Projet : Arbre binaire de recherche

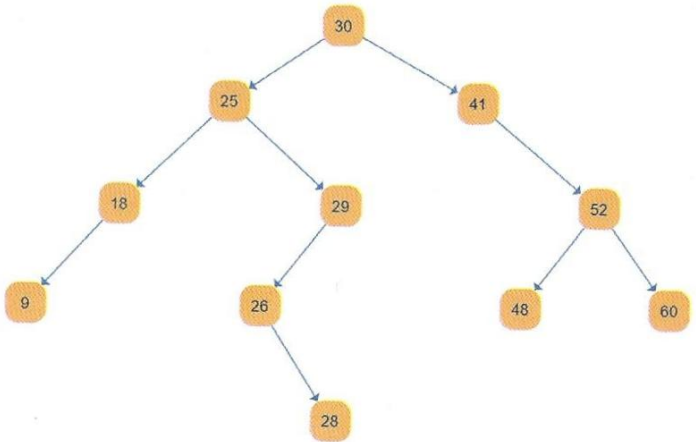
Un **arbre binaire de recherche (ABR)** est un arbre pour lequel l'étiquette d'un nœud est appelée **clé**. L'arbre binaire de recherche satisfait aux deux critères suivants :

- Les clés de tous les nœuds du sous-arbre gauche d'un nœud X sont inférieures ou égales à la clé de X.
- Les clés de tous les nœuds du sous arbre droit d'un nœud X sont strictement supérieures à la clé de X.

Remarque :

- Il en résulte que l'ensemble des clés est totalement ordonné.
- Une liste de nombres peut-être représentée par plusieurs ABR.
- Nous mettrons en œuvre ici des ABR avec des clés de type *int*, mais il serait possible de le faire sur toute structure possédant une relation d'ordre….

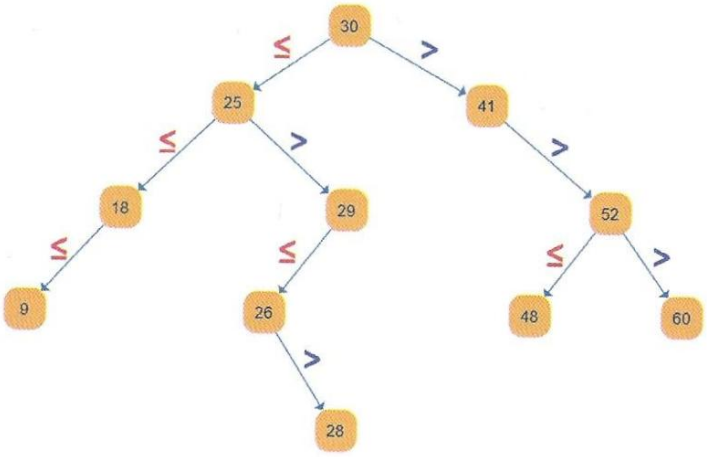
Voici un exemple d'arbre binaire de recherche :



Les nœuds sont insérés par comparaisons successives depuis la racine de l'arbre.

Il se lit comme si, sur chaque lien direct en provenance d'un nœud, il y avait le symbole :

- " $\leq$ " pour les liens situés à gauche
- " $>$ " pour les liens situés à droite.



Projet à rendre en binôme et à présenter à l'oral:

Dans un fichier *ABR_numgroupe.py* (important la classe BinTree implémentée dans la partie précédente), vous présenterez :

- Une modélisation de l'arbre présenté en exemple ci-dessus, support de vos tests.
- Une fonction ***estABR(A :BinTree)->bool*** qui renvoie True si A est un arbre binaire de recherche.
- Une fonction ***rechercheCle(A :Bintree,n :int)->bool*** qui renvoie True si n est une clé de l'arbre A.
- Une fonction ***insereCle(A,n :int)->BinTree*** qui renvoie un arbre dans lequel on a inséré le nœud n.
- Une fonction ***creerABR(valeurs :list)->BinTree*** qui construit un ABR à partir d'une liste d'entiers.
- Une fonction **sommeCle(A :BinTree)->int** qui renvoie la somme de toutes les clés de l'arbre A.

Remarque :

- N'ayant aucune contrainte sur le type N des étiquettes des nœuds de la classe BinTree, vous pourrez facilement créer un objet arbre dont les étiquettes sont des nombres entiers.
- Certaines fonctions nécessiteront la vérification de préconditions, que vous mettrez en œuvre à l'aide d'assertions (comme vues dans la partie A).
- Là encore, vous documenterez rigoureusement vos fonctions à l'aide de docstrings et commenterez votre code.
- Vous insèrerez des tests exécutables portants sur l'exemple précédent.
- Pour la conduite de ce projet, vous pourrez faire comme bon vous semble. Il pourra cependant être judicieux de définir un espace de partage. Vous pourrez ainsi utiliser **google drive** si vous disposez tous dans le groupe d'adresses gmail, ou **github** si vous souhaitez vous initier à cet outil (plus complexe) : https://www.christopheducamp.com/2013/12/15/github-pour-nuls-partie-1/

Vous rendrez compte en classe (à l'oral) de votre démarche et présenterez plus particulièrement vos algorithmes de recherche et d'insertion de clé. Vous pourrez biensur vous documenter sur les ABR, et présenter également les avantages et/ou limitations de cette structure de donnée.