Arbres Binaires de Recherche

Un ***arbre binaire de recherche*** ou **ABR** (en anglais, ***binary search tree*** ou ***BST***) 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.

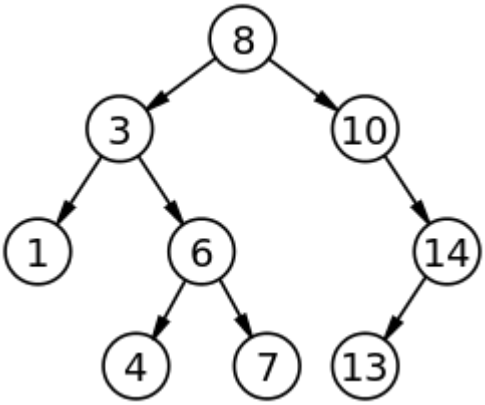
Un arbre binaire est un arbre binaire de recherche (ABR) est un arbre binaire composé de noeuds dont toutes les valeurs peuvent être comparées* et qui sont organisés de la manière suivante

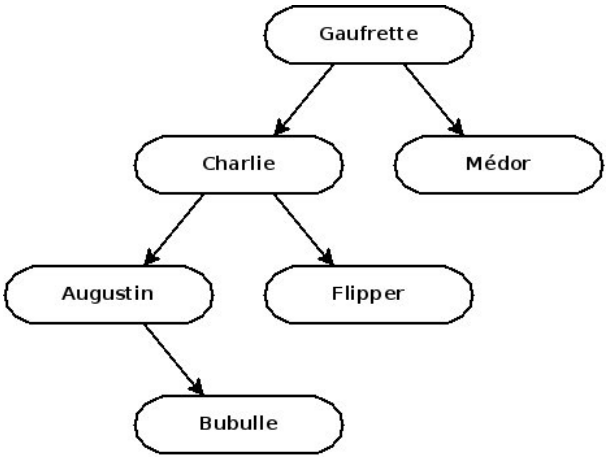
- la clé (ou étiquette) du nœud est strictement plus grande que les clés figurant dans son sous-arbre gauche
- la clé du nœud est strictement plus petite que les clés figurant dans son sous-arbre droit.

Remarque : on utilisera les mots inférieur et supérieur dans le sens de la comparaison de 2 valeurs avec un ordre. Cet ordre peut être l'ordre des nombres mais aussi l'ordre alphabétique ou un autre ordre prédé#ni.

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

Exemples d'arbre binaire de recherche :





Remarque : **La structure d'un arbre binaire de recherche est dépendante de l'ordre dans lequel on insère les éléments** !

Exercice 1 :

A partir d'un arbre vide, construire un arbre binaire de recherche en insérant successivement les valeurs suivantes :
7, 3, 4, 10, 2, 6, 14, 5

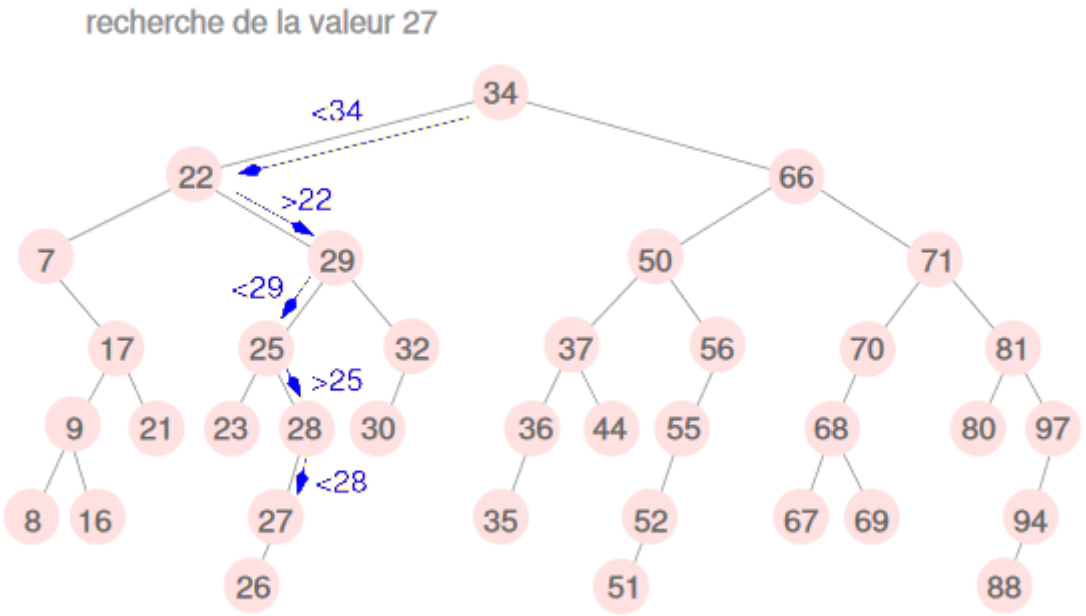
Dans un ABR, où est situé la clé la plus grande ? la clé la plus petite ?

