Chapitre II: Arbre binaire de recherche

Chapitre II: Arbre binaire de recherche

problématique

- La plupart des bons algorithmes fonctionnent grâce à une méthode astucieuse pour organiser les données.
- Avoir une structure de donnée où l'insertion et la recherche sont efficace .
- l'utilisation des arbres en informatique est une solution tés performante parce que :

➢ Les informations à stockées sont souvent hiérarchisées, et peuvent être représentées naturellement sous une forme arborescente,

➢ Les structures de données arborescentes permettent de stocker des données volumineuses de façon que leur accès soit efficace

Chapitre II: Arbre binaire de recherche

Définitions et terminologie

- Un arbre est un ensemble organisé de nœuds dans lequel chaque nœud a un père et un seul, sauf un nœud que l'on appelle la ***racine***.
- Un nœud peut être :
  - ***Racine :*** le seule nœud qui n'a pas de père
  - ***Nœud intérieur :*** un nœud qui a un père et au moins un fils
  - ***Feuille :*** un nœud qui a un père et qui n'a aucun ils

- Un nœud est défini par son étiquette et ses sous-arbres

Chapitre II: Arbre binaire de recherche

les Arbres en théorie de graphe

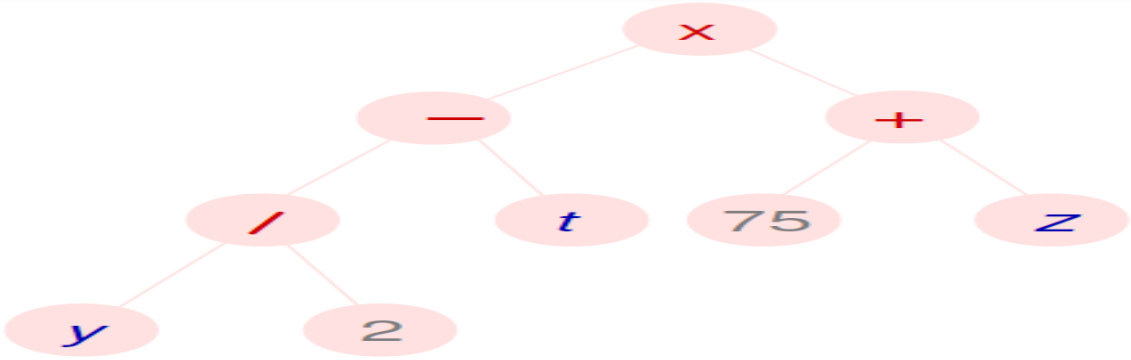
- Un arbre est un graphe connexe et sans cycles : entre deux sommets quelconques du graphe il existe un chemin et un seul.
- On parle dans ce cas d'arbre non planté, puisqu'il n'y a pas de sommet particulier qui joue le rôle de racine.
- Les sommets sont voisins les uns des autres, il n'y a pas de hiérarchie qui permet de dire qu'un sommet est le père (ou le fils) d'un autre sommet.

