou ABR) est une structure de donnée qui permet de représenter un ensemble de valeurs sur lesquelles on dispose d'une relation d'ordre.
En pratique, les valeurs sont des clés (= étiquettes) permettant d'accéder à des enregistrements.

Soit E un ensemble muni d'une relation d'ordre, et soit A un arbre binaire portant des valeurs de E.

L'arbre A est un arbre binaire de recherche si pour tout nœud p de A, la valeur de p est strictement plus grande que les valeurs figurant dans son sous-arbre gauche et strictement plus petite que les valeurs figurant dans son sous-arbre droit.

Cette définition suppose donc qu'une valeur n'apparaît au plus qu'une seule fois dans un arbre de recherche.

I. Définition : Exemple

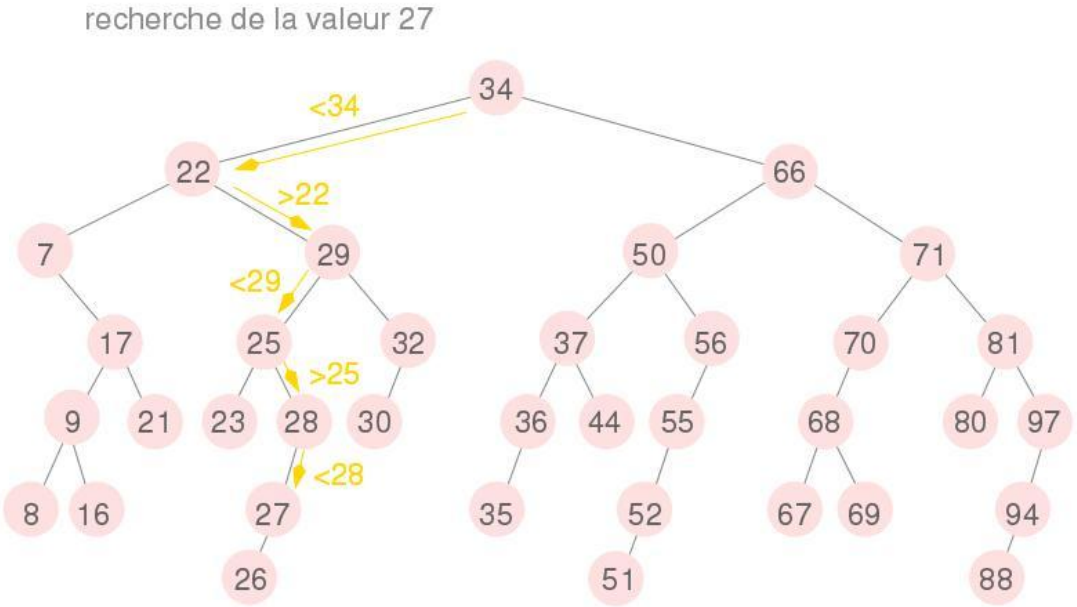


2. Opérations sur les ABR

a) Recherche d'une valeur de clé

- La recherche d'une valeur dans un ABR consiste à parcourir une branche en partant de la racine, en descendant chaque fois à gauche ou à droite suivant que la clé portée par le nœud est plus grande ou plus petite que la valeur cherchée.

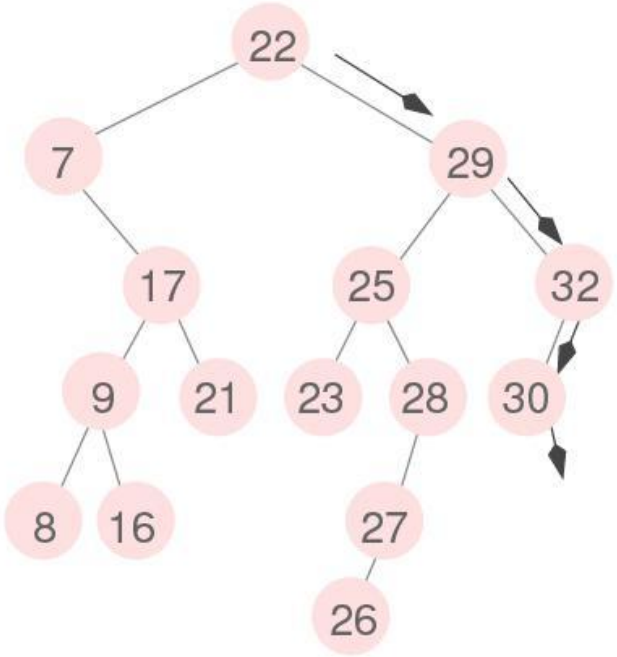
La recherche s'arrête dès que la valeur est rencontrée ou que l'on a atteint une feuille, c'est-à-dire que le fils sur lequel il aurait fallu descendre n'existe pas (la valeur recherchée est absente).

