Arbres Binaires de Recherche (ABR)

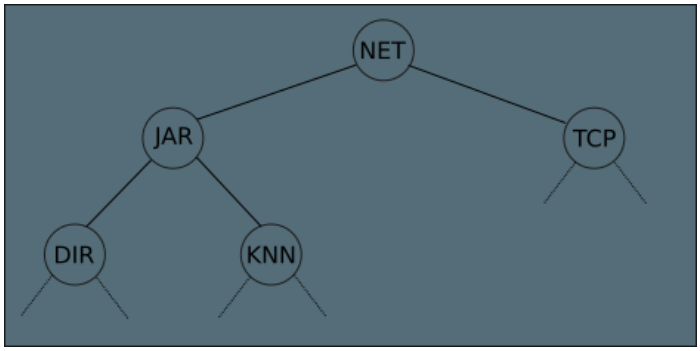
1. Premier Exemple

D'après *Numérique et Sciences Informatiques, 24 leçons avec exercices corrigés*, Balabonski, Conchon, Filliâtre, Nguyen, Editions Ellipses

Imaginez une bibliothèque contenant un très très grand nombre de livres. Cette bibliothèque est organisée de la manière suivante :

- Il y a 17 576 pièces différentes.
- Chaque pièce est repérée par une suite de trois lettres, et dans cette pièce sont rangés tous les livres dont les titres commencent par ces trois lettres.
- Chaque pièce possède deux sorties, une à droite et une à gauche.
- La sortie de gauche mène **toujours** soit à une salle dont les trois lettres sont situées avant dans l'ordre alphabétique, soit nulle part.
- La sortie de droite mène **toujours** soit à une salle dont les trois lettres sont situées après dans l'ordre alphabétique, soit nulle part.

Une représentation de cette bibliothèque peut être donnée sous la forme d'un rabre binaire tel que le suivant :



Question

Enoncé

1. Dans cet arbre, préciser où sont situés les livres dont le titre commence par :
   a. `KNU`
   b. `UDP`
   c. `JET`
2. Pourquoi y-a-t-il 17 576 pièces différentes ?

Réponses

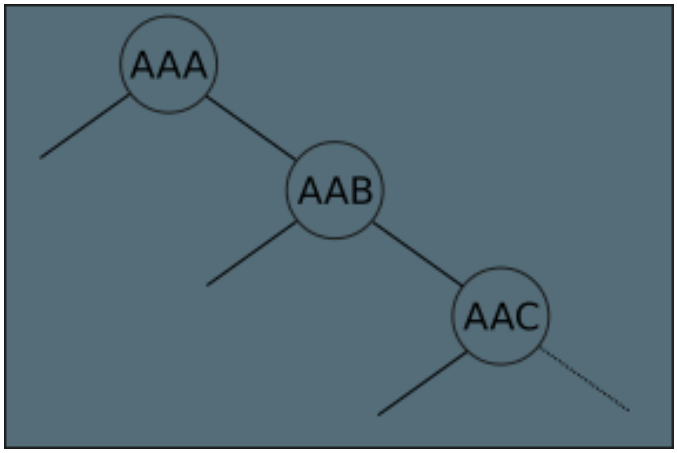
A venir !

Cette répartion, *pour peu qu'elle soit correctement faite* (c'est-à-dire que le choix des lettres soit pertinent), peut être incroyablement efficace. Dans **le meilleur des cas**, il ne faudra traverser qu'au maximum 15 salles pour trouver n'importe quel livre.

Cette structure sera particulièrement utile pour effectuer des recherches : on l'appelle ainsi un **arbre binaire de recherche** (ou **BST**, *Binary Search Tree* en anglais).

Une mauvaise répartition

L'importance de l'organisation des salles est ici primordial, toutes les solutions ne se valant pas. Ci-dessous une répartition qui est dans le pire des cas : les sous-arbres gauche sont toujours vide.



Dans cette situation, il faudra traverser les 17 576 pièces pour atteindre les livres dont le titre commence par `ZZZ`.