Proposer le *parcours infixe* (en profondeur) de l'arbre binaire que vous venez de construire. Que remarque-t-on ?

*Les opérations caractéristiques sur les arbres binaires de recherche sont **l'insertion**, la **suppression**, et la **recherche** d'une valeur. Ces opérations sont peu coûteuses si l'arbre n'est pas trop déséquilibré.*

Recherche d'une clé dans un arbre binaire de recherche :

La recherche d'une valeur dans un ABR 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'arrête dès 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*).



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

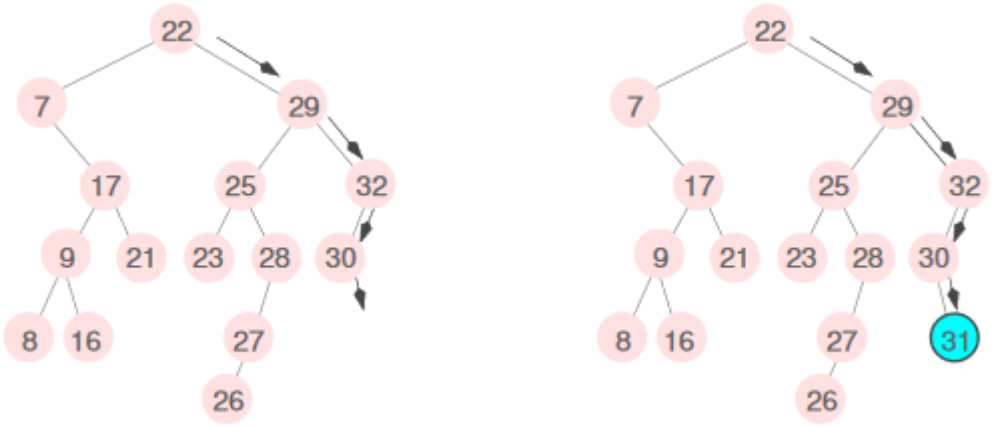
Exercice 2 :

Implémenter cet algorithme en python

Insertion d'une clé dans un arbre binaire de recherche :

Le principe est le même que pour la recherche. Un nouveau nœud est créé avec la nouvelle valeur et inséré à l'endroit où la recherche s'est arrêtée.

Ex : insertion de la valeur 31 dans l'arbre binaire de recherche suivant :



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
        renvoyer cree_arbre(v,cree_arbre_vide(),cree_arbre_vide())
    sinon
        si v < val(a) alors
            renvoyer [ a, Insertion(v,fils_gauche(a)), fils_droit(a)]
        sinon
            renvoyer .....
        finsi
    finsi
fin
```

Exercice 3 :

Compléter la fonction ci-dessous qui permettra de construire l'arbre binaire de recherche de l'exercice 1.