Chapitre II: Arbre binaire de recherche

Exemples

Arbre d'expression arithmétiques

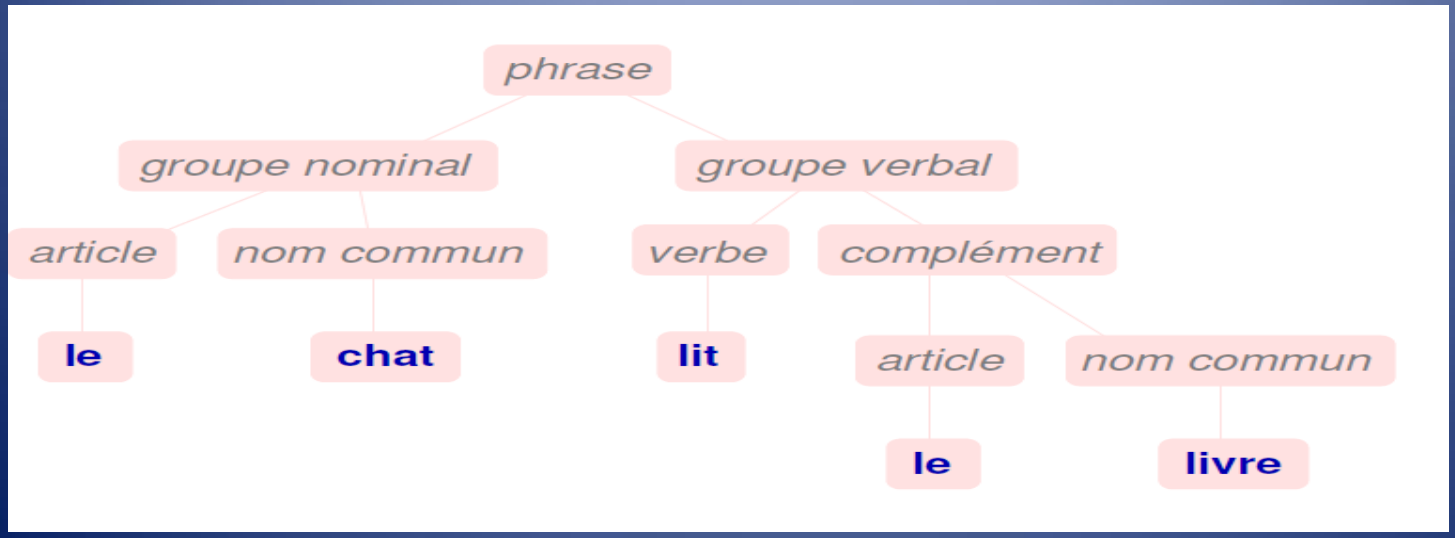
On peut représenter les expressions arithmétiques par des arbres étiquètes par des operateurs, des constantes et des variables. La structure de l'arbre rend compte de la priorité de operateurs et rend inutile tout parenthèse .



Chapitre II: Arbre binaire de recherche

Arbre syntaxique

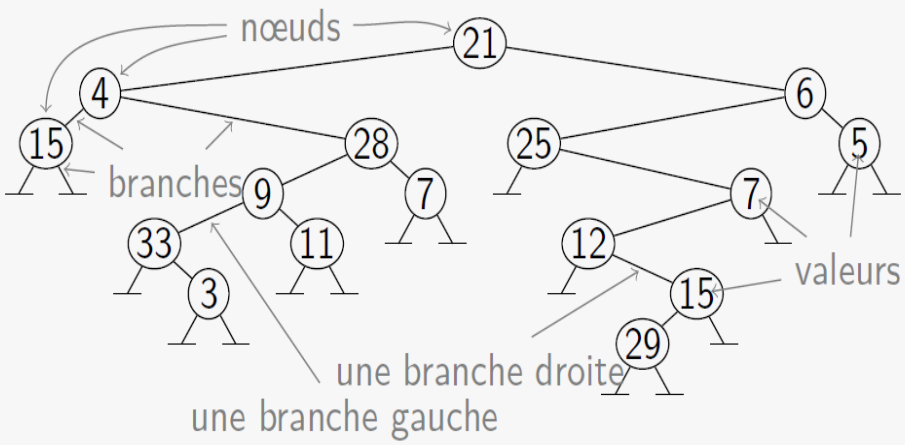
Un arbre syntaxique représente l'analyse d'une phrase à partir d'un ensemble de règles qui constitue la grammaire :
une phrase est composée d'un groupe nominal suivi d'un groupe verbal, un groupe nominal peut-être constitué d'un article et d'un nom commun,. . .



Chapitre II: Arbre binaire de recherche

Arbre binaire

- Un arbre est un ensemble organise de nœuds dans lequel chaque nœud a un père et un seul, sauf un nœud que l'on appelle la racine.

Représentation graphique d'un arbre binaire	Vocabulaires
	❑ Arbre, nœuds, branches; ❑ Arbre binaire, branches gauches, branches droites; ❑ Valeurs (ou étiquettes) des nœuds.

Chapitre II: Arbre binaire de recherche

Arbre binaire (Définition récursive)

- Les arbres binaires (AB) forment une structure de données qui peut être définie récursivement de la manière suivante :
- 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 droit.