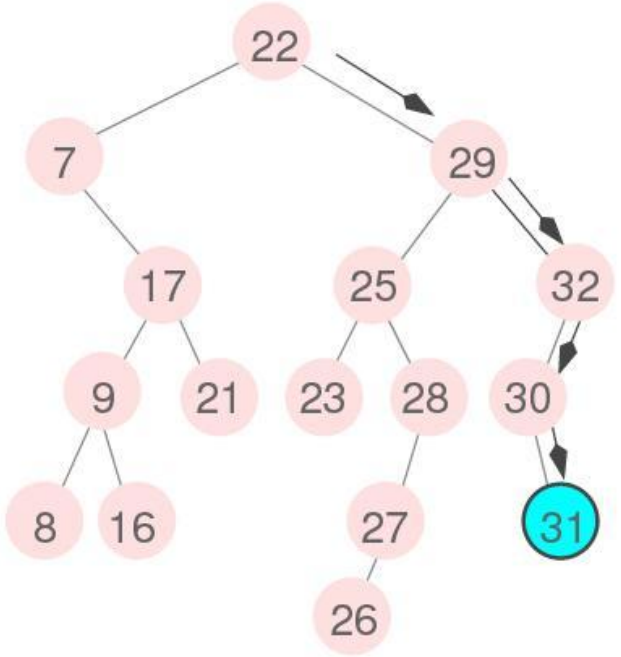


2. Opérations sur les ABR

b) Insertion d'une valeur.

Le principe est le même que pour la recherche. Un nouveau nœud est créé avec la nouvelle valeur, insérée a l'endroit où la recherche s'est arrêtée en tant que fils droit ou gauche de la feuille trouvée.





2. Opérations sur les ABR

Algorithme d'insertion d'un nœud
En entrée = un arbre, une clé à insérer
En sortie = l'arbre modifié

```
ABR = Inserer (ABR, cle)
{
r = racine(A);
 SI r est vide ALORS
    ABR = Construire l'arbre avec (cle, fg =vide, fd = vide);
SINON
      SI (cle < r) ALORS
            ABR= Inserer (A, fg = cle);
     SINON
            ABR= Inserer (A, fd = cle);
FINSI
 }
```

2. Opérations sur les ABR

c) Recherche du successeur d'un nœud.

- Etant donné un nœud *p* d'un arbre *A*, le successeur de *p* s'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* apparaît dans son sous-arbre gauche. Si cet ascendant n'existe pas c'est que *p* portait la valeur la plus grande de *A*.