2. Arbres Binaires de recherches et algorithmes

2.1. Définition

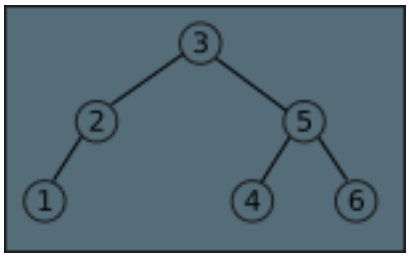
Arbre Binaire de Recherche (ABR)

Un **ABR** ou **Arbre Binaire de Recerche** est un arbre binaire vérifiant les propriétés suivantes :

- les noeuds contiennent des valeurs appelées **clés** pouvant être comparées entre elles (nombres, chaînes de caractères, ...) ;
- toutes les clés situées dans le **sous-arbre gauche** (resp. droit) d'un noeud sont **inférieures** (resp. supérieures) à la clé du noeud.

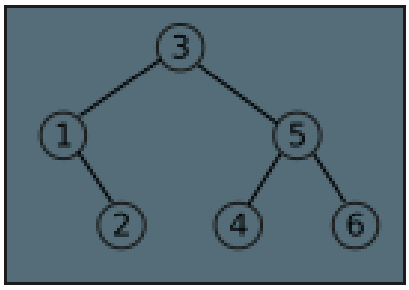
Exemples

Exemple 1



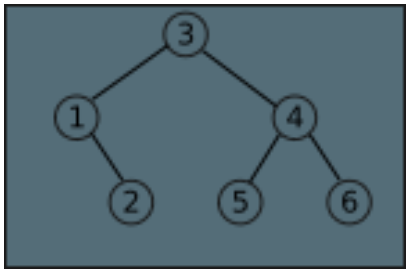
Il s'agit bien d'un ABR.

Exemple 2



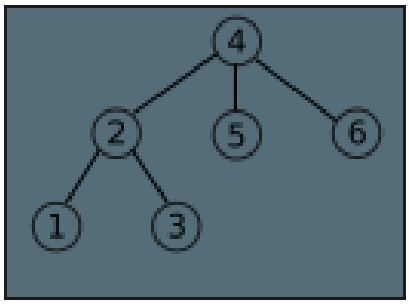
Il s'agit bien d'un ABR. On peut remarquer qu'il possède le même ensemble de noeuds que le précédent, mais pas dans le même ordre.

Contre-exemple 1



Ce n'est pas un ABR, le noeud 5 étant dans le sous-arbre gauche du noeud 4, alors que les sous-arbres gauches doivent posséder des clés de valeurs inférieures.

Contre-exemple 2



Ce n'est pas un ABR tout simplement parce qu'il n'est pas binaire.

2.2. Implémentation en Python

Il n'y a aucune différence entre un ABR et un arbre binaire en terme d'implémentation, on pourra donc conserver :

- les objets `Node` ;
- les fonctions `hauteur`, `taille` et `est_vide` ;
- les fonctions `visitePrefixe`, `visiteInfixe` et `visiteSuffixe`.

2.3. Recherches dans un ABR

Un ABR est spécialement conçu pour la recherche, particulièrement pour la recherche récursive. La méthdoe est la suivante :

> **Algorithme de recherche**
>
> On compare la valeur cherchée avec la clé de la racine :
>
> - si elle est égale, la valeur est trouvée et on peut renvoyer `Vrai` ;
> - si elle est inférieure, alors on recherche récursivement dans le sous-arbre gauche ;
> - si elle est supérieure, alors on recherche récursivement dans le sous-arbre droit.
>
> Si on atteint une feuille dont la clé n'est pas la valeur recherchée, on sait alors que la valeur recherchée n'est la clé d'aucun nœud, elle ne figure donc pas dans l'arbre de recherche. Pratiquement dès qu'on tombe sur un arbre vide, on peut affirmer que la clé n'est pas dans l'arbre, et on renvoie `Faux`

2.4. Ajout dans un ABR

2.5. Suppression dans un ABR

3. Arbre Binaires Equilibrés

3.1. Pourquoi équilibrer

3.2. Un exemple d'arbre binaires de recherches équilibrés : les AVL

Les AVL

Rotations droites et gauches

Application en Python