Chapitre II: Arbre binaire de recherche

Manipulation d'un arbre binaire

Pour pouvoir manipuler des arbres de recherche, on doit disposer des fonctionnalités suivantes

- ***booléen : est vide(AB a)*** : une fonction qui teste si un arbre est vide ou non,
- AB : fils gauche(AB a) et AB : fils droit(AB a), des fonctions qui retournent les fils gauche et droit d'un arbre non vide,
- ***CLE : val(AB a),*** une fonction qui retourne l'´etiquette de la racine d'un arbre non vide,

Chapitre II: Arbre binaire de recherche

Manipulation d'un arbre binaire (suite)

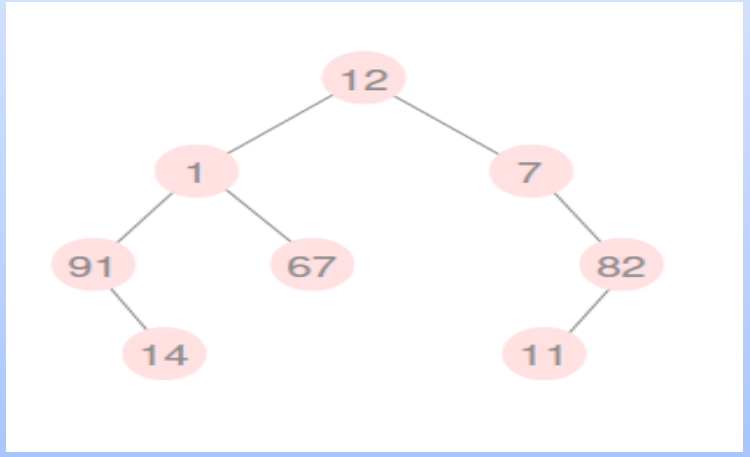
- ***AB cree arbre vide(),*** une fonction qui crée un arbre vide,
- ***AB cree arbre(CLE v, AB fg, fd),*** une fonction qui crée un arbre dont les fils gauche et droit sont fg et fd et dont la racine est étiquetée par v
- ***change val(CLE v, AB a),*** une fonction qui remplace l'´etiquette de la racine de a par v, si a est non vide, et ne fait rien sinon
- ***change fg(AB fg, AB a) et change fd(AB fd, AB a),*** des fonctions qui remplacent le fils gauche de a (resp. le fils droit de a) par fg (resp. par fd), si a est non vide, et ne font rien sinon

Chapitre II: Arbre binaire de recherche

Représentation des arbres binaire

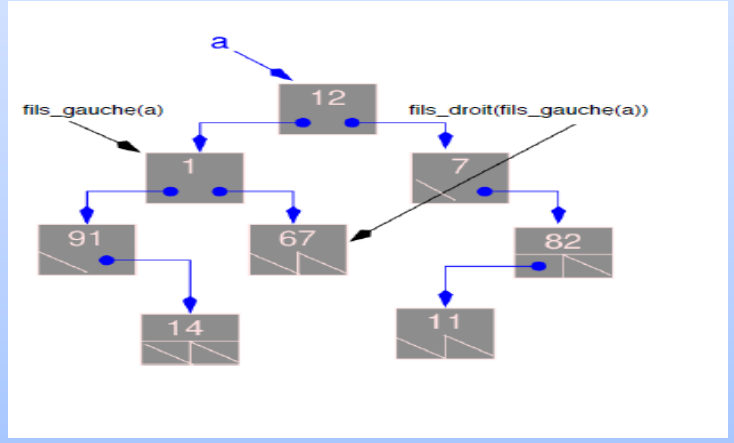
En C

Utilise en général des pointeurs pour représenter les liens entre un nœud et ses Fils dans un arbre binaire.



En Python

Représente un arbre vide par une liste vide [] et un arbre non vide par une liste comprenant 3 éléments [clé ,fils gauche ,fils droit].