Remarques

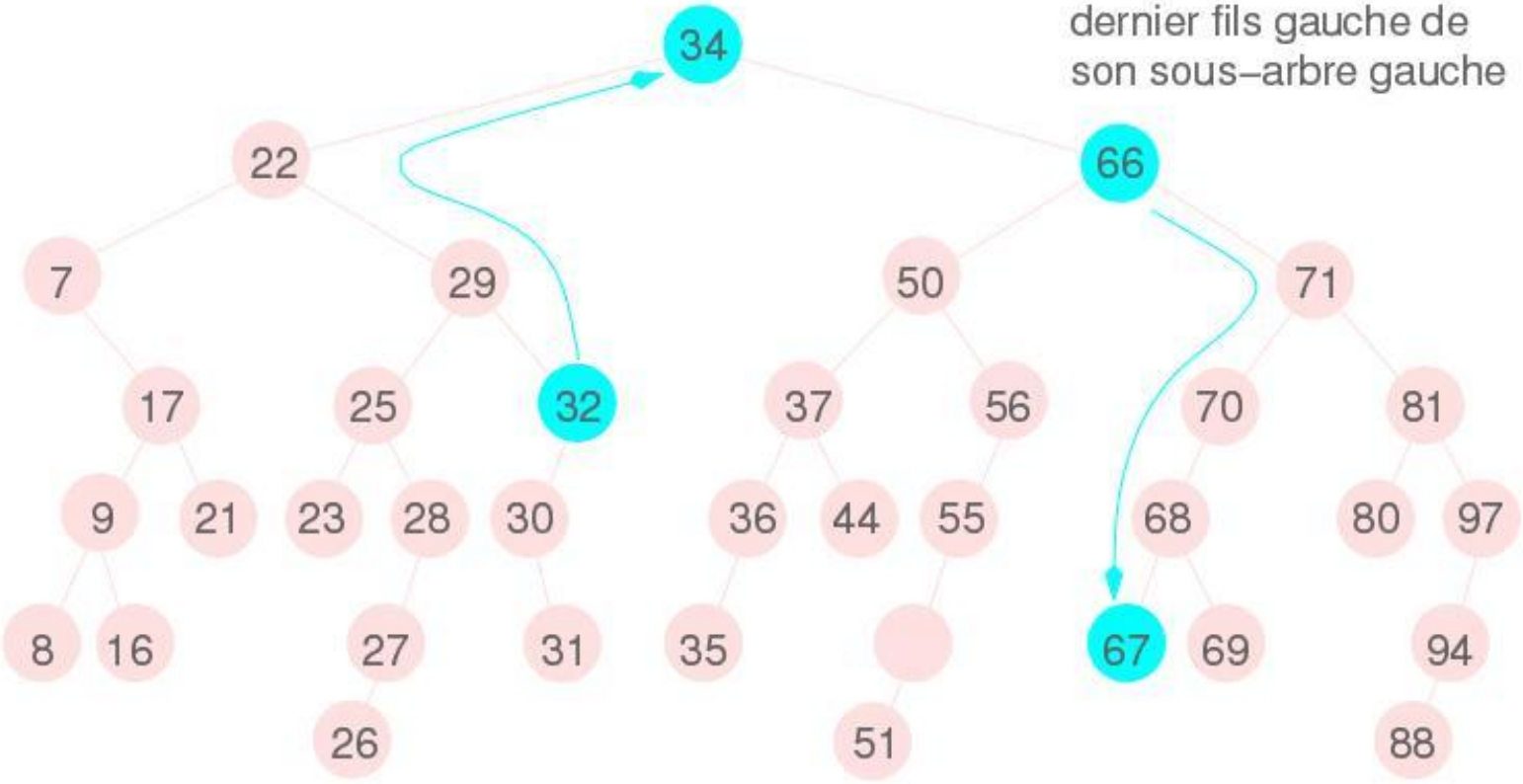
1. *Il est nécessaire d'avoir pour chaque nœud un lien vers son père pour mener à bien cette opération.*
2. *Une édition de la liste ordonnée (parcours infixe) donne le successeur de n'importe quel nœud. Encore faut-il que l'arbre ne soit pas trop volumineux !*

