Chapitre 17 - Arbres binaires de RECHERCHE

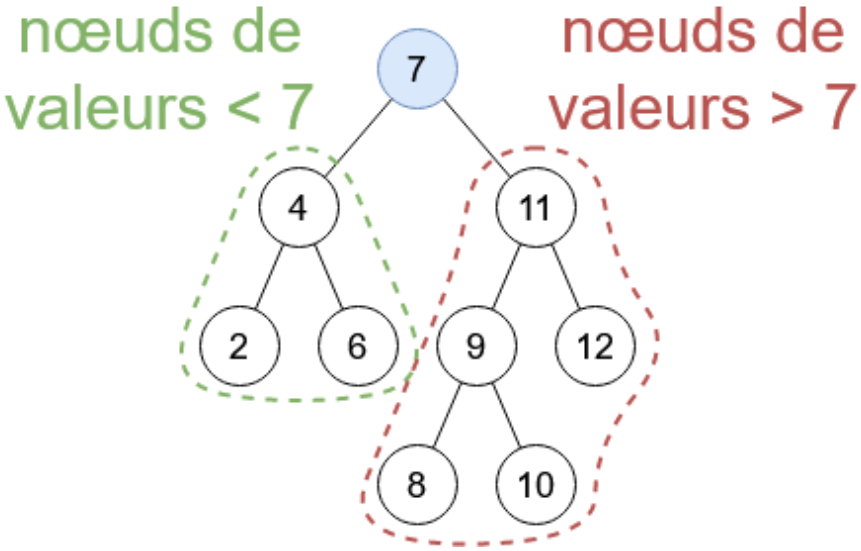
Les **arbres binaires de recherche** sont avant tout des arbres binaires. Ainsi toutes les techniques récursives ou itératives vues dans le chapitre précédent, s'appliquent à cette sous-famille d'arbres binaire. On voit dans ce chapitre des techniques supplémentaires qui sont spécifiques à ces ABR.

1- DEFINITION :

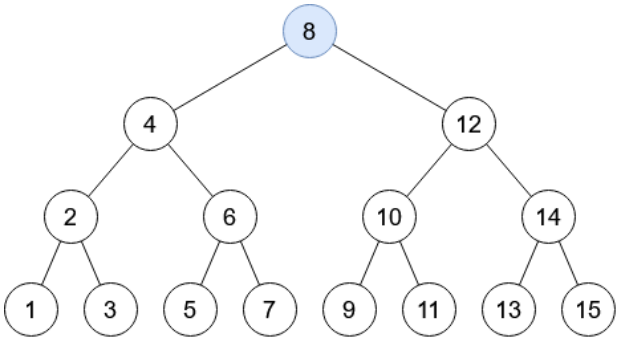
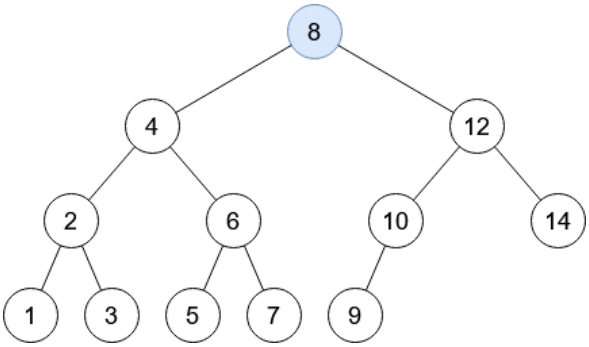
Un **arbre binaire de recherche** (**ABR**) (*binary search tree – BST*) est un arbre binaire tel que :

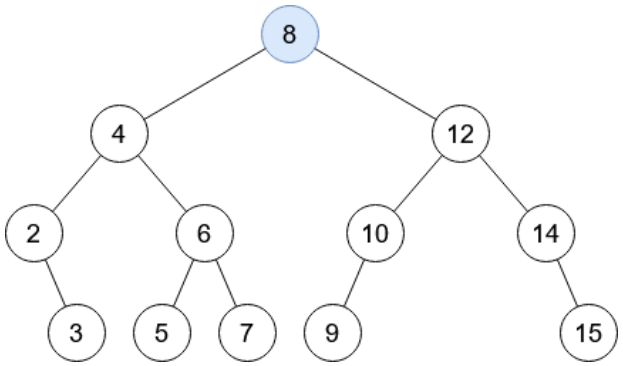
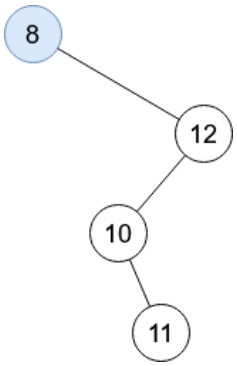
- les valeurs des nœuds doivent être **ordonnables** (il doit exister entre elles une relation d'ordre)
- pour chacun de ses nœuds:
  - chaque nœud du **sous-arbre gauche** a une valeur **inférieure (ou égale)** à celle du nœud considéré,
  - chaque nœud du **sous-arbre droit** possède une valeur **supérieure (ou égale)** à celle-ci