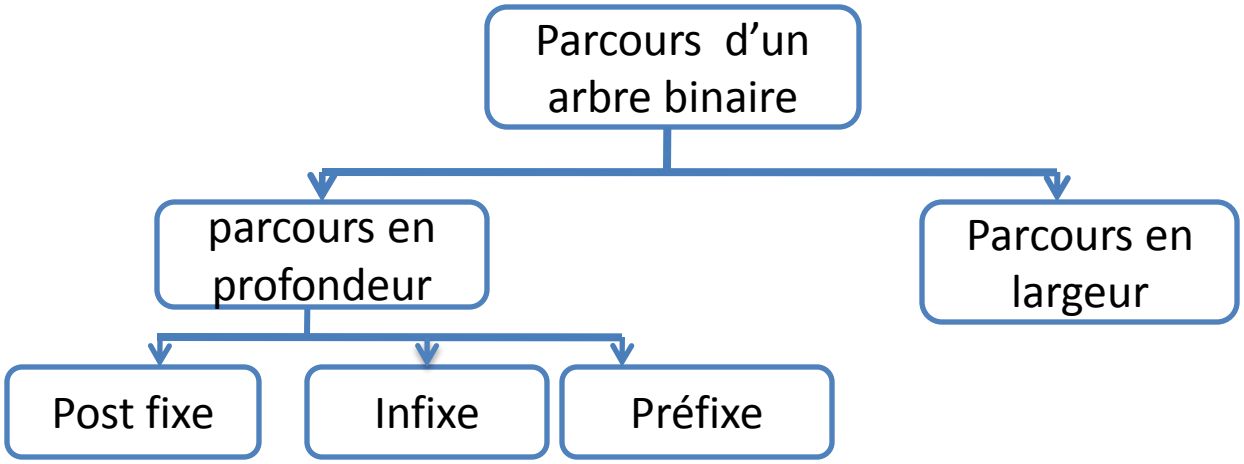


Chapitre II: Arbre binaire de recherche

Parcours d'un arbre binaire

Un parcours d'arbre est une façon d'ordonner les nœuds d'un arbre afin dles parcourir.

Types de parcours d'un arbre binaire



Chapitre II: Arbre binaire de recherche

Parcours en profondeur d'un arbre binaire

- Les parcours en profondeur se définissent de manière récursive sur les arbres.
- Le parcours d'un arbre consiste à traiter la racine de l'arbre et à parcourir récursivement les sous-arbres gauche et droit de la racine.
- Les parcours préfixe, infixe et suffixe se distinguent par l'ordre dans lequel sont faits ces traitements.

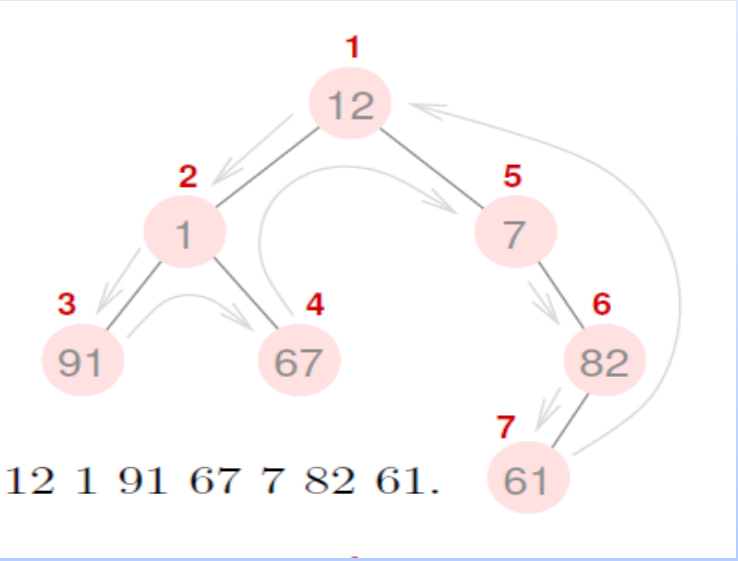
```
Parcours(AB a)
    si NON est_vide(a)
        Traitement_prefixe(val(a))
        Parcours(fils_gauche(a))
        Traitement_infixe(val(a))
        Parcours(fils_droit(a))
        Traitement_postfixe(val(a))
```