3. Opérations sur les ABR

Exemple de recherche d'un successeur

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

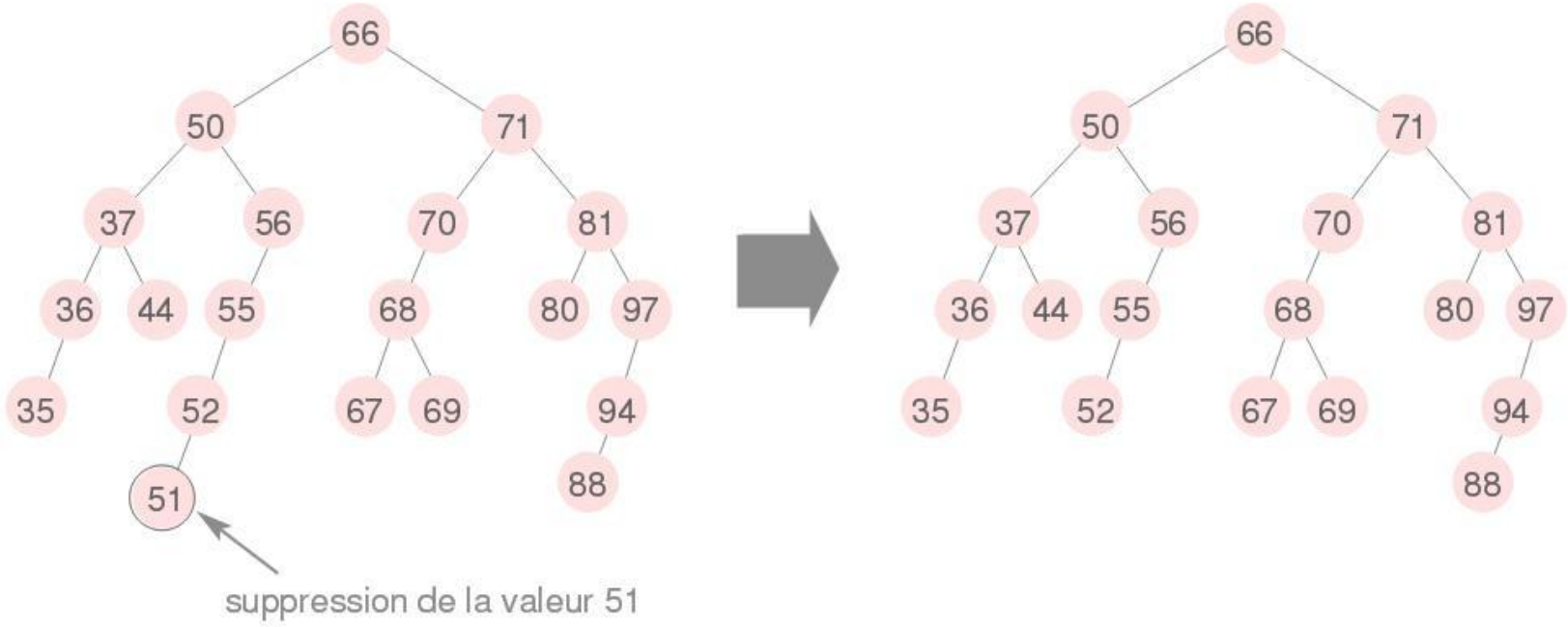


3. Opérations sur les ABR

d) Suppression d'un nœud.

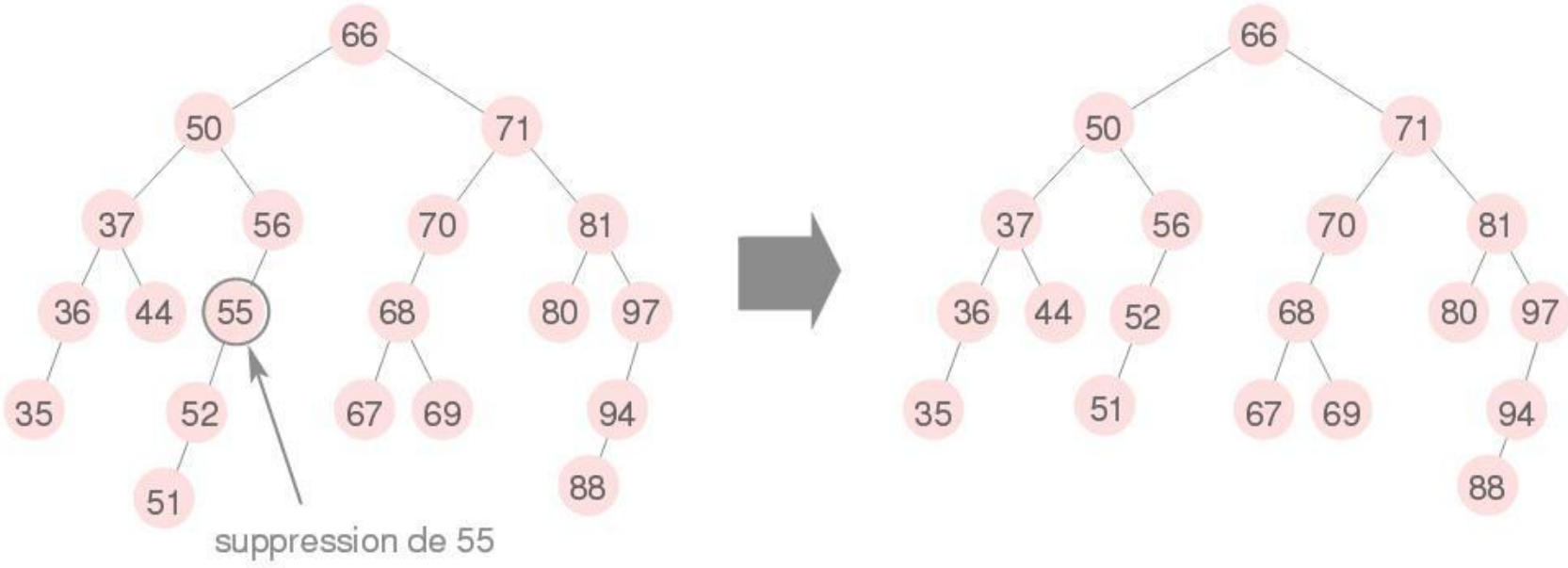
L'opération dépend du nombre de fils du nœud à supprimer. Trois cas à considérer.

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.



3. Opérations sur les ABR

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.



3. Opérations sur les ABR

Cas 3. le nœud à supprimer p a deux fils (cas général). 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.