Par exemple, l'arbre ci-contre est un ABR :

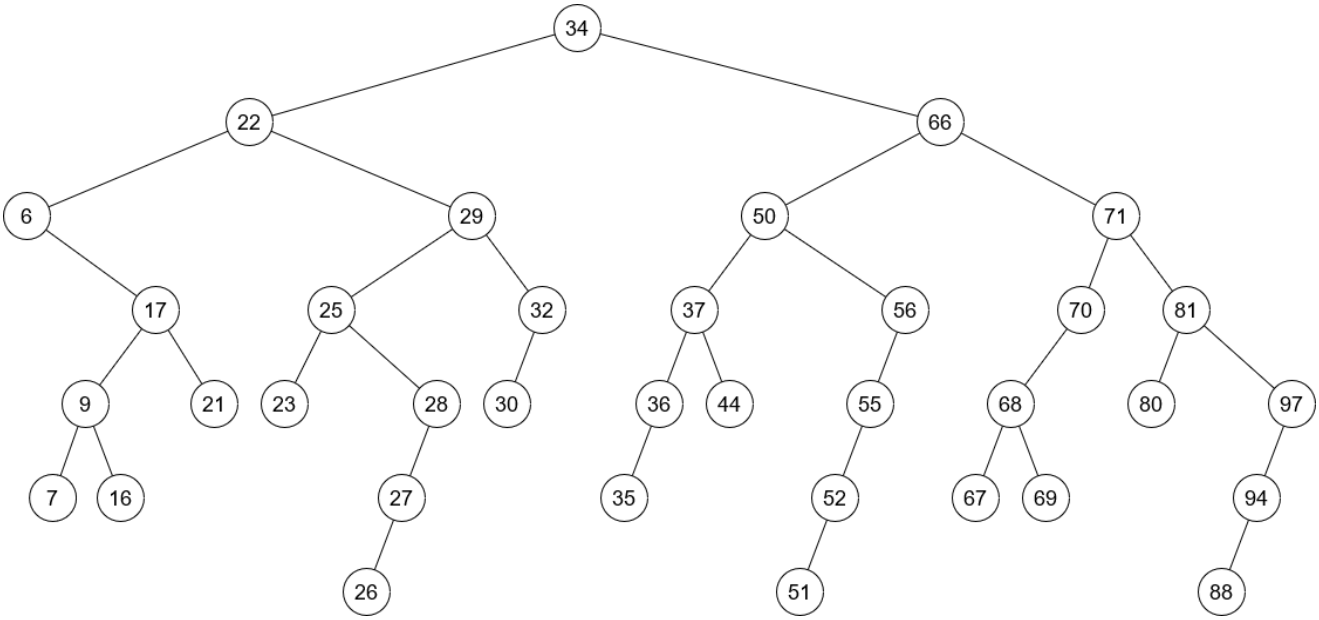


Comme pour les arbres binaires, les ABR peuvent être parfaits ou complet ou équilibrés ou dégénérés :

Arbre binaire **parfait** : arbre dont tous les niveaux sont remplis et dont toutes les feuilles sont à la même profondeur (le dernier niveau est complètement occupé).	Arbre binaire **complet** : tous les niveaux de l'arbre sont remplis, sauf éventuellement le dernier, sur lequel les nœuds (feuilles) sont à gauche.
	

Arbre binaire **équilibré** : tous les chemins de la racine aux feuilles ont la même longueur.	Arbre binaire **dégénéré** : chacun de ses nœuds a au plus un fils.
	

Exercice : L'arbre ci-dessous est-il un ABR ? Insérer dans cet arbre les valeurs 2 , 53 et 72



2- METHODES PROPRES AUX ABR :

Comme pour les arbres binaires simples, il existe plusieurs façons d'implémenter un ABR. Dans ce cours, on utilisera comme dans le chapitre précédent, une classe *Arbre*

```
class Arbre :
    def __init__(self, info = None , fg = None , fd = None) :
        self.info = info
        self.fg = fg
        self.fd = fd
```

a. METHODE POUR LIRE LES VALEURS MINIMALE ET MAXIMALE :

Point Cours :

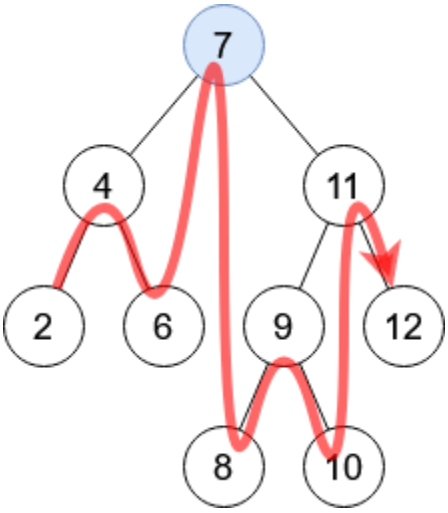
Sur un ABR, la valeur minimale se trouve facilement en partant du nœud racine et en parcourant les fils gauches successivement. Le dernier fils gauche visité a la valeur minimale de l'arbre. Pour trouver la valeur maximale, on parcourt successivement les fils droits.