Chapitre II: Arbre binaire de recherche

Parcours en profondeur d'un arbre binaire (préfixe)

Affichage des en préfixe :

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

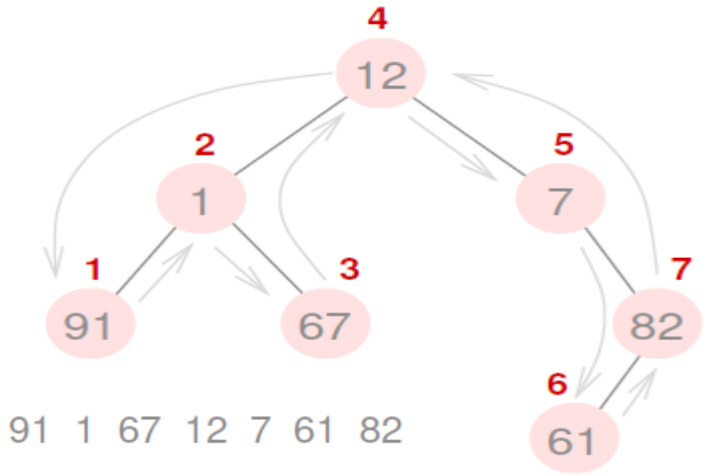


Chapitre II: Arbre binaire de recherche

Parcours en profondeur d'un arbre binaire(Infixe)

Affichage en infixe :

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

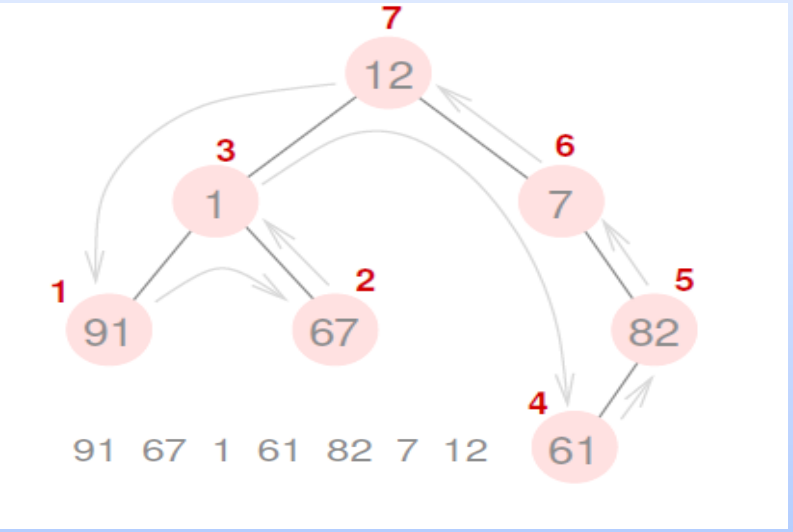


Chapitre II: Arbre binaire de recherche

Parcours en profondeur d'un arbre binaire (poste fixe)

Affichage des en post fixe :