```
def construction(liste) :
        #param : liste (list) une liste de nombres tous distincts
        #return : une arbre de recherche binaire
```

Remarque : J'ai déposé sur le site un fichier qui permet de visualiser l 'arbre construit.

Recherche du successeur d'un nœud :

Le **successeur** d'un nœud d'un ABR, s'il existe, est le nœud de cet arbre qui porte la clé la plus petite parmi les clés plus grandes que cette clé.

Réflexion :

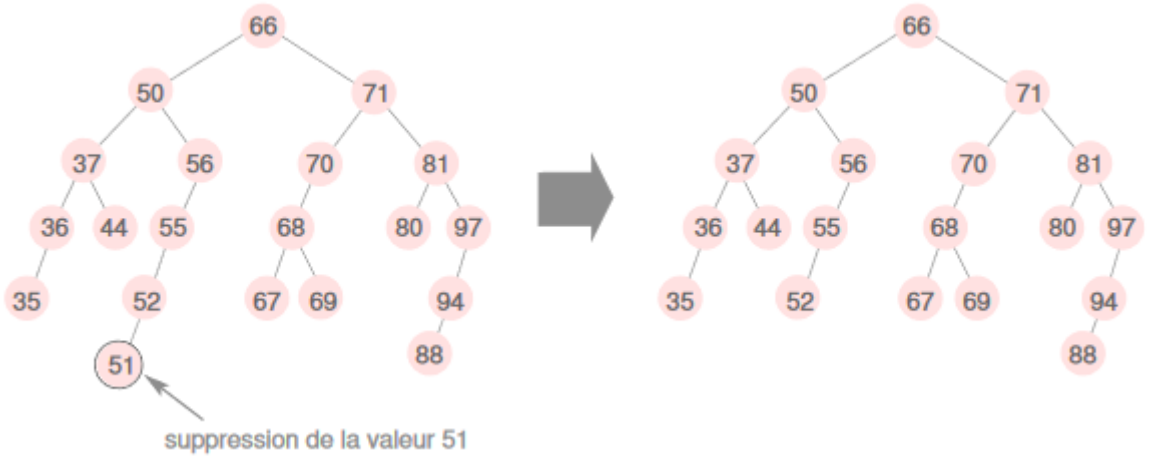
Où est situé le successeur d'un nœud qui a un fils droit ?

Où est situé le successeur d'un nœud qui n'a pas de fils droit ?

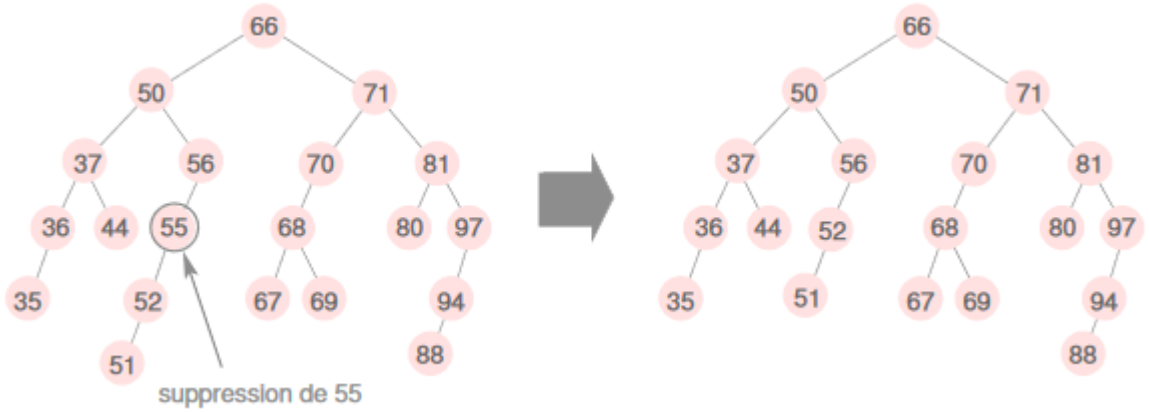
Suppression d'une clé dans un arbre binaire de recherche :

Cette opération dépend du nombre de fils qu'a le 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, s'ill existe, vers ce fils. Si le père n'existe pas l'arbre devient l'arbre vide.



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.



Cas 3 : le nœud à supprimer 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.