b. METHODE POUR LIRE LES VALEURS TRIEES :

Point Cours :

Sur un ABR, Pour obtenir la suite ordonnée de toutes les valeurs d'un ABR, il suffit de le parcourir l'arbre dans l'ordre [redacted]

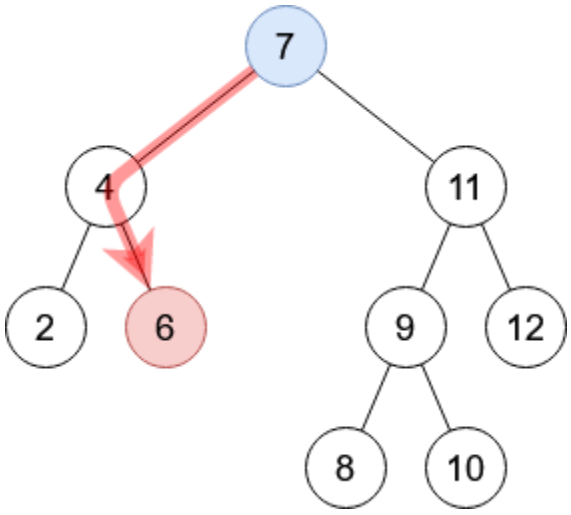
Exemple : Lecture des valeurs dans l'ordre croissant :



c. METHODE POUR RECHERCHER UNE VALEUR :

Une méthode de recherche prend en paramètre une valeur. Si celle-ci est présente dans l'arbre, la méthode retourne ***True***. Elle retourne ***False*** dans le cas contraire.

Exemple : Recherche de la valeur 6 :



Point Cours : Pour rechercher une valeur, on parcourt l'arbre en comparant cette valeur avec celle des nœuds successivement visités. Suivant le résultat de cette comparaison on s'oriente vers le fils gauche ou vers le fils droit. Arrivé en fin de parcours, si la valeur n'a pas été retrouvée, on retourne *False*.

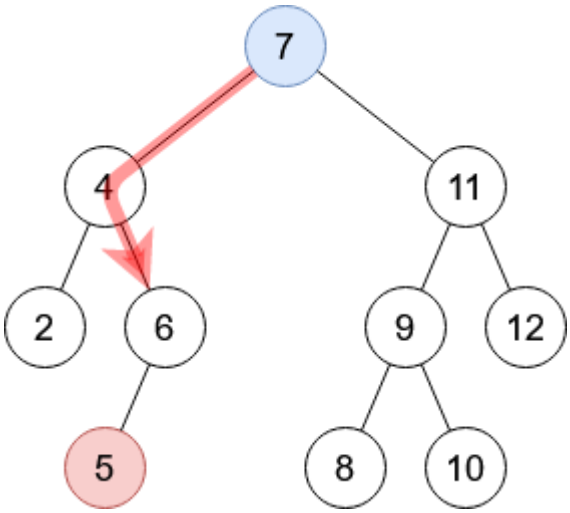
d. METHODE POUR INSERER UNE NOUVELLE VALEUR :

Une méthode d'insertion prend en paramètre une valeur. Elle modifie l'ABR en insérant en bout d'arbre, un nouveau nœud qui a comme valeur celle donnée. Si l'insertion s'est bien déroulée, la méthode retourne ***True***. Elle retourne ***False*** dans le cas contraire.

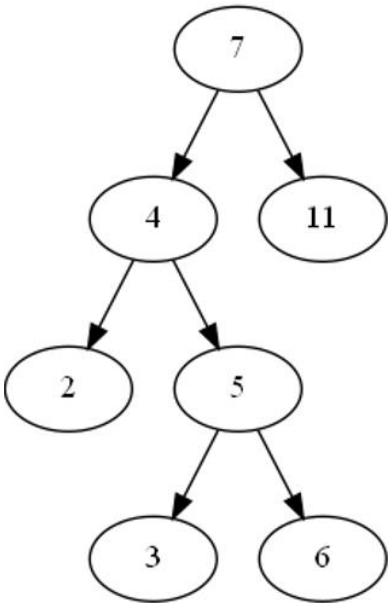
Point Cours : Pour insérer une nouvelle valeur :

- on recherche le nœud en bout d'arbre, à partir duquel on insérera la nouvelle valeur. On procède comme dans l'algorithme de recherche, sauf qu'ici le nœud recherché n'est pas sensé être trouvé,
- arrivé en bout d'arbre, on crée et on ajoute le nœud :
  - à gauche si la valeur est inférieure à celle de ce dernier noeud
  - ou à droite dans le cas contraire.

Exemple : Insertion d'un nouveau nœud dont la valeur est 5 :



e. METHODE POUR VERIFIER QU'UN ARBRE EST ABR :



La méthode vérifie toutes les valeurs présentent dans l'arbre et renvoie *True* si l'arbre est un ABR. Elle retourne *False* dans le cas contraire.

Par exemple, pour l'arbre donné ci-contre, la méthode retourne *False*.