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



Chapitre II: Arbre binaire de recherche

Arbres binaires de recherche

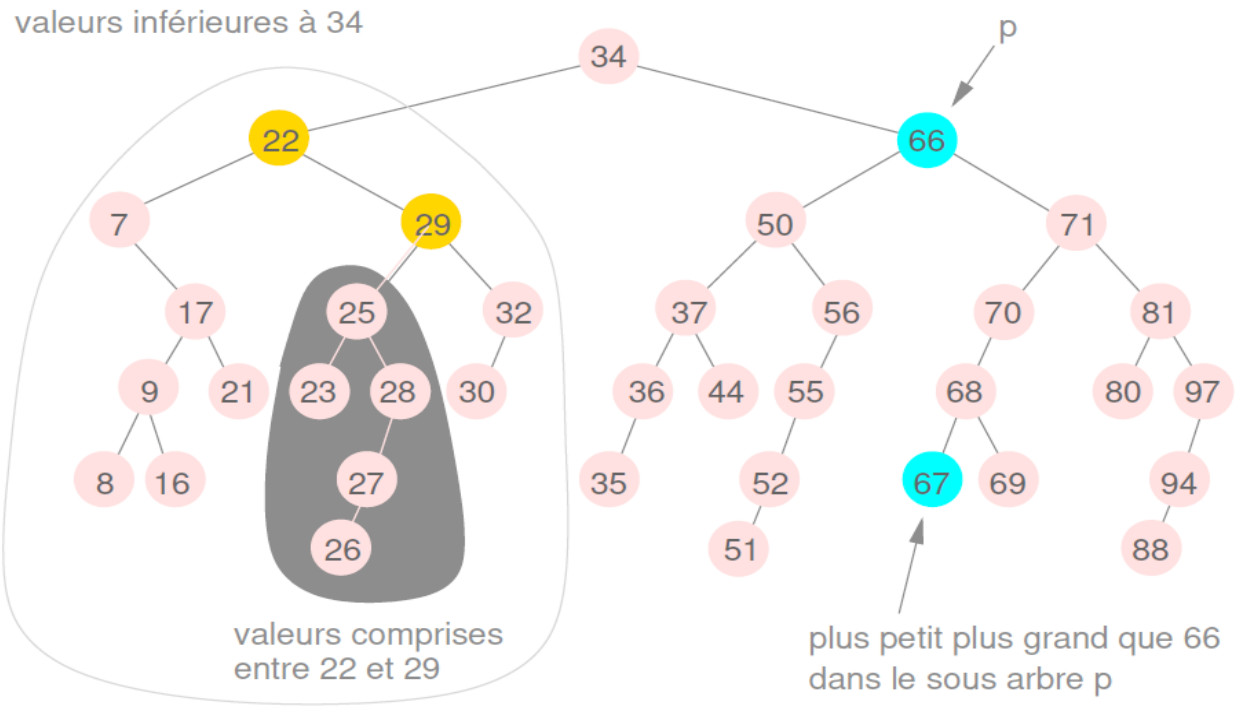
- Un arbre binaire de recherche (ou ABR) est une structure de donnée qui permet de représenter un ensemble de valeurs si l'on dispose d'une relation d'ordre sur ces valeurs.

- 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'apparaıt au plus qu'une seule fois dans un arbre de recherche.

Chapitre II: Arbre binaire de recherche

Arbres binaires de recherche (Exemple)



Chapitre II: Arbre binaire de recherche

Opération sur les ABR

Les opérations caractéristiques sur les arbres binaires de recherche sont :

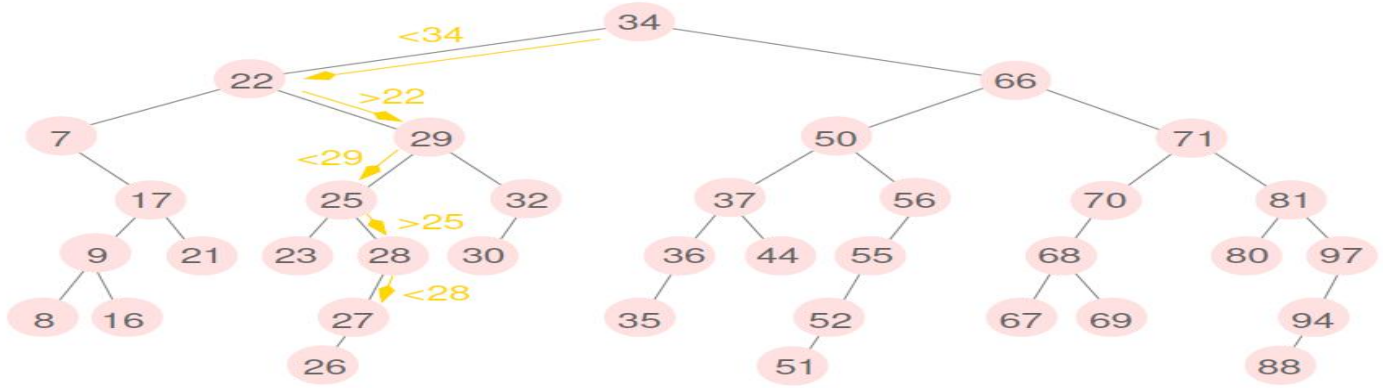
- ***L'insertion,***
- ***La suppression,***
- ***La recherche d'une valeur. .***

