Séquence 3 : TP 2 – Implémentation des arbres binaires avec des classes. Term NSI 2020/2021

Dans le TP précédent, nous avons représenté la structure d'arbre à l'aide d'une structure linéaire. Nous allons proposer dans ce TP une implémentation orientée objet du type abstrait arbre binaire d'éléments de type N. Comme dans le TP1, nous définirons également cette structure de manière récursive selon les définitions suivantes :

- Un arbre vide est représenté par un objet dont tous les attributs ont la valeur None
- Un objet de la classe BinTree possède :
    - Une racine *root* de type N
    - Un sous-arbre gauche *left* de type BinTree
    - Un sous-arbre droit *right* de type BinTree

Voici un diagramme représentant une première spécification de la classe BinTree :

Classe : BinTree
Attributs : *root* : type N (ou None) *left* : type BinTree (ou None) *right* : type BinTree (ou None)
Méthodes : Constructeur(root : N ,left : BinTree , right : BinTree) # Créer un arbre de racine root, de sous-arbre gauche left et de sous-arbre droit right Constructeur() # Créer un arbre vide estVide() :bool # Renvoie vrai si l'arbre est vide gauche() : BinTree # Renvoie le SAG left droit() : Bintree # Renvoie le SAD right racine() : N # Renvoie l'étiquette de la racine setGauche(B :Bintree) # Le SAG devient B setDroit(B :Bintree) # Le SAD devient B setRacine(r :N) # La valeur de la racine devient r taille() : int # Renvoie la taille de l'arbre hauteur() :int # Renvoie la hauteur de l'arbre view() # Fournit une vue lisible de l'arbre
Condition : La racine est définie si et seulement si l'arbre est non vide.

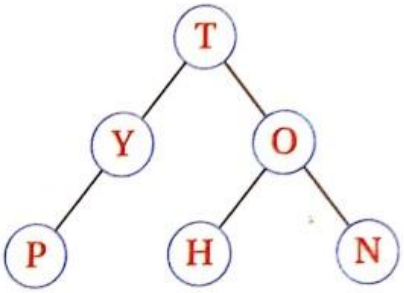
Partie A : Implémentation de la classe BinTree

1) Traduisez la condition portant sur les attributs de la classe BinTree à l'aide d'une expression booléenne.

   ……

2) Implémenter cette classe et ses méthodes en Python. On pourra biensur s'inspirer des fonctions programmées dans le cadre du TP1 et les adapter à la structure de cette classe.
   Vous sauvegarderez votre programme dans un fichier *BinTree.py* (nous serons amenés à réutiliser cette classe par la suite pour modéliser cette structure).
   Vous documenterez classes et méthodes à l'aide de docstring.

   Vous pourrez tester votre structure en utilisant l'arbre du TP1 :



Remarques :

- Deux constructeurs sont présentés dans le diagramme ci-dessus. Vous pourrez consulter la documentation suivante sur les arguments optionnels et nommés pour les mettre en œuvre : https://python.developpez.com/cours/DiveIntoPython/php/frdiveintopython/power_of_introspection/optional_arguments.php

- La condition sur les attributs pourra être vérifiée à l'aide des assertions (dans le constructeur). Au besoin, voici une documentation pour cette fonction : https://python.developpez.com/tutoriels/apprendre-programmation-python/notions-avancees/?page=gestion-d-erreurs#LIV-B

- La méthode view() pourra être remplacée par la méthode spéciale __str__ permettant d'obtenir une vue de l'arbre par appel de la fonction *print* appliquée à un objet de type BinTree. Voici une documentation pour cette méthode : https://allen-downey.developpez.com/livres/python/pensez-python/?page=classes-et-methodes#L17-6

- N'hésitez pas à utiliser http://www.pythontutor.com/ pour visualiser l'exécution de votre programme, en observant notamment la structure d'arbre mise en œuvre avec des objets.

3) A ce stade, vous pouvez ajouter à votre guise les autres méthodes étudiées dans le TP1 à votre classe.

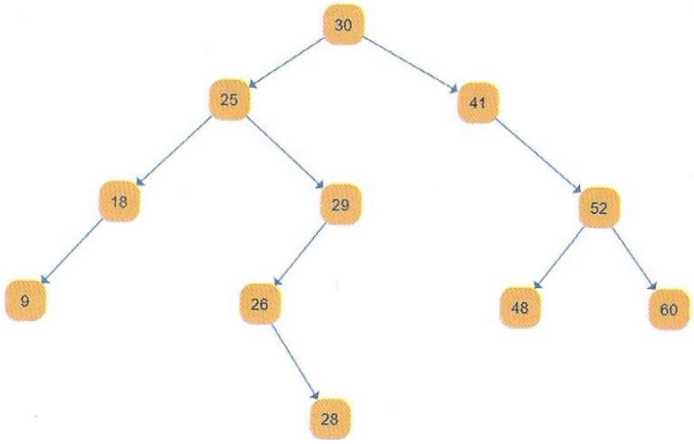
Partie B : Arbre binaire de recherche.

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.

Remarques :

- 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….

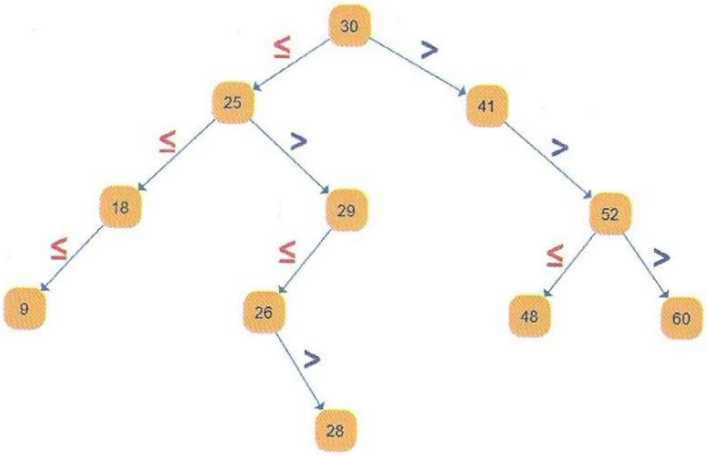
Voici un exemple d'arbre binaire de recherche :



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 trinôme (par mail Ecole Directe) et à présenter à l'oral:

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

- 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.
- Une modélisation de l'arbre présenté en exemple ci-dessus, support de vos tests.

Remarques :

- 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 (tous) vous initier à cet outil (plus complexe) : https://www.christopheducamp.com/2013/12/15/github-pour-nuls-partie-1/

Vous renderez 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.