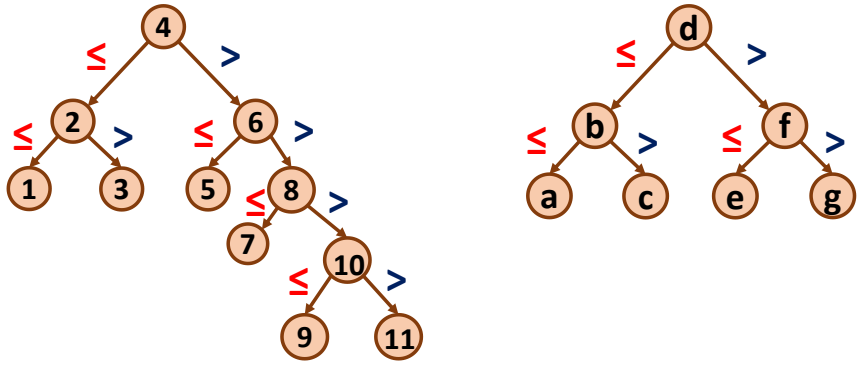
Arbres binaires de recherche

I./ Définition :

Un arbre binaire de recherche (ou ABR) est un arbre binaire dont les nœuds contiennent des valeurs qui peuvent être comparées entre elles, comme des nombres ou des chaînes de caractères, et tels que, pour tout nœud de l'arbre :

- Toutes les valeurs situées dans le SAG de ce nœud sont plus petites (ou égales) que la valeur du nœud considéré.
- Toutes les valeurs situées dans le SAD de ce nœud sont plus grandes que la valeur du nœud considéré.

Deux exemples d'ABR, un avec des nombres entiers (à gauche) et l'autre avec des chaînes de caractères (à droite) :

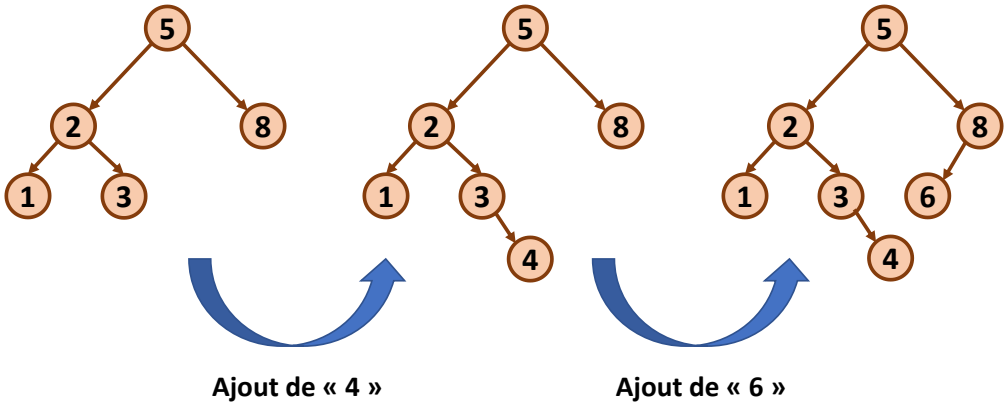


II./ Implémentation des ABR en Python :

Le but est de créer une classe « **ABR** » permettant de gérer les arbres binaires de recherche. Les ABR étant avant tout des arbres binaires, on peut reprendre la classe « **Arbre_binaire** » vue précédemment et la renommer « **ABR** », en gardant les différents « getters » et « setters », ainsi que les méthodes « **est_vide** », « **hauteur** » et « **taille** ».

On rajoutera deux méthodes :

- « **appartient(self, element)** », qui renvoie « **True** » si l'élément passé en argument est présent dans l'arbre, « **False** » sinon.
- « **ajoute(self, element)** », qui va rajouter l'élément donné en argument au bon endroit dans l'arbre (voir exemple ci-dessous) :



Le premier ajout (4) se fera obligatoirement dans le SAG du nœud racine « 5 », car 4 ≤ 5. Ce SAG a comme nœud racine « 2 ». Comme 4 > 2, l'ajout doit se faire dans le SAD du nœud « 2 ». Ce SAD a comme nœud racine « 3 ». Comme 4 > 3, l'ajout doit se faire dans le SAD du nœud « 3 ». Comme ce SAD est vide, on le remplace par un nœud contenant « 4 ». Le même raisonnement permet de comprendre le positionnement de l'élément « 6 » : comme 6 > 5, donc il faut l'insérer à droite de « 5 », et comme 6 ≤ 8, il faut l'insérer à gauche de « 8 ».

Quelques pistes :

- Pour implémenter la fonction « **appartient** » : si on compare la valeur de « **element** » à la valeur du nœud racine de l'arbre, on a trois cas de figures :
    - La valeur de « **element** » est égale à la valeur du nœud racine, dans ce cas on a trouvé « **element** ».
    - La valeur de « **element** » est inférieure à la valeur du nœud racine, dans ce cas si « **element** » est présent dans l'arbre, il est dans le SAG du nœud racine.
    - La valeur de « **element** » est supérieure à la valeur du nœud racine, dans ce cas si « **element** » est présent dans l'arbre, il est dans le SAD du nœud racine.
- Pour implémenter la fonction « **ajoute** », on peut de la même façon comparer la valeur de « **element** » à la valeur du nœud racine de l'arbre. Il n'y a ici que deux cas de figures :
    - La valeur de « **element** » est inférieure ou égale à la valeur du nœud racine, dans ce cas il faut ajouter « **element** » dans le SAG de l'arbre.
    - La valeur de « **element** » est supérieure à la valeur du nœud racine, dans ce cas il faut ajouter « **element** » dans le SAD de l'arbre.
- La description des deux opérations précédentes devrait vous convaincre de l'utilité d'utiliser la récursivité. Vous pourrez notamment vous inspirer des méthodes « **taille** » et « **hauteur** ».

Pour aller plus loin :

Implémenter une méthode « **affiche(self)** » qui permet d'afficher de façon sommaire l'ABR dans la console Python. Si on reprend l'exemple de l'ABR précédent :

```
>>> arbre = ABR(5)
>>> arbre = arbre.ajoute(2)
>>> arbre = arbre.ajoute(1)
>>> arbre = arbre.ajoute(3)
>>> arbre = arbre.ajoute(8)
>>> arbre = arbre.ajoute(6)
>>> arbre.affiche()
                              5

              2                               8

       1              3               6
```

On peut imaginer une représentation graphique (à l'aide par exemple du module « **turtle** ») mais cela peut devenir vite compliqué !

Remarque sur la suppression d'un nœud :

Afin d'être totalement complète, la structure de données ABR doit permettre à l'utilisateur de supprimer une valeur dans l'arbre. Cette opération est plus complexe (car il faut parfois modifier en profondeur la structure de l'arbre) et *n'est pas au programme de Terminale NSI*. Il est toutefois formateur d'essayer d'implémenter cette fonctionnalité.

2025
Bruno PEYROT