Chapitre II: Arbre binaire de recherche

Recherche d'une valeur

- Consiste à parcourir une branche en partant de la racine, en descendant chaque fois sur le fils gauche ou sur le fils droit suivant que la clé portée par le nœud est plus grande ou plus petite que la valeur cherchée.
- La recherche s'arrete dés que la valeur est rencontrée ou que l'on a atteint l'extrémité d'une branche (le fils sur lequel il aurait fallu descendre n'existe pas).

Exemple recherche de la valeur 27



Chapitre II: Arbre binaire de recherche

Recherche d'une valeur (Algorithme)

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* $v \neq val(a)$ **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
    retourner Vrai
  finsi
fin

Chapitre II: Arbre binaire de recherche

Recherche d'une valeur (Algorithme)

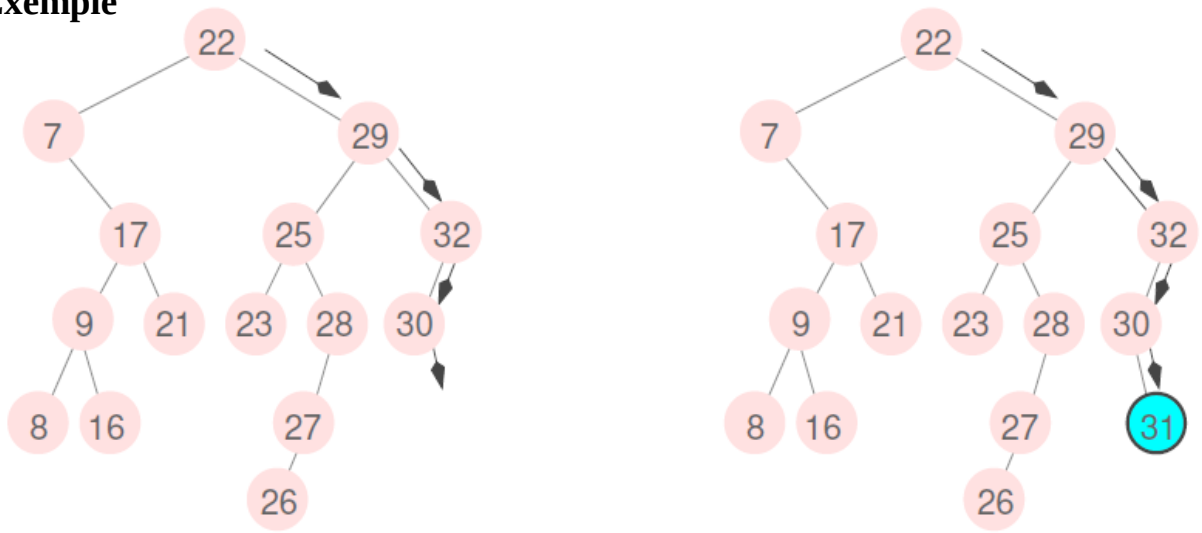
```
Fonction recherche(a, v) : version récursive
  entrée : a est un ABR, v est une clé.
  sortie : Vrai si v figure dans a et Faux sinon.
  début
      si est_vide(a) alors
          retourner Faux
      sinon
          si v == val(a) alors
