TP 8 - Arbre d'Huffman

L'objectif est de développer le module `arbre_huffman.py`.

1 Définition

Un arbre d'Huffman est un arbre binaire non vide tel que :
— les feuilles possèdent deux informations : un élément et un nombre d'occurrences ;
— les noeuds possèdent, en plus d'un fils gauche et d'un fils droit, un nombre d'occurrences dont la valeur est égale à la somme des nombres d'occurrences des racines des deux fils.

2 Exceptions

Le module `arbre_huffman.py` propose les exceptions :
— `ArbreHuffmanErreur` sous classe de la classe `Exception` ;
— `DoitEtreUneFeuilleErreur` sous classe de la classe `ArbreHuffmanErreur` ;
— `NeDoitPasEtreUneFeuilleErreur` sous classe de la classe `ArbreHuffmanErreur` ;
— `ArbreHuffmanIncoherentErreur` sous classe de la classe `ArbreHuffmanErreur`.

3 Classe `ArbreHuffman`

Le module `arbre_huffman.py` propose aussi le classe `ArbreHuffman`. Cette classe possède :
— la méthode `__init__(self, element=None, nb_occurrences=None, fils_gauche=None, fils_droit=None)` tel que les paramètres formels `element` et `nb_occurrences` permettent de créer une feuille et, `fils_gauche` et `fils_droit` permettent de créer un noeud. L'utilisation des deux premiers paramètres formels exclue l'utilisation des deux autres et inversement, si tel n'est pas le cas une exception `ArbreHuffman IncoherentErreur` est levée. De plus cette même exception est levée si le fils gauche et le fils droit sont identifiques.
— la propriété `est_une_feuille` qui permet de savoir si l'arbre est une feuille ;
— la propriété `nb_occurrences` qui permet d'obtenir le nombre d'occurrences de la racine de l'arbre ;
— la propriété `element` qui permet d'obtenir l'élément d'une feuille. Cette propriété est susceptible de lever l'exception `DoitEtreUneFeuilleErreur` ;
— les propriétés `fils_gauche` et `fils_droit` qui permettent d'obtenir les fils gauche ou droit d'un arbre. Ces propriétés sont susceptibles de lever l'exception `NeDoitPasEtreUneFeuille Erreur` ;
— la méthode `equivalent(self, autre)` qui retourne `True` lorsque les deux arbres sont structurellement équivalent ;
— les méthodes de comparaison (`__gt__`, `__ge__`, `__lt__`, `__le__`) : un arbre de huffman `a1` est plus petit que un arbre `a2` si `a1.nb_occurences` $<$ `a2.nb_occurences` ;

— la méthode `__add__` qui permet de créer un arbre d'huffman à partir de deux arbres d'huffman (l'opérande gauche représente le fils gauche du nouvel arbre, l'opérande droit, le fils droit) ;
— les méthodes de représentations formelles et informelles.

Vous testerez votre module à l'aide du test unitaire pytest fourni. Vous veillerez à ce que le score de pylint3 soit le plus grand possible (idéalement de 10/10).