              retourner Vrai
          sinon
             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

Chapitre II: Arbre binaire de recherche

Insertion d'une valeur

- Le principe est le même que pour la recherche
- Un nouveau nœud est crée avec la nouvelle valeur et insère à l'endroit ou` la recherche s'est arrêtée

Exemple



Chapitre II: Arbre binaire de recherche

Insertion d'une valeur (Algorithme)

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

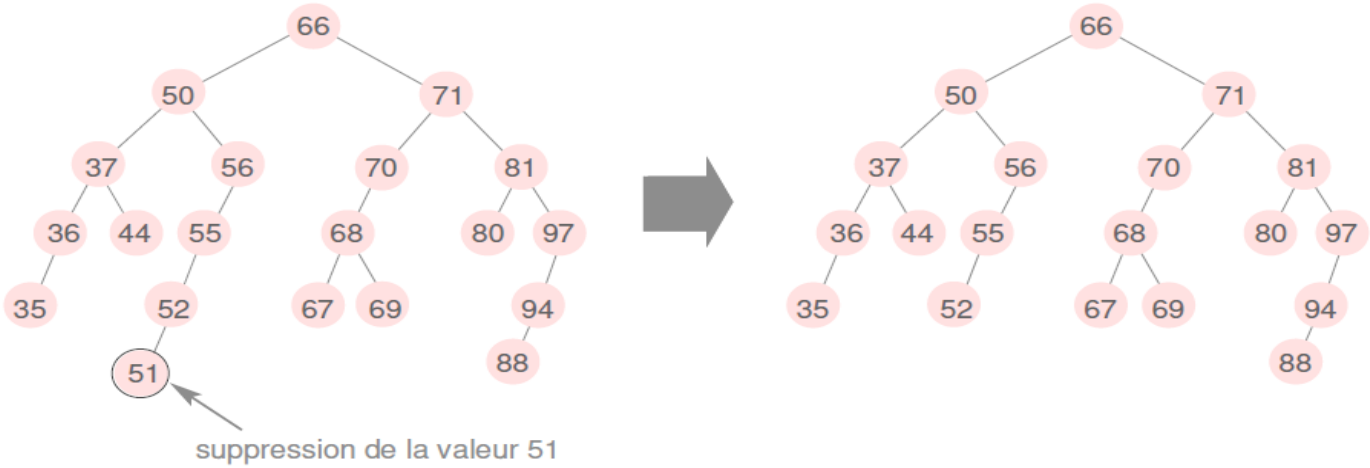
Chapitre II: Arbre binaire de recherche

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-`a-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.

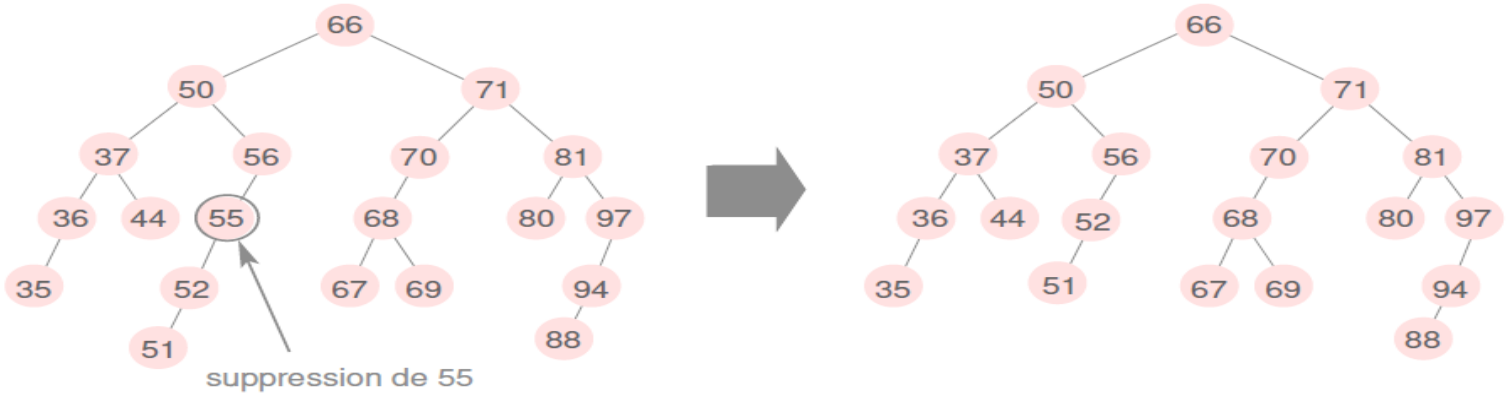


Chapitre II: Arbre binaire de recherche

Suppression d'un nœud

cas 2

le nœud a supprimer a un fils et un seul. Le nœud est décroche 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.



Chapitre II: Arbre binaire de recherche

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 suffi 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